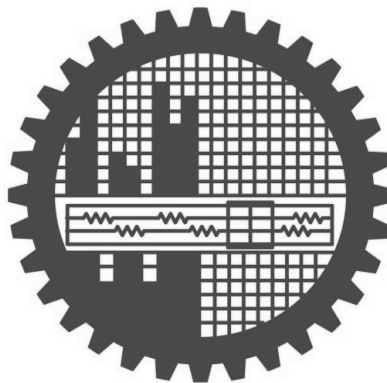


Real time Train, Vehicle and Pedestrian Detection System for Automated Railway Level Crossing

By

Jiban Mahmud Joy

POST GRADUATE DIPLOMA IN INFORMATION AND COMMUNICATION
TECHNOLOGY



Institute of Information and Communication Technology
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Dhaka-1000

March 2022


The project titled "**Real time Train, Vehicle and Pedestrian Detection System for Automated Railway Level Crossing**" submitted by Jiban Mahmud Joy, Roll No. 1018311029, Session October, 2018 has been accepted as satisfactory of the requirement for the degree of Post Graduate Diploma in Information and Communication Technology is on 9 April, 2022.

BOARD OF EXAMINERS



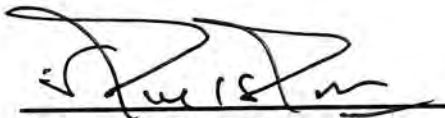
1. Mohammad Muntasir Hassan
Lecturer
IICT, BUET, Dhaka-1000.

Chairman



2. Dr. Md. Rubaiyat Hossain Mondal
Professor
IICT, BUET, Dhaka-1000.

Member



3. Dr. Md. Saiful Islam
Professor
IICT, BUET, Dhaka-1000.

Member

Candidate Declaration

It is hereby declared that this project or any part of it has not been submitted elsewhere for the award of any degree or diploma.



Jiban Mahmud Joy

Dedicated
To
My Parents and Daughter

Table of Contents

Title	Page No.
Board of Examiners.....	ii
Candidate Declaration.....	iii
Table of Contents.....	v
List of Figures.....	vii
Acknowledgement.....	viii
Abstract.....	ix

CHAPTER ONE

1. Introduction.....	1
1.1 Background and present state of the problem.....	1
1.2 Objectives with specific aims.....	2
1.3 Possible outcome.....	2
1.4 Organization of the project.....	3

CHAPTER TWO

2. Literature Review.....	4
2.1 Detection system.....	4
2.2 Types of algorithms of object detection system.....	4
2.2.1 Fast R-CNN.....	4
2.2.2 Faster R-CNN.....	5
2.2.3 Histogram of oriented gradients (HOG).....	6
2.2.4 Region-based convolutional neural networks (R-CNN).....	7
2.2.5 Region-based fully convolutional network (R-FCN).....	8
2.2.6 Single shot detector (SSD).....	10
2.2.7 Spatial pyramid pooling (SPP-net).....	11
2.2.8 YOLO.....	12

CHAPTER THREE

3. Methodology.....13
 3.1 Beginning.....13
 3.2 Analysis of the proposed system.....13
 3.3 Block diagram of the system.....13
 3.4 Platform ready for our system.....14
 3.5 Outline of methodology/ experimental design.....15

CHAPTER FOUR

4. Results and Discussions.....17
 4.1 Introduction.....17
 4.2 Result and discussions.....17
 4.3 Output.....17
 4.3.1 Train detection.....17
 4.3.2 Pedestrian detection.....21
 4.3.3 Bus detection.....21
 4.3.4 Truck detection.....22
 4.3.5 Car detection.....23
 4.4 Real time train detection.....23

CHAPTER FIVE

5. Conclusion.....24
 5.1 Conclusion.....24
 5.2 Suggestions for future works.....24

References.....26

Appendix.....29

List of Figures

Title	Figure Caption	Page No.
Figure 2.1:	Fast R-CNN network.....	5
Figure 2.2:	The architecture of Faster R-CNN.....	6
Figure 2.3:	Histogram of oriented gradients (HOG).....	7
Figure 2.4:	Architecture of the region based convolutional neural network.....	8
Figure 2.5:	Region based fully convolutional networks R-FCN architecture.....	9
Figure 2.6:	Single shot detector (SSD).....	10
Figure 2.7:	Spatial pyramid pooling (SPP-net).....	11
Figure 2.8:	YOLO architecture.....	12
Figure 3.1:	Block diagram of the proposed system.....	14
Figure 4.1 (a):	Long distance real time train detection via webcam.....	18
Figure 4.2 (b):	Real time train detection at a time two trains via webcam.....	18
Figure 4.3 (c):	Real time train detection when train overlap via webcam.....	19
Figure 5:	Dusty weather train detection.....	19
Figure 6:	Foggy weather train detection.....	20
Figure 7:	Rainy weather train detection.....	20
Figure 8:	Pedestrian detection...../.....	21
Figure 9:	Bus detection/.....	22
Figure 10:	Truck detection...../.....	22
Figure 11:	Car detection...../.....	23

Acknowledgement

First of all, I express my gratefulness to the Almighty Allah for enabling me to perform this task successfully. I would like to express my deepest sense of gratitude to my honorable supervisor, Mohammad Muntasir Hassan, Lecturer, Institute of Information and Communication Technology (IICT), Bangladesh University of Engineering and Technology (BUET), for his scholastic supervision, valuable guidance, adequate encouragement and helpful discussions throughout the progress of this work and for giving me the opportunity to conduct this project. I am highly grateful to him for allowing me to pursue this study under his supervision. I am grateful to pursue my project under his supervision. Without his assistance, this project would not have succeeded.

I would like to convey my thanks to Professor Dr. Md. Rubaiyat Hossain Mondal, Director and Dr. Md. Saiful Islam, Members, IICT, BUET. Their insightful suggestions and inspirations gave me the courage to do this work.

I gratefully acknowledge the restless support and advice of my fellow classmate and friend during the design and implementation phase of this project. I want to thanks to my project supervisor, Mohammad Muntasir Hassan for giving his valuable time and helping while I was buying the apparatus for my project. My special thanks to all the teachers, students and staffs of IICT, BUET.

Finally, I am much grateful to my family members especially to my parents, all of my friends and well-wishers for their encouragement and supports.

Author

Abstract

In this project, we have analyzed images and videos of the train from different angles in different train stations of Dhaka in different environmental conditions. With these recorded videos we have developed a real time train detection system. We have used OpenCV, SSD model, and frozen inference graph for increased detection speed. In addition, the system can perform dynamic detection of vehicles and pedestrians. In this detection system, we compare the real time image with the model images containing for vehicle detection with preset threshold values. If the resultant value is higher than the threshold value, then the system detects the presence of the object i.e., vehicles or pedestrians. In the beginning we worked with images to make this detection system. After being successful there, we shifted to the moving images i.e., video. After successfully analyzing the images from the video, we started working with real-time video footages. In the case of real-time videos, many frames come together, and they are converted into images, and are matched with the model of the detection system. With higher frame rates, the system performs slower. That is why we optimized the frame rate so that the system can operate in real time with adequate accuracy. Although we started working with YOLO model, later we shifted to the Single Shot Detector (SSD) model as the later was faster and more accurate. The proposed system can enable an automated level-crossing system in Bangladesh for which accurate and faster detection of the train is a key component.

CHAPTER ONE

Introduction

There are many unprotected rail crossings in Bangladesh and collision occurs between the train and other vehicle in these crossings very frequently. The problem is especially severe in rural and sub-urban areas where the level-crossings are unmanned and unprotected. In these unfortunate cases, a number of people perish every year, and many properties are damaged. Considering these situations, we have taken a project to detect trains in real-time which is a key aspect in automated level-crossing system. The inputs are images, video and real time camera data. In this detection system, we have compared the real time image with preset threshold values. The threshold value is carefully set for successful and accurate detection of the train. If the resultant value is higher than the threshold value, then the train is detected otherwise the system assumes that there is no train. In the beginning we have worked with images to make this detection system. After being successful there, we have started working with videos. After being successful, we have worked with real-time footages from the camera. In the case of video and camera, many frames come together. By converting them into images we have matched the output with the image detection system. In this process, the system becomes a bit slow. That is why we have worked by removing some frames so that the system can work in real-time without compromising accuracy. This system was made with the YOLO model. Later we had used the Single shot detector (SSD) model instead of the YOLO model to make the detection faster [5]. The threshold value was optimized to exclusively detect the desired object like trains, other vehicles or pedestrians.

1.1 Background and present state of the problem

According to a 2021 data there are 964 unmanned railway level crossings in Bangladesh and serious accidents occur frequently in these rail crossings [6]. Automated rail crossing system can aid in this respect to prevent this unwanted loss of lives and resources. An essential part of automated level crossing system is to detect the train for which several techniques are available and commercially employed [3,5,7]. As an alternative, we are proposing here real-time train detection from the live feed employing OpenCV (version 4.5.2) [7] that is based on

convolutional neural network [8,5]. Moreover, this method can detect other vehicles and pedestrians as well which is necessary to ensure that the railway line is free from potential obstacles. Here the challenges are to detect the desired objects in real time with limited computational resources and to ensure accuracy. In recent times, different object detection methods have garnered tremendous improvement, but the problem remains unresolved in uncontrolled places, especially when objects are placed indiscriminately in a chaotic and confined environment. In this regard, a real time object detection system especially for trains and other vehicles is essential for automated railway level crossing.

1.2 Objectives with specific aims

The objective of this project is to analyze different images to detect train, other vehicles and pedestrians and generate unique signal for each of them. To achieve this goal, we have determined the following objectives

- i. Detection of the train, other vehicles and pedestrians using coco.names data set and OpenCV SSD mobile net version 3
- ii. To achieve faster detection from real time video feed by employing different frame rates without compromising accuracy.
- iii. To detect the desired objects in foggy, dusty and rainy water.

1.3 Possible outcome

Successful completion of the project will assist in automated railway level crossing system and instigate further research in this field [3]. Considering this detection system first of all we have classified the image. In this case here are also training images of the system. Secondly we have focused object classification and localization. In this case the image has a background part, but our system only detects our specified object that is called localization. Third is multiple objects detection and localization. In this case the image there are many objects at a time in one image. They are separately detected and localized separately. The traditional object detection model there are three stages one is informative region selection another is feature extraction and one another is classification. Region proposal based framework is composed of three correlated stages, including region proposal generation, feature extraction with CNN [9], classification and

bounding box regression, which are usually trained separately. Regression or classification based framework is one-step frameworks based on global regression or classification, mapping straightly from image pixels to bounding box coordinates and class probabilities, can reduce expense. Two significant frameworks one is you only look once (YOLO), and another is single shot multi box detector (SSD). We have used SSD of our system to detect object.

1.4 Organization of the project

Chapter 1 of this report describes the introduction and relevant background of this project.

Chapter 2 of this report, issues from all aspects of developing the proposed system has been reviewed from the literature.

Chapter 3, methodologies of the proposed system has been discussed in detail.

Chapter 4 step by step design and development and obtained results of the proposed system have been discussed.

In the final chapter (Chapter 5), conclusion and recommendation for future works has been stated. The project report ends with an appendix A that contains the program code of the system.

CHAPTER TWO

Literature Review

2.1 Detection system

One of the main safety requirements of a train system is that before a train is moving along a section of train line, there should be no other traffic. Thus, the ability to detect the presence of a train on a particular stretch of track is a key enabling for automated level crossing system and automatic signaling, and it is an integral part of modern train control.

2.2 Types of algorithms of object detection system

1. Fast R-CNN
2. Faster R-CNN
3. Histogram of oriented gradients (HOG)
4. Region-based convolutional neural networks (R-CNN)
5. Region-based fully convolutional network (R-FCN)
6. Single shot detector (SSD)
7. Spatial pyramid pooling (SPP-net)
8. YOLO, etc.

2.2.1 Fast R-CNN

Fast R-CNN is fast region based convolutional neural networks and they are known to machine learning models for computer vision and specifically for object detection [10]. RPN (Region proposal network) try to find out this area where the object can be possibly found. Finally, we have got the area where the object is present. Then we have labeled that area as a foreground class where we have got an object in an image we have identified the label that area as a foreground class. In figure 2.1, the area where the object is not present showing the area [11]. All area where the object is not present this area should get label as background class. The task of RPN is to predict foreground and background anchor boxes and finally the anchor boxes which are labeled as foreground class got the next stage.

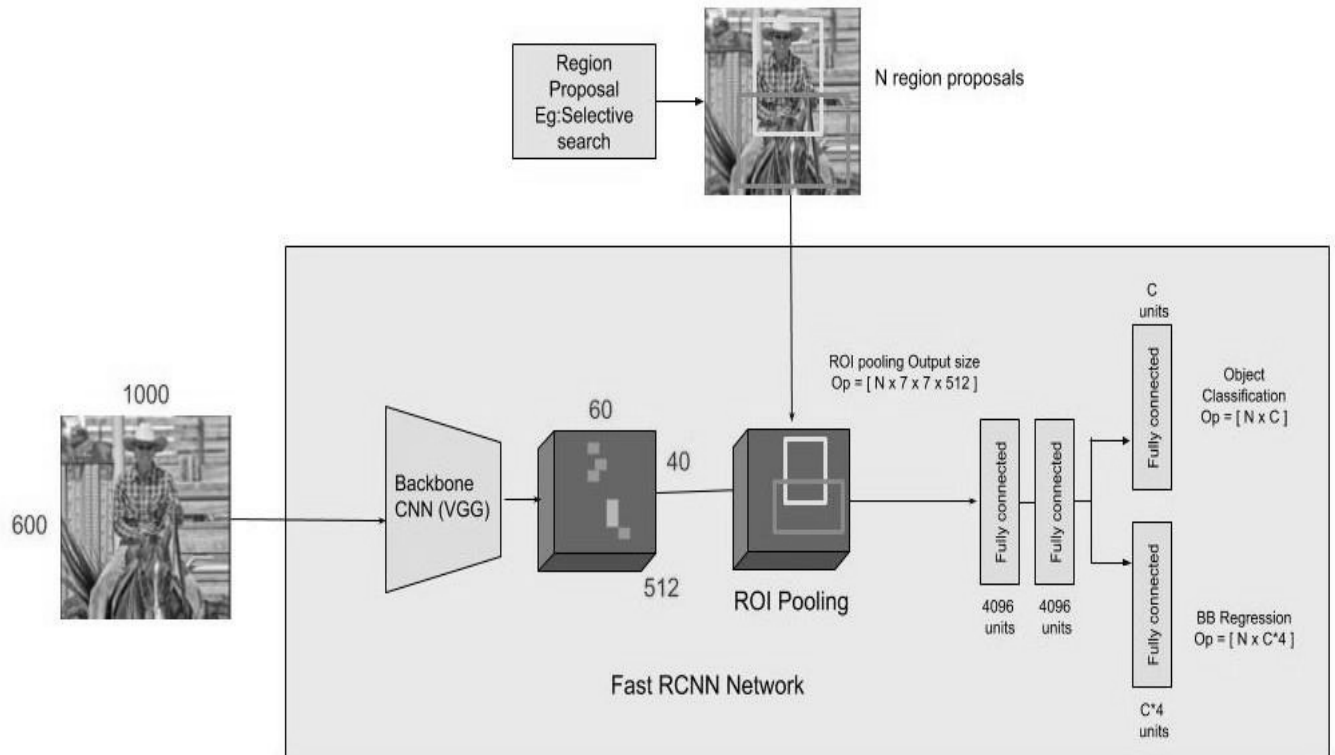


Figure 2.1: Fast R-CNN network [11]

2.2.2 Faster R-CNN

Faster R-CNN means fast region based convolutional neural networks are known to machine learning models for computer vision and specifically of object detection [12]. The area of the object in the picture can be found where object can be found if you see the picture where the object can be possibly found. In this stage have showed the output of this RPN region of the object [13]. The output of the object is the anchor boxes which are labeled as foreground class. This is also a sub CNN network where some convolutional neural network tasks are performed. It can work as a feature map. Another stage is ROI (Region of interest) pooling.

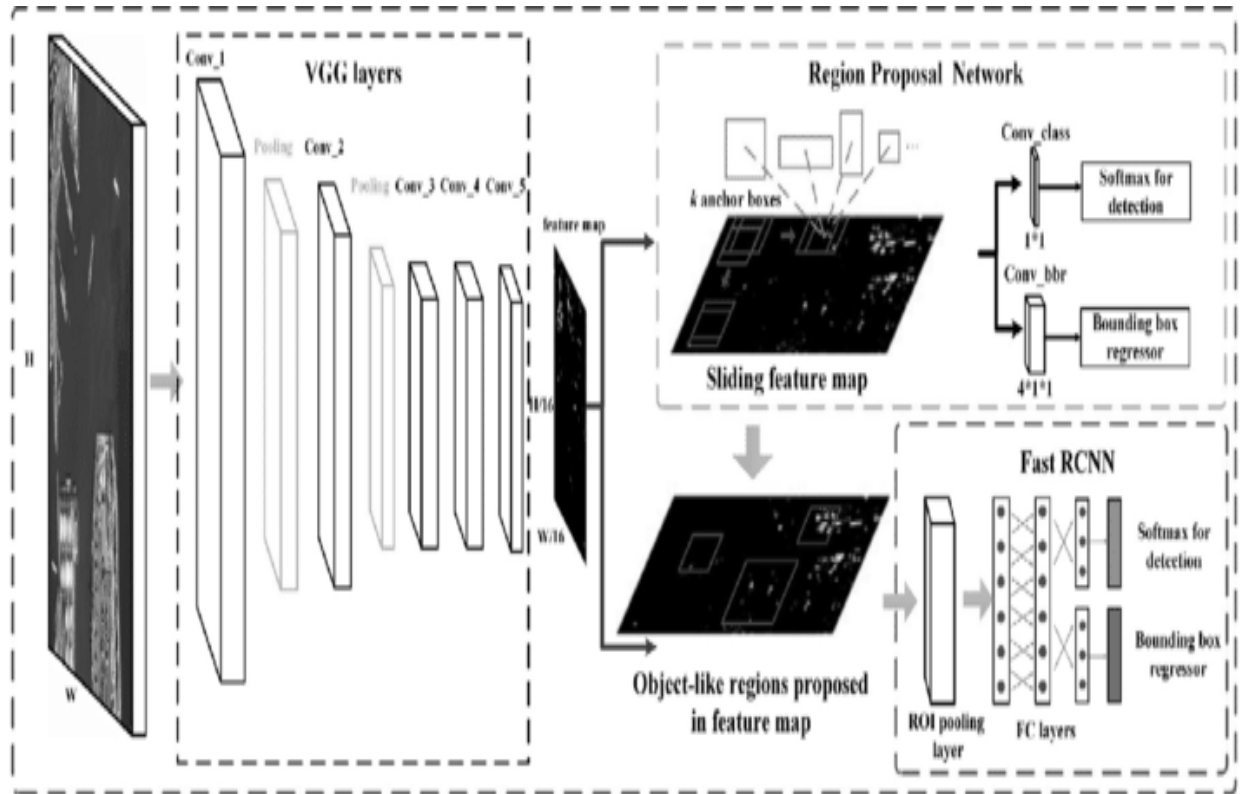


Figure 2.2: The architecture of Faster R-CNN [13]

2.2.3 Histogram of oriented gradients (HOG)

HOG is a feature of utilizing to detect an object in image processing and other computer vision techniques [14]. This stage is changed the image vertical and horizontal change. The whole image divided into 16×16 blocks of 50 percentage overlapping. The process of detection object is same as fast R-CNN and Faster R-CNN.

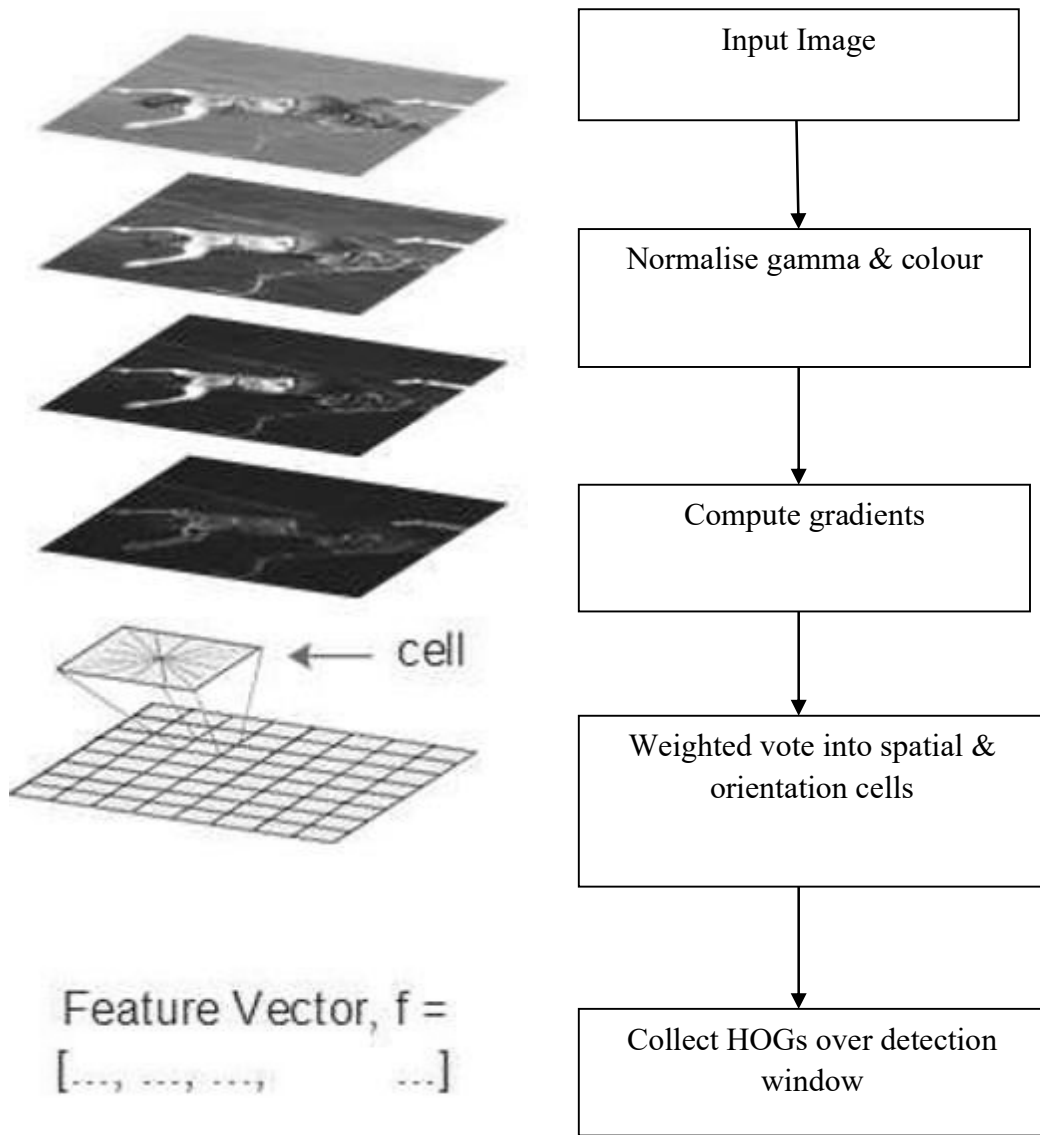


Figure 2.3: Histogram of oriented gradients (HOG) [15]

2.2.4 Region-based convolutional neural networks (R-CNN)

The R-CNN is a technique of combination of region proposals with CNNs [16]. It is the selective search that means an image where we have found some objects. Selective search is a color changing technique which portions change the color and that portion contains an object. All of that similar color is the selected bounding box that the region of the object.

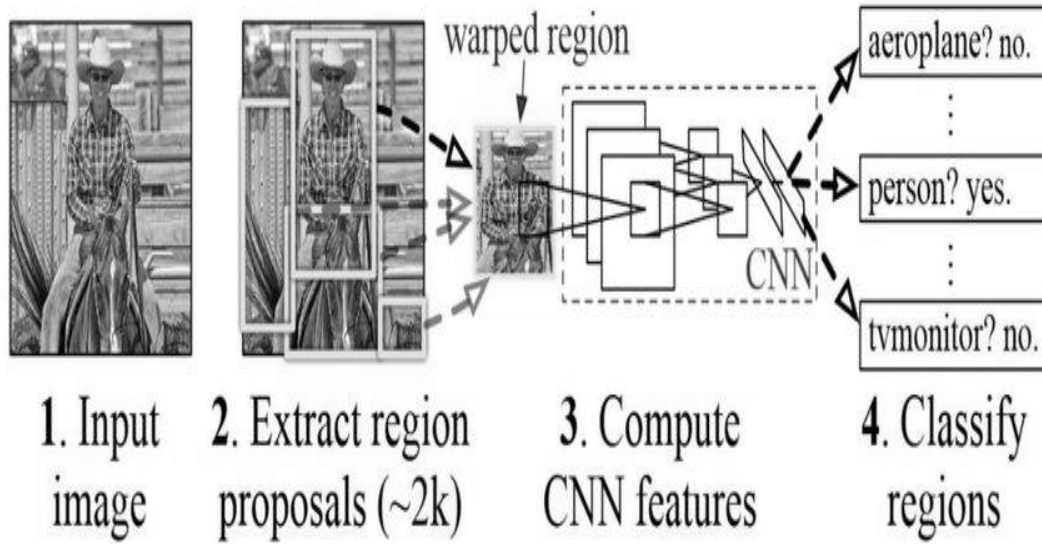


Figure 2.4: Architecture of the region based convolutional neural network [17]

2.2.5 Region-based fully convolutional network (R-FCN)

R-FCN is known for machine learning models for computer vision and specificity of object detection. This detection process is a regionally based fully convolutional network for accurate and efficient object detection [18]. This method uses position sensitive score map to solve two issues. First are the translation and the variance issue in image classification. And second is the translation variance issue in object detection.

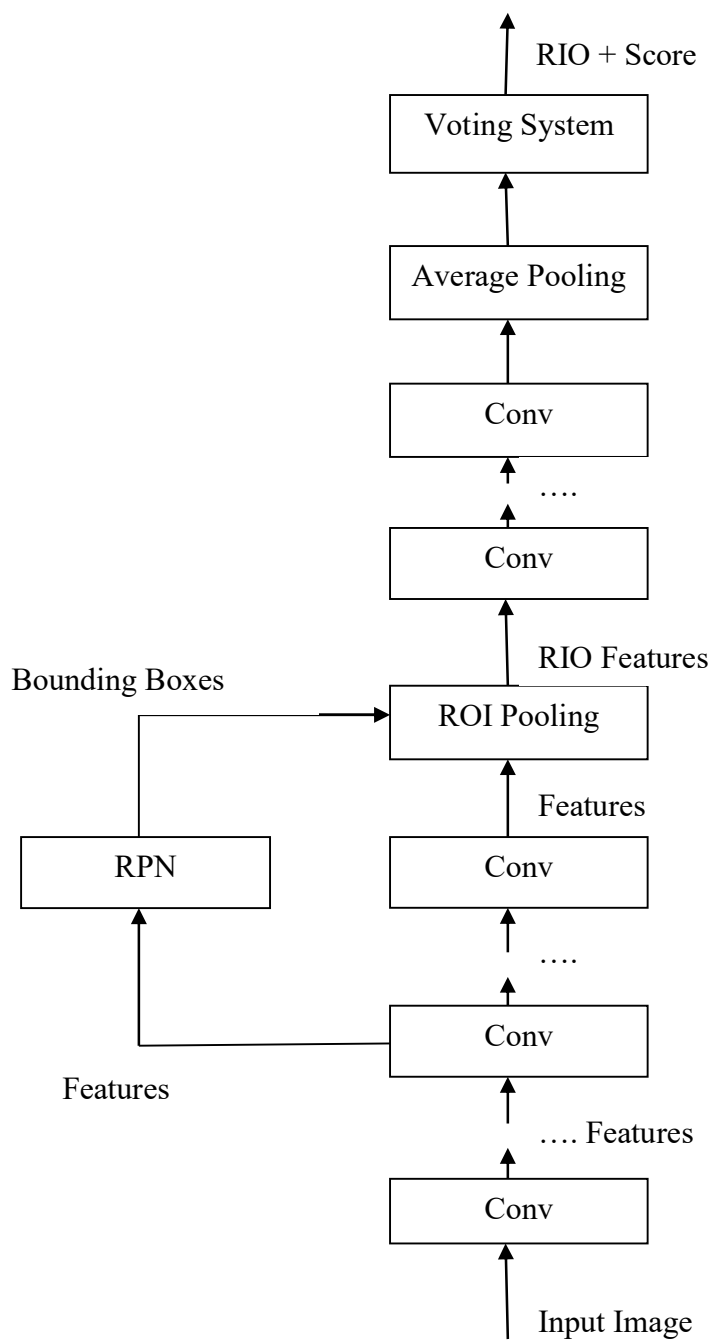


Figure 2.5: Region based fully convolutional networks R-FCN architecture [19]

2.2.6 Single shot detector (SSD)

SSD is a technique for detecting objects in an image using a single deep neural network [20]. This approach discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios of the output objects ratio. Discrediting, the method scales per feature map of the object location. The SSD network combines the predictions from multiple features of maps with different resolutions to naturally handle objects of various sizes of the objects. Our proposed method is flowed this process.

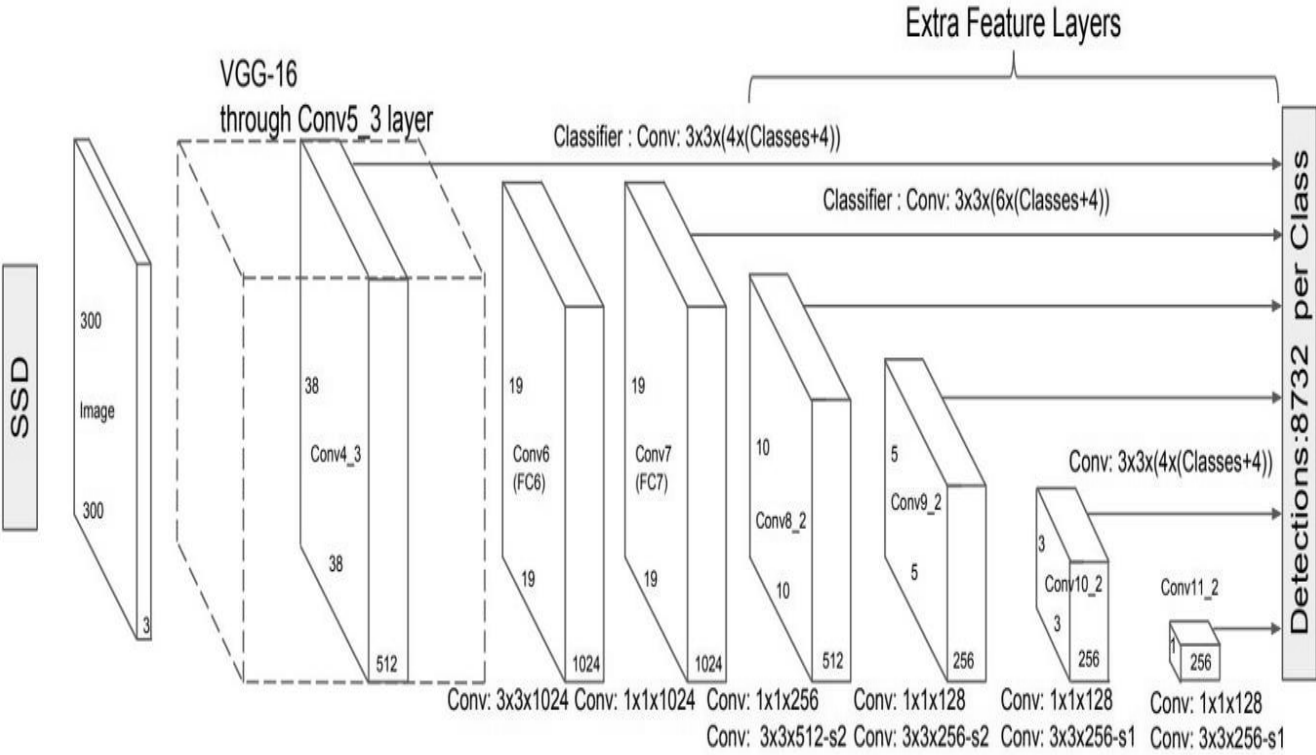


Figure 2.6: Single shot detector (SSD) [21]

Advantages of SSD

SSD completely eliminates proposal generation and subsequent pixel or feature resembling stages and encapsulates are in a single deep network [22]. Easy to detect trains and straightforward to integrate into the systems that requires a detection component. SSD has

competitive accuracy of methods that utilize an additional object proposal step, and it is much faster while providing a unified framework for both training and inference.

2.2.7 Spatial pyramid pooling (SPP-net)

SPP-net is a network structure that can generate a fixed-length representation regardless of image size or scale [23]. Aggregate pooling is also robust to object distortions. With these advantages, SPP-net should in general ameliorate all CNN- grounded image bracket methods. The power of SPP-net is also significant in object discovery. Using SPP-net, we have ciphered the point maps from the entire image only formerly, and also pool features in arbitrary regions (sub-images) to induce fixed-length representations for training the sensors. Precisely divide each walnut half into equal slices. Use shanks wisely, so they do not fall piecemeal. Place a slice of walnut on top of each cut date. Sprinkle it with cinnamon greasepaint and serve.

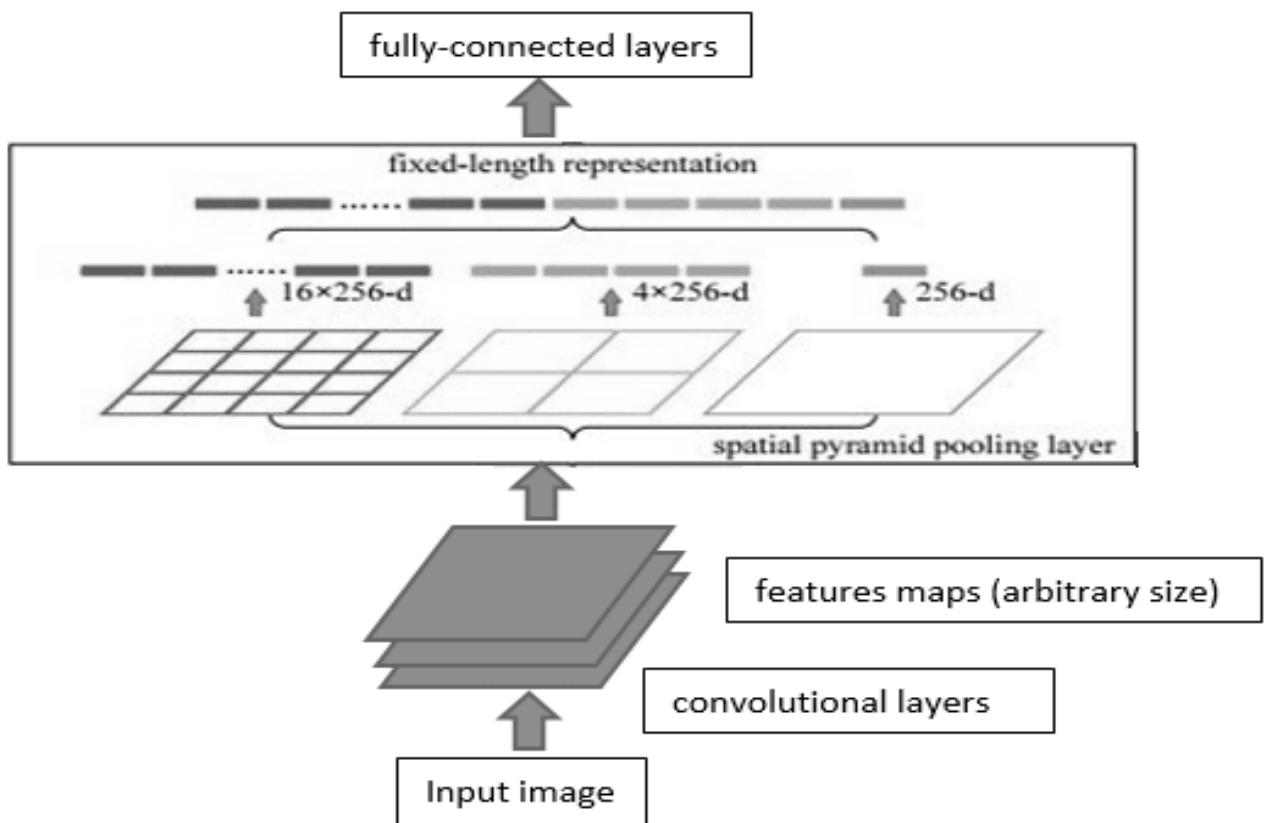


Figure 2.7: Spatial pyramid pooling (SPP-net) [24]

2.2.8 YOLO

YOLO is an object detection algorithm which works coco.names data sets. It is a simple regulation problem which takes an input image learns the class probability and then the bounding box coordinate [25]. It is a single shoot detector which detects the image in a single scan. It is also faster but struggles with small object as it is a single shoot detector or sometimes it misses the small objects which are there. It is better because it is fast and does the detection in a single scan.

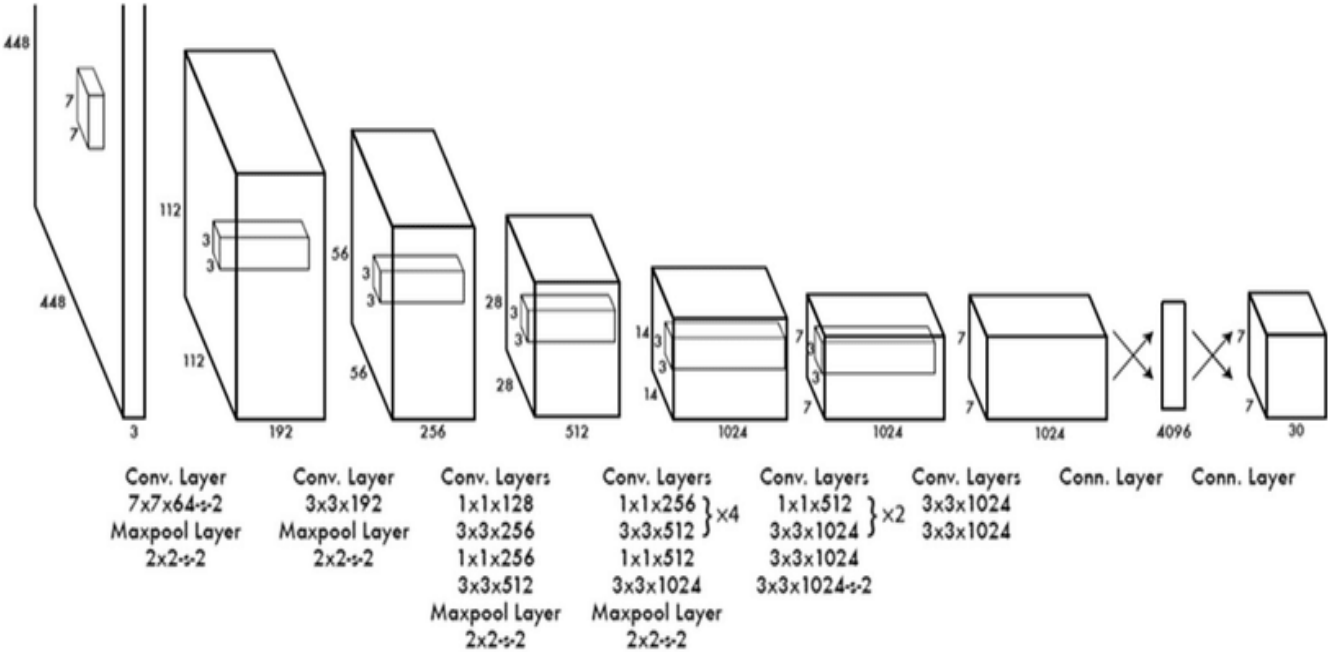


Figure 2.8: YOLO architecture [26]

CHAPTER THREE

Methodology

3.1 Beginning

In train detection system is main issue of managing the quality of image frames. The image qualities are a major part of the object detection system. The resolution of the image has increased the frame number is also increased. In number of frames increases its need extra time is for executing. In this case the process is sometimes slow [22].

3.2 Analysis of the proposed system

This project concentrates on the frame issues for maintaining the image frame. An SSD with OpenCV technique has been introduced in the proposed to minimize the frame and time consumption for detection an object. The object is detected in sequence with each other of the frames and move to the next frame. If each frame is executed in every short span of time human eye can see the result until the frame is not ended.

3.3 Block diagram of the system

The basic units of the proposed system can be divided into seven parts.

1. Real time image converted
2. Extract features from a real time image
3. Reference image model (Object)
4. Extract features from image
5. Match Features of both
6. Calculate the object matching value
7. Recognize the object and display if the threshold value is less than

Here the real time image is collected from a camera or other sources and sends it to the next stage. Then features are extracted from the real time image and the data is collected which is used in future for matching the reference model data. Reference image model (Object) is pre-trained images with labeling. The features of the data set images are extracted and sent to the

matching table. The matching is based on which parts are match and how much percentage is matched. The next stage is execution. Then object is recognized if the stored matching value is greater than the threshold value. Then it sends to the labeling stage. The name of vehicle is labeled in the displayed frame.

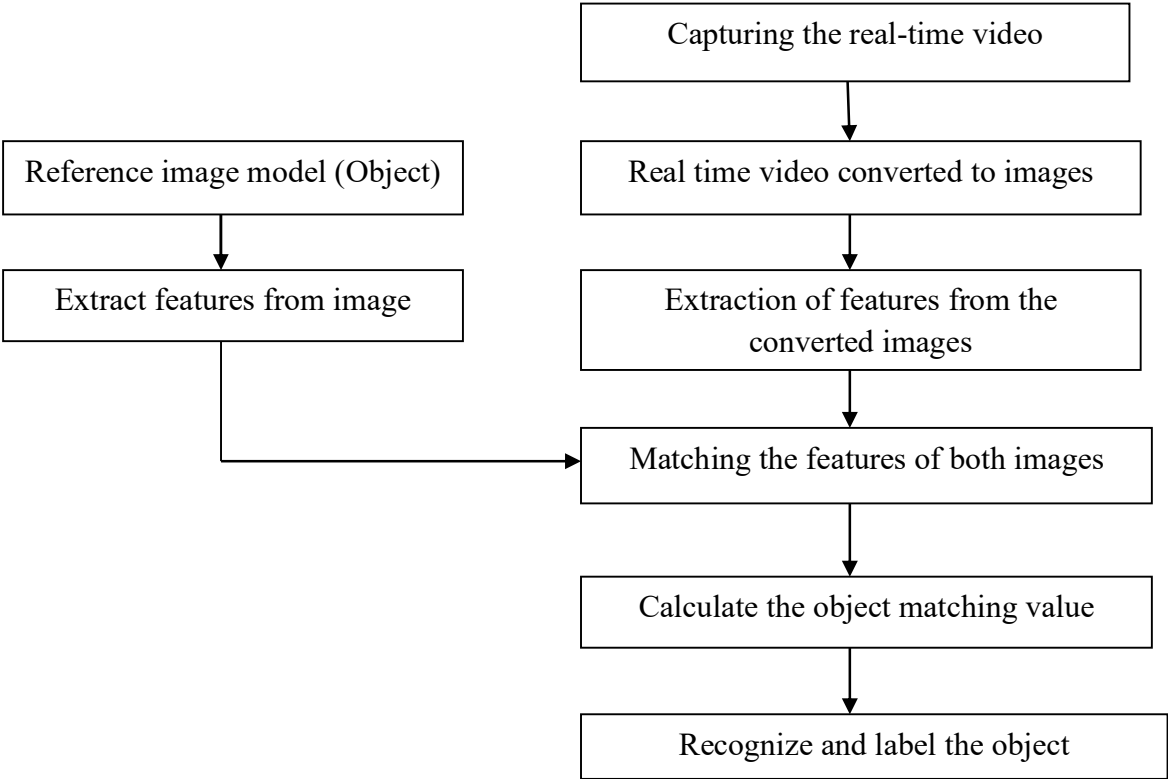


Figure 3.1: Block diagram of the proposed system

3.4 Platform for the system

Python is installed in our computing system. Its version 3.9.5 is updated from the official python website via command prompt. We have also downloaded python packages, cuda, numpy, OpenCV etc. We have downloaded object detection model file from github. We used sublime text as the code editor.

Numpy

NumPy is library of the Python programming language, adding support for large, multi-dimensional array and matrix, along with a large collection of high-level mathematical function

to operate over these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several developers. In 2005 Travis Olphant created NumPy by incorporating features of computing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. NumPy is open-source software and has many contributors.

OpenCV

OpenCV is a library of programming functions mainly aimed at real time computer vision. Originally developed by Intel, it is later supported by *Willow Garage* than issues. The library is a cross-platform and free to use under the open-source BSD license.

ImageAI

The imageAI provides an API to recognize 1000 different objects in a picture using pre-trained models that was trained on the ImageNet-1000 dataset. The model implementations provided are SqueezeNet, ResNet, InceptionV3 and DenseNet.

Anaconda

Anaconda environment is installed for this detection system. At times, it made the system slower, so we turned it off.

Jedi

Jedi for auto compilation of python code is installed.

3.5 Outline of methodology/ experimental design

You only look once- YOLO (OpenCV) is a unique being a state-of-the-art, real-time object detection system that offers higher detection accuracy at a faster speed. Further, by reducing the frame rate, higher detection speed can be achieved without compromising the accuracy. The proposed method can have significant impact in the automation of railway level crossings [27]. Considering this detection system first of all we have classified the image. In this case, here are also training images of the system. Secondly, we have focused object classification and localization. In this case the image has a background part, but our system only detects our

specified object that is called localization. Third is multiple objects detection and localization. In this case the image there are many objects at a time in one image. They are separately detected and localized separately. The traditional object detection model, there are three stages one is informative region selection another is feature extraction and one another is classification. Region proposal based framework is composed of three correlated stages, including region proposal generation, feature extraction with CNN, classification and bounding box regression, which are usually trained separately. Regression or classification based framework is a one-step framework based on global regression or classification, mapping straightly from image pixels to bounding box coordinates and class probabilities, can reduce expenses. Two significant frameworks one is YOLO, and another is single shot multi box detector (SSD) [28]. We have used SSD by our system to detect objects. The objectives have achieved by the following methodologies-

- i. The model has trained using coco.names dataset.
- ii. The classification of images is using vector matrix of array and list.
- iii. Accuracy, precision and threshold value have calculated for different objects.
- iv. The frame rate has been optimized for achieving higher detection speed.
- v. Live video feed has been used to test the trained model.
- vi. The model has trained to detect objects from foggy and rainy images.

CHAPTER FOUR

Results and Discussions

4.1 Introduction

This chapter describes the result and discussions of our project.

4.2 Result and discussions

All of the objects are passed through the matching condition. If the condition is true, then next stage processes and shows our desired output. Otherwise, it moves to the next frame. Several times the system could not detect the presence of train due to the inferior quality of the image. In those cases, the object matching value is less than the threshold value.

4.3 Output

In Kamalapur Railway Station, Dhaka, we ran our system and detected train in real time through webcam footages. Some screenshots of those instances are given below.

4.3.1 Train detection

The program was launched from the platform of Kamalapur Railway Station, when the train arrived at the platform, train detection took place. As shown in Figure 4.1 (a), the train is detected as soon as it comes in front of the camera, even if the distance is long. As shown in Figure 4.1 (b), two or more trains in the same image can be detected simultaneously. As shown in Figure 4.1 (c), if more than two trains are moving within the same frame, the system can detect the trains well.

The system was then tested in different environmental conditions. Figure 5 shows that even in dusty weather the system can detect the presence of the train. Figure 6 shows the performance of the detection system in foggy weather. Figure 7 shows that our project can actually detect the train in rainy weather as well. So, we can say that whatever the weather condition is, our project is successful in detecting objects in that weather.



Figure 4.1 (a): Long distance real time train detection via webcam



Figure 4.1 (b): Real time detection of two trains in the same frame. The frame is captured using webcam.



Figure 4.1 (c): Real time detection of overlapping trains



Figure 5: Train detection in dusty environment.



Figure 6: Train detection in foggy weather.



Figure 7: Train detection in rainy weather.

4.3.2 Pedestrian detection

After we launched our project, if a pedestrian comes in front of the camera, then our project can detect pedestrians. It can detect multiple persons in the same image. Figure 8 shows the pedestrian detection performance of the system. So, we can say that our project can detect a train or a pedestrian passing through in front of the camera.



Figure 8: Pedestrian detection

4.3.3 Bus detection

The program is able to detect a number of vehicles including passenger bus. While the program is running, if a bus comes in front of the camera, then the program can detect bus. It can detect multiple buses in the same image. Figure 9 shows an instance of bus detection by the program.



Figure 9: Bus detection

4.3.4 Truck detection

Then we tested the system to detect truck in roads. After launch of the program, if a bus comes in front of the camera, then the system can detect presence of the truck in the frame. It can detect multiple trucks in the same image. Figure 10 shows the performance of the system in truck detection.



Figure 10: Truck detection

4.3.5 Car detection

The program is equipped to detect cars as well. It can detect multiple cars in the same image. Figure 11 shows the car detection performance in a dynamic environment. So, we can conclude that our project can detect multiple objects whether it is a train or other vehicles like bus, truck, car or even a pedestrian coming in front of the camera.



Figure 11: Car detection

4.4 Real time train detection

In the above images, we have added screenshots of real time train, vehicles and pedestrian detection using laptop webcam and running the detection program. In real time object detection, we first labeled images, then set up our training repository of train and other vehicles and then executed the detection program. The labeled data are saved in a directory or folder. When the label is matched, then it is given an object label and shown as output.

CHAPTER FIVE

Conclusion

5.1 Conclusion

Real time train, vehicle and pedestrian detection system for automated railway level crossing system is a modern technique. This detection system is a software system that is running an infinity loop for webcam and terminated point for fixed resources. The Single shot detector is faster than previous advanced techniques (YOLO) and is significantly more accurate. SSD predicts category scores, and box shifts for a fixed number of default boundary boxes using convolution filters applied to feature maps. To obtain a high precision, we have produced predictions of different scales from characteristic maps of different dimensions, and then separate the predictions in relation to appearance. These functions lead to high precision, even for low resolution input images. Other algorithms normally use object proposal methodology where they would come up with a way to break down the image segmented into parts to suggest where they could potentially be objects. Algorithms like this sacrifice precision. Therefore, researchers came up with an interesting solution where they do everything in one single shot. The system just looks at the picture once; it does not need to return to the picture again, it doesn't need to run a lot of convolutional neural networks.

5.2 Suggestions for future works

Computers have become the most important technology of our last lives and can solve increasingly difficult problems. Specifically, computers are generally better than humans in repetitive tasks, which require a great deal of data and computation. In the past ten years, computers have become so powerful that they can be used for complex tasks. This project elucidates one of those highly computerized applications that became possible in the recent year, which is the detection of objects. The complex task of finding objects within a specific image or video frame.

The future of object detection technology is proving itself, and much like the original Industrial Revolution, it has the potential, at the very least, to free people from tedious work that have done more efficiently and effectively by machines. Thus, these challenges circumvent the need for a

lot of training requiring a massive number of datasets to serve more nuanced tasks, with its continued evolution, with the features and techniques that make it possible, it might soon become the next great thing in the future.

In future we will work on this project in hardware parts and implementing of the project.

References

- [1] A Burgos, A Goni, A Illarramendi , and J Bermudez , "Real-Time Detection of Apneas on a PDA," IEEE Transactions on Information Technology in Biomedicine, pp. 995-1002, 2009.
- [2] Biswas D, Su H, Wang C, Stevanovic A, and Wang W, "An automatic traffic density estimation using Single Shot Detection (SSD) and MobileNet-SSD," Physics and Chemistry of the Earth, pp. 176-184, Apr 2019.
- [3] Banuchandar J, Kaliraj V, Balasubramanian P, Deepa S, and Thamilarasi N, "Automated unmanned railway level crossing system.," International Journal of Modern Engineering Research (IJMER), vol. 2(1), pp. 458-463, Jan 2012.
- [4] Pawełczyk M and Wojtyra M, "Real world object detection dataset for quadcopter unmanned aerial vehicle detection," IEEE Access, vol. 8, pp. 174394-409, Sep 2020.
- [5] Xu T, Sun X, and Diao W, "ASSD: Feature aligned single-shot detection for multiscale objects in aerial imagery," IEEE Transactions on Geoscience and Remote Sensing, Jun 2021.
- [6] "Making level crossings safer," The Financial Express, January 2021.
- [7] Zhao ZQ, Zheng P, and Xu ST, "Object detection with deep learning: A review," IEEE transactions on neural networks and learning systems, vol. 30(11), pp. 3212-3232, 2019.
- [8] Ho RK, Wang Z, and Tang SS, "Real-time object detection technology in railway operations," 2021.
- [9] Ji Y, Zhang H, and Zhang Z, "CNN-based encoder-decoder networks for salient object detection: A comprehensive review and recent advances," Information Sciences, vol. 546, pp. 835-857, Feb 2021.
- [10] Wen L, Ding J, and Loffeld O, "Video SAR moving target detection using dual faster R-CNN," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 14, pp. 2984-2994, Feb 2021.
- [11] Shilpa Ananth, "Fast R-CNN for object detection," Towards Data Science, Aug 2019.
- [12] Mansour RF, Escorcía-Gutiérrez J, and Gamarra M, "Intelligent video anomaly detection and classification using faster RCNN with deep reinforcement learning model," Image and Vision Computing, vol. 112, p. 104229, Aug 2021.

- [13] Deng , Zhipeng & Sun , and Hao & Zhou , "Multi-scale object detection in remote sensing imagery with convolutional neural networks," ISPRS Journal of Photogrammetry and Remote Sensing, May 2018.
- [14] Khan MN, Das A, and Ahmed MN, "Multilevel weather detection based on images: A machine learning approach with histogram of oriented gradient and local binary pattern-based features," Journal of Intelligent Transportation Systems, vol. 25(5), pp. 513-532, Sep 2021.
- [15] Lu , Heng & Ma , and Lei & Fu , "Landslides Information Extraction Using Object-Oriented Image Analysis Paradigm Based on Deep Learning and Transfer Learning," Remote Sensing, vol. 12, p. 752, Feb 2020.
- [16] Yee LR, Kamaludin H, and Safar NZ, "Intelligence Eye for Blinds and Visually Impaired by Using Region-Based Convolutional Neural Network (R-CNN).," JOIV: International Journal on Informatics Visualization, vol. 5(4), pp. 409-414, Dec 2021.
- [17] Horak , Karel & Sablatnig , and Robert , "Deep learning concepts and datasets for image recognition: overview 2019," p. 100, AUG 2019.
- [18] Huang H, Huang Y, and Mu X, "Research on Recognition and Location Method of Insulator in Infrared Image Based on Deep Learning," Journal of Physics: Conference Series, vol. 2087, p. 012090, Nov 2021.
- [19] Nguyen-Meidine , Le Thanh & Granger , and Eric & Kiran , "A Comparison of CNN-based Face and Head Detectors for Real-Time Video Surveillance Applications," Nov 2017.
- [20] Jee G, Gm H, and Gourisaria MK, "Efficacy Determination of Various Base Networks in Single Shot Detector for Automatic Mask Localisation in a Post COVID Setup," Journal of Experimental & Theoretical Artificial Intelligence, pp. 1-20, Oct 2021.
- [21] Jonathan Hui , "SSD object detection: Single Shot MultiBox Detector for real-time processing," Mar 2018.
- [22] Chen DJ, Hsieh HY, and Liu TL, "Adaptive image transformer for one-shot object detection," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12247-12256, 2021.
- [23] Kumar S and Gaur D, "Thermal Object Detection Using Yolov3 and Spatial Pyramid Pooling," Proceedings of International Conference on Machine Intelligence and Data Science Applications, pp. 553-565, 2021.

- [24] Dewi and Christine , "Evaluation of Robust Spatial Pyramid Pooling Based on Convolutional Neural Network for Traffic Sign Recognition System," Electronics, May 2020.
- [25] Warnicke A and Jönsson J, "Implementation of a Vision System for an Autonomous Railway Maintenance Vehicle: Track and Object Detection with YOLO, Neural Networks and Region Growing.," 2021.
- [26] Wu and Jiayu, "Complexity and accuracy analysis of common artificial neural networks on pedestrian detection," MATEC Web of Conferences, Jan 2018.
- [27] Fayyaz, M.A.B, and C., Johnson, "Object detection at level crossing using deep," Micromachines, vol. 11(12), p. 1055, 2020.
- [28] "Track vacancy detection systems," Siemens.
- [29] S Yadav, "Deep learning based safe social distancing and face mask detection in public," International Journal for Research in Applied, vol. 8(7), pp. 1368-1375, 2020.

Appendix

```
#Variable set for train

    if (names[c])=='train':
        print("\n\n")
        print("Train found")
        break
    else:
        print("\n\n Train not found")

# Image load

img = cv2.imread("C:/Users/joy_c/Dropbox/Personal/BUET/Project/Jibon.jpg" , 1)

# 0 for black, 1 for color image

print("image in Pixel \n", img);

cv2.imshow("Output Image", img)

cv2.waitKey(2000)

print("First image end ")

#url = "http://192.168.0.113:8888/shot.jpg"

img2 = cv2.imread("Copy.jpg", img)

cv2.destroyWindow("img");

img_copy = cv2.imread("C:/Users/joy_c/Dropbox/Personal/BUET/Project/copy.jpg", 0 )

cv2.imshow("Output Image Live", img_copy)

cv2.waitKey(5000)

print("Second image Copy end")

#Webcame video

cap1 = cv2.VideoCapture(0)
```

```

# default webcam 0, set webcam 1

cap1.set(3, 640)      # 3 for width of image upper for openCV
cap1.set(4, 480)      # 4 for height of image up to down
cap1.set(10, 100)     # 10 for Brightness of image

while True:

    success, imgv1 = cap1.read()

    cv2.imshow("Webcame ", imgv1)

    if cv2.waitKey(100) & 0xFF == ord('q'):

        break

print("End webcam")

# Video load

cap = cv2.VideoCapture("C:/Users/joy_c/Desktop/road.mp4")

while True:

    success, imgv = cap.read()

    cv2.imshow("Video", imgv)

    if cv2.waitKey(100) & 0xFF == ord('q'):

        break

print("End video")

print("End of the program");

cv2.destroyAllWindows()

# Face detection

import cv2

# XML file load haarcascade_frontalface_default.xml

faceCascade=
cv2.CascadeClassifier("C:\\Users\\joy_c\\AppData\\Roaming\\Python\\Python39\\site-
packages\\cv2\\data\\haarcascade_frontalface_default.xml")

```

```

# Image load

img = cv2.imread('C:/Users/joy_c/Dropbox/Personal/BUET/Project/Copy.png')

# the input image is converted to gray scale

imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

faces = faceCascade.detectMultiScale(imgGray,1.1,4)

for (x,y,w,h) in faces:

    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

cv2.imshow("Face of mouth", img)

cv2.waitKey(0)

# Train detection

import cv2

import numpy as np

# yolov3.weights, yolov3.cfg, coco.names, detection.py and other files three are same folders

load = cv2.dnn.readNet('yolov3.weights', 'yolov3.cfg')

classes = []

with open('coco.names', 'r') as f:

    classes = f.read().splitlines()

#img = cv2.imread('zidane.jpg')

cap = cv2.VideoCapture(0)

while True:

    _, img = cap.read()

    height, width, _ = img.shape

    # image scaling

    blob = cv2.dnn.blobFromImage(img, 1/255, (416, 416), (0, 0,0), swapRB = True, crop =
False)

```



```

'''
# How many frames
for b in blob:
    for n, img_blob in enumerate(b):
        cv2.imshow(str(n), img_blob)
'''

load.setInput(blob)
output_layers_names = load.getUnconnectedOutLayersNames()
layerOutPuts = load.forward(output_layers_names)

boxes = []
confidences = []
class_ids = []

for output in (layerOutPuts):
    for detection in output:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]

        if confidence > 0.5:
            center_x = int(detection[0]*width)
            center_y = int(detection[1]*height)
            w = int(detection[2]*width)
            h = int(detection[3]*height)
            x = int(center_x - w/2)
            y = int(center_y - h/2)
            boxes.append([x, y, w, h])

```

```

        confidences.append((float(confidenc)))

        class_ids.append(class_id)

#print(len(boxes))

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

#print(indexes.flatten())

font = cv2.FONT_HERSHEY_PLAIN

colors = np.random.uniform(0, 255, size = (len(boxes), 3))

if len(indexes) > 0:

    for i in (indexes.flatten()):

        x,y,w,h = boxes[i]

        label = str(classes[class_ids[i]])

        if label=='train':

            confidenc = str(round(confidences[i], 2))

            color = colors[i]

            cv2.rectangle(img, (x, y), (x+w, y+h), color, 2)

            cv2.putText(img, label + " " + confidenc, (x, y+20), font, 2, (255,
255, 255), 2)

            cv2.imshow('Output_File To exit press q 2 times', img)

            key = cv2.waitKey(1)

            if cv2.waitKey(1) & 0xFF == ord('q'):

                break

cap.release()

'''

if label == 'train':

    print('Train detection successfull')

```

```

else:
    print('Train detection unsuccessfull')
'''
cv2.destroyAllWindows()
#Sound set for train
if (label)=='train':
    print("\n\n")
    x=37
    y=2500
    for i in range(2):
        win32api.Beep(x,y)
        #if cv2.waitKey(1) & 0xFF == ord('q'):
            # break
        x+=100
        y-=120
    ## break
    #time.sleep(2) # Delay 2 seconds for next time detection
#else:
    #break;
    #print("\n Train not found")
# show the reference image
##cv2.imshow("ref_image", ref_image)
# initialize the camera object so that we
# can get frame from it
cap = cv2.VideoCapture(0)

```

```

# looping through frame, incoming from
# camera/video
while True:
    # reading the frame from camera
    _, frame = cap.read()
    # calling face_data function to find
    # the width of face(pixels) in the frame
    face_width_in_frame = face_data(frame)
    # check if the face is zero then not
    # find the distance
    if face_width_in_frame != 0:
        # finding the distance by calling function
        # Distance finder function need
        # these arguments the Focal_Length,
        # Known_width(centimeters),
        # and Known_distance(centimeters)
        Distance = Distance_finder(
            Focal_length_found, Known_width, face_width_in_frame)
        # draw line as background of text
        cv2.line(frame, (30, 30), (230, 30), RED, 32)
        cv2.line(frame, (30, 30), (230, 30), BLACK, 28)
        # Drawing Text on the screen
        cv2.putText(
            frame, f"Distance: {round(Distance,2)} CM", (30, 35),
            fonts, 0.6, GREEN, 2)

```

```
# show the frame on the screen
cv2.imshow("frame", frame)
# quit the program if you press 'q' on keyboard
if cv2.waitKey(1) == ord("q"):
    break
# closing the camera
cap.release()
# closing the windows that are opened
cv2.destroyAllWindows()
```

```

#/*Project Coding*/

import cv2

thres = 0.65 # Threshold to detect object

cap = cv2.VideoCapture("recording_video/IMG_0979.mp4")
cap.set(3,1280)
cap.set(4,720)
cap.set(10,70)

classNames= []
class_ids = []
classFile = 'coco.names'
with open(classFile,'rt') as f:
    classNames = f.read().rstrip('\n').split('\n')

configPath = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
weightsPath = 'frozen_inference_graph.pb'

net = cv2.dnn_DetectionModel(weightsPath,configPath)
net.setInputSize(320,320)
net.setInputScale(1.0/ 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)

#boxes = []
#colors = np.random.uniform(0, 255, size = (len(boxes), 3))
while True:
    success,img = cap.read()
    classIds, confs, bbox = net.detect(img,confThreshold=thres)
    #print(classIds,bbox) # Array of objects

```

```

if len(classIds) != 0:
    for classId, confidence, box in zip(classIds.flatten(), confs.flatten(), bbox):
        #cv2.rectangle(img, (x, y), (x+w, y+h), color, 2)
        #cv2.rectangle(img, box, color=(0,255,0), thickness=2)
    label = str(classNames[classId-1])
    #print(label)
    if (label=='train' or label=='bicycle' or label=='car' or label=='bus' or label=='truck'):
        cv2.rectangle(img, box, color=(0,255,0), thickness=2)
        cv2.putText(img, classNames[classId-1].upper(), (box[0]+10, box[1]+30),
                    cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)
        cv2.putText(img, str(round(confidence*100,2)), (box[0]+200, box[1]+30),
                    cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)
        print(label, ' Detect')
        #print("Length = ", len(label))
    cv2.imshow("Output File press q to exit", img)
    key=cv2.waitKey(1)
    if key == ord('q') or key == 27:
        break
    cap.release()
'''
if label == 'train':
    print('Train detection successfull')

else:
    print('Train detection unsuccessful')
'''
cv2.destroyAllWindows()

```