Int. J. Nonlinear Anal. Appl. 14 (2023) 1, 2871–2880
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2023.7353

A monitoring system for railways based on WSN

Zena Abd Alrahman[a,b,*], Ali Adham[a,c]

[a] *The Informatics Institute for Graduate Studies, Baghdad, Iraq*

[b] *Medical Instrumentation Engineering Department, AL-Esraa University College, Baghdad, Iraq*

[c] *Al-Nahrain Center for Strategic Studies, Baghdad, Iraq*

(Communicated by Madjid Eshaghi Gordji)

Abstract

The process of controlling railway systems is one of the important and effective topics in the process of maintaining the flow of trains' movement and organizing the travel process, as well as providing early readings of any defect or problem that occurs in the railway network to avoid, treat accidents and ensure a safe environment for the movement of train cars across the geographical area. Therefore, a continuous follow-up of the railway condition must be provided to ensure that services continue to be provided. Intelligent railway maintenance improves safety and efficiency. This work presents the design and implementation of a real-time monitoring system for railways based on WSN. This study proposes a system consisting of a base station server and a Rail controller. The Base Station (BS) can automatically monitor and control most railway paths. For that, a classification-based deep learning model for object detection near the railway and making appropriate decisions were proposed. To improve classification-model performance, the yolov3 algorithm for object detection was proposed. On the Rail controller side, the Raspberry Pi 4 was utilized as a low-cost processing unit that can be used as a control unit to control some processes, such as streaming video from a camera, gathering information from railway sensors, and sending data to the central station server (PC) by using WiFi protocol. The model can detect and control railway issues in real-time by receiving streaming data and directly detecting, classifying issues, and making the best decisions. By alarming or controlling the desired train and stopping it.

Keywords: Railway Inspection, Real-Time Monitoring System, Deep Learning, object detection, YOLOV3, Raspberry pi 4
2020 MSC: 68T07

1 Introduction

Checking the structure and geometry of railway rails can help prevent accidents. In recent decades, the railroad sector spent 40% of its income on infrastructure maintenance, renovations, and expansion [1, 2]. With more people utilizing the train, it's more complicated, overworked, and likely to break down. Environmental variables and mechanical forces make railway tracks break faster [3]. Because of this, railways started to deploy AI technologies (AI). Fleet reliability is one-way innovative rail operations can boost efficiency and reduce costs. Predictive maintenance reduces failures, unscheduled maintenance, and rail operators' reserve asset capacity [3]. Wheel failures, track deterioration,

*Corresponding author
Email addresses: zena.abd@esraa.edu.iq (Zena Abd Alrahman), aliadham12@gmail.com (Ali Adham)

Received: November 2022 *Accepted:* January 2023

and the presence of vehicles or animals near the tracks are common threats that can derail a train. Derailments cause 50% of train accidents and cost billions [1, 4]. Now more than ever, railways are keeping a careful eye on performance to discover and fix problems before they significantly affect them [5]. Many techniques for detecting rail problems exist. The Instrumented cars with video cameras, accelerometers on the car body, acoustic sensors, ultrasonic wave sensors, GPS, linear potentiometers, and strain gauges on the top chord. Besides an instrumented car and sensors, a rail defect detection system includes a data-gathering system, a data analytics tool, and knowledge extraction techniques are all components of a rail fault detection system. Inspectors would walk the track or drive slowly in a high-rail vehicle in the past to detect issues. This is costly, time-consuming, reduces schedule performance (because of tracking possession), and puts workers in danger. Recent machine-vision advancements have adopted and improved inspections.

2 Object Detection and Computer Vision

Object detection is a computer technique that makes use of machine learning and deep learning. The primary goals of computer vision and image processing are recognizing instances of items of a certain class in images and videos. Face and pedestrian detection are two areas of object detection that have gotten a lot of attention. People tracking, people counting, automated CCTV surveillance, person detection, and vehicle detection may all be performed with object detection [6]. Before implementing a machine learning-based object detection method, features must be manually extracted from photos. by employing image-based feature extraction techniques such as the Histogram of Oriented Gradients (HOG), speed-up robust features (SURF), and Local Binary Patterns (LBP), etc [7]. Based on deep learning The object identification technique may extract the feature automatically using a deep learning algorithm such as a convolutional neural network, an autoencoder, Variance Auto-Encoder, and so on, which generates the feature from the image such as an edge, a form, and so on [7]. Begin with deep learning object detection by either designing and training a custom object detector or using a pre-trained object detector. There is various object detection tasks, as shown in Fig. 1.

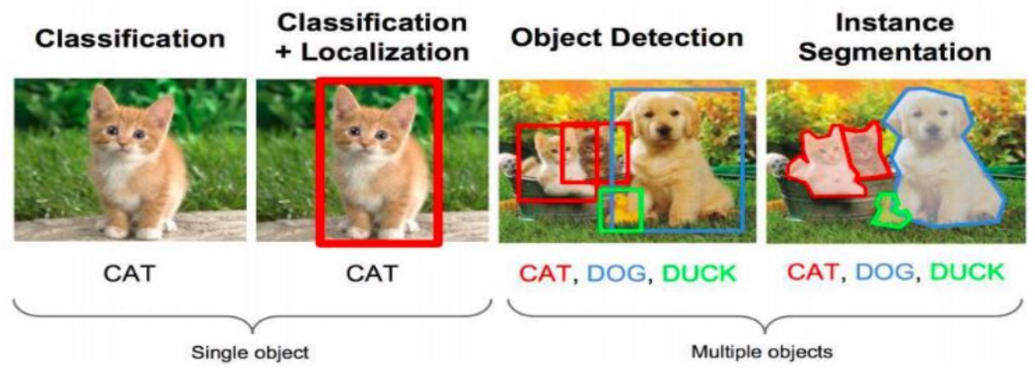

Figure 1: Object Detection Tasks

3 YOLOV3 Algorithm

YOLOv3 is a real-time object detection technique for videos, live feeds, or images. Its network consists of Dark-Net53, FPN, and a detection network (YOLO layer). It is Fast and accurate in mAP and IOU values. The YOLO technique uses deep convolutional neural network features to recognize objects in real-time. This includes the class label, bounding box coordinates, and sizes. CNNs process input images as structured data arrays and recognize patterns [4]. The YOLOv3 algorithm provides a high accuracy rate, accurate positioning, and rapid speed. When multi-scale prediction algorithms are used, they can detect small targets and are very robust [5]. YOLOv3's convolutional layers predict detection after transferring learned features to a classifier or regressor. Residual networks improve feature extraction networks. The backbone network is upgraded from Darknet-19 to Darknet-53 to extract more features [5]. YOLO is built with Keras or OpenCV.

3.1 Anchor Boxes

YOLOv3 object detectors predict log-space transforms that offset "default" bounding boxes. Bounding boxes are called anchors. Transforms are later applied to anchor boxes to predict [4]. Three anchors are in YOLOv3. Each boundary box represents one object and has a confidence score indicating how accurate the prediction should be. In order to build the boundary box, the dimensions of the ground truth boxes from the original dataset are grouped [4]. As for each border-box, the network offsets 4 element values (bx, by, bh, bw) and predicts 4 coordinates (tx, ty, tw, th), as illustrated in Figure 2. If the cell is offset from the top left corner of the image by (cx, cy), the bounding box prior has a width PW, ph and the predictions correspond to equations as shown in Fig. 2:

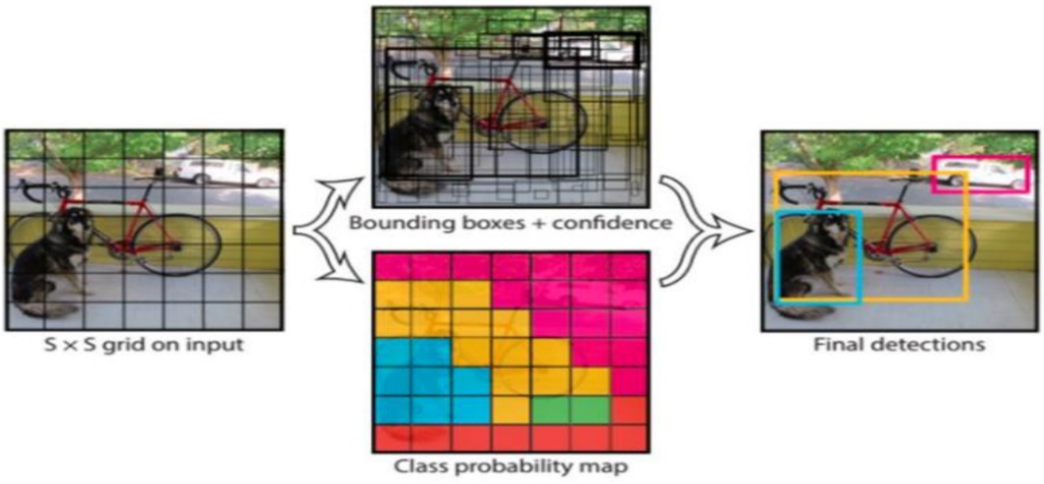

Figure 2: Anchor Box with Dimension Priors and Location Prediction [8]

3.2 Non-Maximum Suppression

Objects can be recognized multiple times if more than one bounding box detects them. Figure 3 shows how the non-maximum suppression (NMS) method removes redundant bounding boxes during testing [9], It avoids this issue and just passes new detections. NMS uses confidence thresholds to prevent double detections. It's an essential part of using YOLOv3 [4]. YOLO chooses the decision with the highest likelihood score. Then, it suppresses the bounding boxes with the highest Intersection over Union with the current high probability box. Repeat until final bounding boxes are obtained [10].

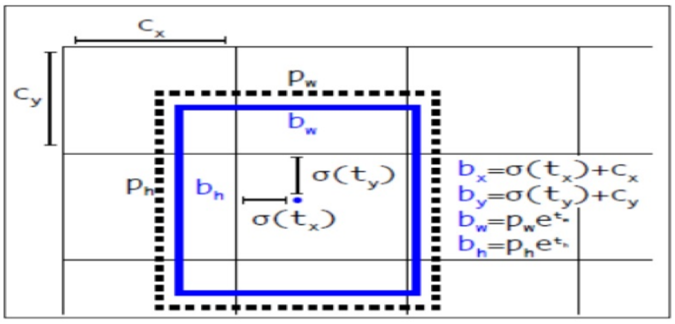

Figure 3: Non-Maximum Suppression

3.3 Class Confidence and Box Confidence Scores

Confidence means an object exists in a box, and class score represents the conditional probability (i.e., the possibility of class x given an object exists in this box). Total class confidence is the product of abjectness and class score [11]. Each box contains an x, y, w, h, and confidence score. The confidence score indicates how likely and precise a box is to contain a class. The bounding box's width and height are set to the image's size. x and y are cell offsets, and all four bounding box values are 0-1. Each cell in YOLOv3 has 20 conditional class probabilities. Class confidence score equals the product of the box confidence score and conditional class probability for each final boundary box used as a positive prediction. Conditional class probability is the likelihood that an identifiable object belongs to a specified class. YOLOv3 predicts h, w, and depth. Finally, boundary boxes with high confidence scores (above 0.25) are preserved as final predictions [4].

4 Project Implementation

4.1 Software System

Python is the most common data science programming language, according to DASCA. Data scientists primarily utilize Python. Python is common among data scientists because it makes machine learning programming easy. we can also create new libraries and tools. It offers ready-to-use machine learning libraries, add-on models, and frameworks to facilitate development. Our project will be developed using Python. YOLOv3 requires OpenCV2 [3]. Our work uses TensorFlow, Numpy, OS, PyTorch, and other frameworks are used in our work.

4.2 Dataset

The touch vision package in the Python Torch library provides access to a wide variety of common data sets. In order to get the PASCAL VOC dataset, we use the touch vision module from the PyTorch library. There are 9963 images and 24,640 objects annotated in the 2007 dataset [12], while 11,530 images and 27,450 annotations are available in the 2012 release [12]. the training utilizes the VOC2012 dataset, while the "test" utilizes the VOC2007 dataset. The image sizes and bounding box shapes to the YOLO standard have been adapted. Since YOLO expects its input images to be 608 × 608 pixels, we upsize them from 320. Images from the PASCAL VOC dataset must be modified before they can be used by the neural network. PyTorch's reshape function provides the required functionality for this to be possible.

For YOLO, Darknet53 was used, a convolutional neural network implemented in C and CUDA. Darknet handles all training and inference. Before training, darknet53 weights are predefined.

4.3 Train Model

The object detection algorithm is based, on the Yolov3 model. YOLOv3 is a widely used and speedy object detection system, but it is less precise than RetinaNet and Faster RCNN. However, it has a significant advantage in terms of detection speed [13]. In our research, and based on related works, the aim is to enhance YOLOv3 using different approaches. Improvements in training methods have led to significant gains in performance for image classification networks. In order to improve the yolov3, several optimizations were made including preprocessing side and training side. Firstly, they proposed several training heuristics for object detection networks. Also, they utilize an Adaptive Spatial Fusion (ASF) for Feature Pyramids. The steps optimization strategy of the proposed model is as follows:

4.3.1 Image Augmentation Using Mixup and Label Smoothing

During the preprocessing, the data was augmented. Because most datasets don't have balanced samples, the model can't predict tiny samples. Image Mixup and label smoothing increase the model's generalization capacity, encouraging linear behavior between training examples. Two examples were selected randomly (xi, yi) and (xj, yj) [14], and create a new instance using linear interpolation, as shown in the equation below. It can reduce false and missed detection rates in complex backgrounds. The Mixup approach stabilizes model training by reducing fluctuations in model predictions and gradient norms between training samples [15].

$$\hat{x} = \lambda xi + (1 - \lambda)xj \tag{4.1}$$

$$\hat{y} = \lambda yi + (1 - \lambda)yj \tag{4.2}$$

$(xi; yi)$ and $(xj; yj)$ are two samples selected at random from our training data. $\lambda \in [0, 1]$ is a number picked at random from the Beta (α, α) distribution. We only use the new example $(\hat{x}, \hat{y})$ in mixup training.

Also, label smoothing is used to prevent the model's weak generalization ability, increase robustness, and improve classification problems. It's a form of output distribution regularization that prevents overfitting by softening training data ground-truth labels. Overfitting occurs when the true-value label isn't "soft" [16]. It works by introducing noise to the actual value of the class to limit the model and lower over-fitting.

4.3.2 Data Augmentation Methods

This method was performed during testing/ validation to improve predictions for cases when the object in the image is too small. The images are just resized by randomly choosing one of the popular interpolation techniques and then normalizing.

4.3.3 Adaptive Spatial Fusion Feature

It uses identical rescaling and adaptive fusing. High and low semantic value characteristics are concatenated or added element-wise [13]. Create a semantically and spatially robust feature. These features offer a better balance of speed and accuracy for detection. Combine these feature maps using only useful information from each scale. In summary, instead of making predictions on features at each level like in normal YOLOv3, features from the three levels are rescaled to the exact shape before fusing. Then, features at different levels are adaptively fused at each spatial location. On those new features, prediction/detection is made. Feature resizing employs interpolation to up-sample and pooling to down-sample [17].

4.3.4 ADAM Optimization Algorithm

The training process was performed to update network weights iterative by using the data-set PASCAL VOC 2012 and using The Adaptive Moment Estimation Optimizer (ADM optimizer). The ADAM algorithm for optimization is an expansion of stochastic descent. The ADAM is not the same as traditional stochastic gradient descent. It combines the benefits of two existing stochastic gradient descent extensions: root mean square propagation that maintains per-parameter learning rates.

5 Main Hardware Component

1. Raspberry Pi 4
2. Raspberry Pi Cameras (5 megapixels)
3. Ultrasonic Sensor HC-SR04
4. RF Transmitter/Receiver (2CH 27MHZ Remote Transmitter & Receiver)
5. Temperature Sensor (MAX6675 Module with K-Type Thermocouple Sensor)

6 Methodology

The methodology in the design part consists of three phases, the software setup phase, where all programs and libraries must be installed in raspberry pi or on a PC. Then, the hardware setup phase includes preparing all hardware devices and connecting them to the raspberry pi structuring. Lastly, configure the overall components of the system and put it inside the railway model that has been created in miniature to simulate reality. Finally, the main part of the proposed system is object detection algorithm-based deep learning. The detection algorithm can detect the object near the railway by gathering information from the camera and sensors and making the appropriate decision.

- The first step starts by connecting the Raspberry Pi with Raspberry Pi OS using Raspberry Pi Imager installed on a Linux laptop. The main program is written using Python on Pi
- Our proposed system works on two sides, the server in the central station (PC) and the rail controller side (raspberry pi).
- The rail Controller (Raspberry Pi) runs a program on the Railside to control (Connection to PiCamera and Sensors), then begins streaming video from the camera after connecting to the Base station (PC). Simple sensors were used to collect data, such as temperature sensors that detect fires near railways. Ultrasonic sensors determine if there are any objects near a railway.
- Base station side program run (Start GUI, start object detector, load weights) waiting order to connect. It can monitor and control all these aspects in an innovative, automated way.
- Connect the Base station to Rail Controller Via Wi-Fi using Secure Shell Protocol (SSH) protocol by adding the IP address of the Rail Controller.

- Rail Controller streaming video and sensors reading to Base station. The Object detector in the Base station starts processing each FPS and detects if there is any desired object in the camera view. In case there is an object detected, the system checks its distance D_{object}. Figure 4 shows the overall system work.
- The base station program checks temperature (T) too, by getting readings from the Rail Controller temperature sensor.

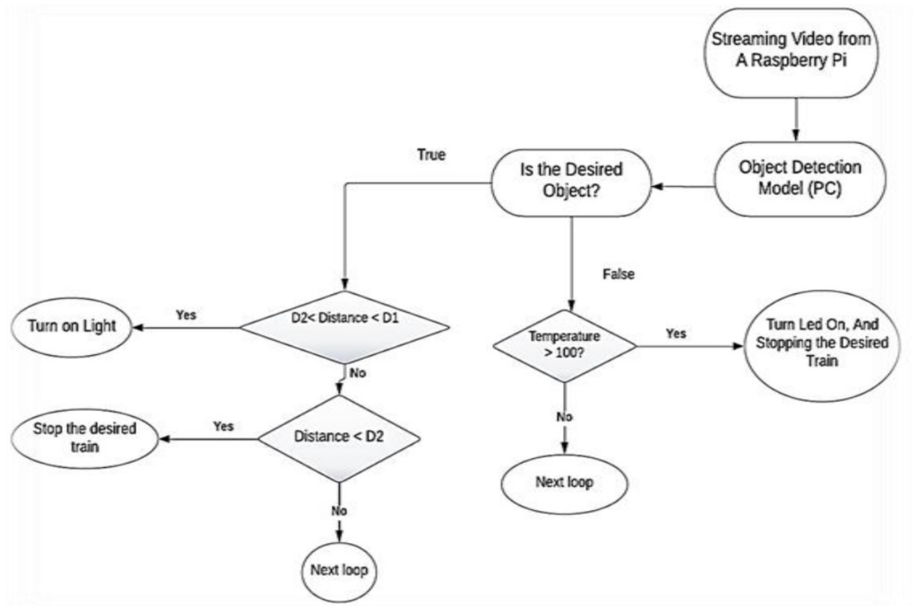

Figure 4: Overall System Flowchart

6.1 Controlling Condition

1. D1: The alarm distance is the distance that objects near the railway but not in a dangerous area.
2. D2: It is the critical distance that an object is very near to a railway, which is dangerous.
3. T: It is the temperature near the railway, which is dangerous.

7 Experimental Results and Analysis

- The experiments conducted in this article are on the platform 11th Gen Intel (R) Core (TM) i7-11800H CPU @ 2.30 GHz. We train the model firstly by predefined darknet53 weights are used to initialize the network before starting the training process, then we use PASCAL VOC 2012 for training, and VOC 2007 for testing with a total of 20 categories, and precisely evaluate the effectiveness of the proposed method on YOLOv3 detectors. In image preprocessing, Mixup images with label smoothing, data augmentation, and ASF are used for data enhancement.
- The initialization weights for YOLOv3-DL training use the weights of the pre-trained DarkNet weights. The process of object detection is enhanced by these improved pre-trained models, which demonstrate significant advantages in transfer learning.
- During the experiment, the learning rate is 10-3, the input picture is uniformly scaled to 320 to 608, the number of channels is 3, and Adam is used for optimization.
- The batch size is set to 8, the IoU threshold is 0.8, the NMS threshold is 0.5, and the score threshold is 0.8. the weight decay is set to 0.0005, and the epochs are set to 100.
- Finally, we evaluate the result of the yolov3 model by measuring the precision, recall, and mAP (mean average precision) for each object.
- The improved YOLO V3 network's detection results, which training on the PASCAL VOC dataset are shown in the figures below.

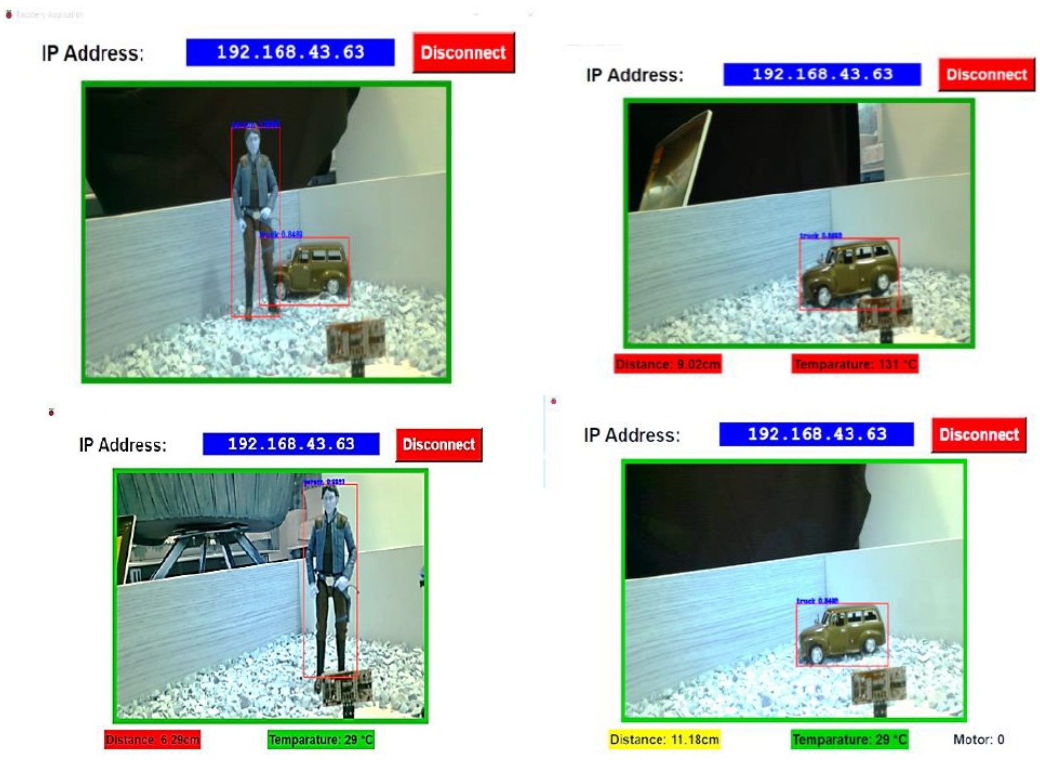

Figure 5: Object Detection Results

7.1 Detection Results

As shown in figure 5, after connecting Raspberry pi to the PC by entering its IP address. The Raspberry pi starts sending video from a camera to a PC. The object tracker starts to detect objects continuously. Using a camera and sensor data, the object detection model locates desired objects and classifies the detected object, Then the distance from the railway is measured using (an ultrasonic sensor). If the critical distance exceeds the limit, the control rail unit (Raspberry pi) warns or stops the train. Detecting a fire near a railway by measuring temperature with a K-Type Thermocouple Sensor, if it reaches the permitted threshold, the secondary station will stop the train. to maintain the flow of trains' movement, avoid and treat accidents. The average precision of each object class was calculated, and the mean accuracy rate for all classes in our proposed work increased and reach 94. 17% by effectively detecting all objects in a different size in a frame. In all results, the model improved swiftly in terms of precision, recall, and mean average precision before plateauing after about 100 epochs. It is able to detect humans with an average precision of 95.94%, and cars with 93.36.

7.2 Evaluation Metrics

In this section, the results of measuring precision Recall, and mAP for each object will be explained. the results of the experimental work had been evidenced pointed out that the deep learning YOLOv3 models which is an effective solution for object detection and classification task. Our object detection method was trained over 100 epochs (each epoch runs 8275 iterations). It took about three days. The confidence metric improved with time throughout each epoch, making detection closer to optimal.

7.2.1 Precision and Recall

Precision is the proportion of true predictions over all other predictions. The formula for precision is shown in (7.1). A Recall is defined as the ratio of positive instances accurately recognized by the detector over ground truth; the formula for a recall is shown in (7.2). First, we will clarify some fundamental concepts Before discussing the results using these metrics, True Positive (TP) denotes a correct prediction with an IOU greater than or equal to a predetermined threshold. False Positive (FP) refers to an incorrect detection of an IOU below the threshold. False negative (FN) means that no ground truth was detected [18].

$$Precision = \frac{TP}{TP + FP} \tag{7.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{7.2}$$

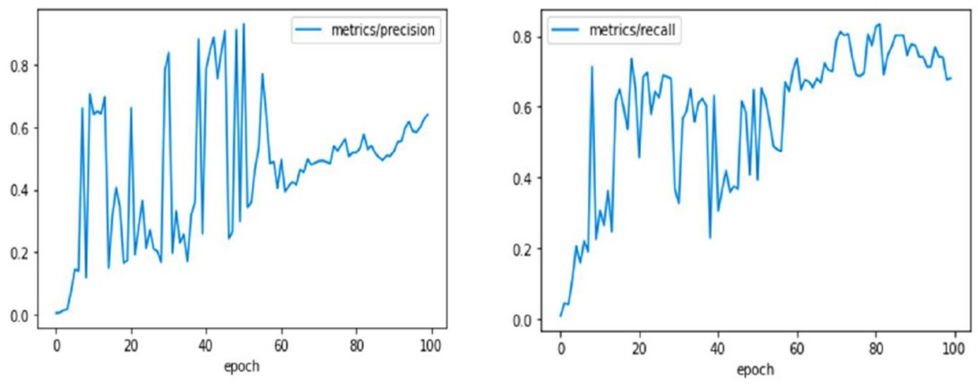

Figure 6: (a): precision curve with 100. (b): The Recall curve with 100 epochs

From figure 6(a) and (b), the precision measures how much of the bbox predictions are correct, and the recall measures how much of the true bbox was correctly predicted. In our model, the achieved results were very good. The precision is raised from 0.0039663 at epoch 1 to reach up to 0.64056. For recall, it is starting from 0.0088496 at epoch 1 to reach up to 0.68157. each epoch runs 8275 iterations.

7.2.2 mAP (Mean Average Precision)

In this paper, the mAP was chosen as the main object detection metric, representing the object detection algorithm's accuracy. The term "mean average precision" (mAP) refers to the average precision (AP) values for all classes. The average of the precisions of all predictions is known as the average precision (AP) [**?**]. Fig. 7 shows the average precision for each object and mAP of the proposed yolov3. as you can see, much better results were achieved. We improve both AP and the total detection accuracy, mAP.

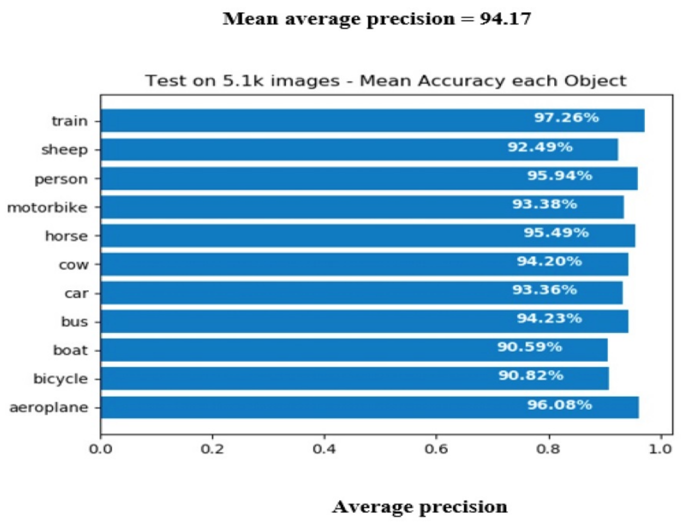

Figure 7: Mean accuracy results for each object.

8 Conclusion

It is vital to have a real-time, high-precision detection system. Deep learning and neural networks for object identification have made enormous progress in recent years, and this technology is now quite popular. The original intention of this thesis was to create a prototype that could be used to monitor the railway and detect the presence of objects in a critical area near the railway. Also, it provides early readings for detecting if there are fires near a railway to maintain the flow of trains' movements, avoid and treat accidents, and ensure a safe environment for the movement of train cars across the geographical area. The YOLOv3 object detection technique is used in this study, which is built using Tensor Flow as a deep learning framework. The data set PASCAL VOC 2012 was used for training, and the data set PASCAL VOC 2007 was used for testing. For model performance comparisons, most new and classic models are tested with the Pascal 2007 data set. We chose mAP as the main object detection metric. The mAP (mean average precision) is calculated to measure the performance of the model for each class in the dataset.

IOU (Intersection Over Union) is determined to be the fraction of the area of union between the anchor box and the ground truth. In this study, we used many optimization strategies of the proposed model, such as image augmentation using mixup and label smoothing, data augmentation methods, the adaptive spatial fusion feature, and the adaptive moment estimation optimizer (ADM optimizer). This increases the ability of the model to detect small target objects and dense objects, and when it detects any object in a critical area near the railway, it sends an order to the microcontroller for alarming or stopping the train. The proposed YOLOv3 has proven to be more efficient than other models, with an average precision rate of 94.17%.

References

[1] Y. Xiao, Z. Tian, J. Yu, Y. Zhang, S. Liu, S. Du and X. Lan, *A review of object detection based on deep learning*, Multimed. Tools Appl. **79** (2020), no. 33–34, 23729–23791.

[2] G. Karimi, *Introduction to Yolo algorithm for object detection*, Section.io.https://www. section.io/engineering education/introduction-to-yolo-algorithm-for-objectdetection/, (2021).

[3] S.J. Agha, M. Schenke and K. Hartmann, *Object detection using YOLOv3*, Bachelor Thesis, Hochschulbibliothek, Hochschule Merseburg, 2021.

[4] V. Meel, *How to analyze the performance of machine learning models*, viso.ai. 2021.

[5] Y.Q. Huang, J.C. Zheng, S.D. Sun, C.F. Yang and J. Liu, *Optimized YOLOv3 algorithm and its application in traffic flow detections*, Appl. Sci. **10** (2020), no. 9.

[6] T. Mikkonen, *Capacity monitoring using object detection algorithms*, Bachelor's Thesis, Metropolia Univ. Appl. Sci., 2021.

[7] A. Patel, *What is object detection?*, ML Research Lab, 2020.

[8] W.A. Ezat, M.M. Dessouky and N.A. Ismail, *Evaluation of deep learning YOLOv3 algorithm for object detection and classification*, Menoufia J. Electr. Eng. Res. **30** (2021), no. 1, 52–57.

[9] T. Wang, F. Yang and K.L. Tsui, *Real-time detection of railway track component via one-stage deep learning networks*, Sensors **20** (2020), no. 15, 1–15.

[10] H. Bandyopadhyay, *YOLO: real-time object detection explained*, www.v7labs.com, 2022.

[11] U. Almog, *YOLO V3 explained*, Towards Data Science, 2020.

[12] VOC, *TensorFlow*, https://www.tensorflow.org/datasets/catalog/voc#voc2007_default_config, 2022.

[13] R.K. Singh, *How to improve YOLOv3*, https://blog.paperspace.com/improving-yolo/, 2020.

[14] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie and M. Li, *Bag of tricks for image classification with convolutional neural networks*, Proc. IEEE/CVF Conf. Comput. Vision Pattern Recog., 2019, pp. 558–567.

[15] H. Zhang, M. Cisse, Y.N. Dauphin and D. Lopez-Paz, *Mixup: beyond empirical risk minimization*, arXiv preprint arXiv:1710.09412, (2017).

[16] P. Shah, *Label smoothing—make your model less (over)confident*, Towards Data Science, 2021.

[17] S. Liu, D. Huang and Y. Wang, *Learning spatial fusion for single-shot object detection*, arXiv preprint arXiv:1911.09516, (2019).

[18] Y. Gao, *A one-stage detector for extremely-small objects based on feature pyramid network*, Yini Gao Kth Royal Institute of Technology School of Electrical Engineering and Computer Science, 2020.

[19] O. Kivrak and M.Z. Gürbüz, *Performance comparison of YOLOv3, YOLOv4 and YOLOv5 algorithms: a case study for poultry recognition*, Avrupa Bilim Teknol. Dergisi **38** (2022), 392–397.