International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys.9, No.5 (Sep-2020)

http://dx.doi.org/10.12785/ijcds/090501

Smart Detection Under Different Weather Conditions

Ali Al Majed[1], Fred Lacy[1] and Yasser Ismail[1]

[1]*Electrical Engineering Department, College of Sciences and Engineering, Southern University and A&M College, Baton Rouge, LA, USA*

Received 6 Jun.2020, Revised 17 Jul. 2020, Accepted 29 Jul. 2020, Published 1 Sep. 2020

Abstract: Object detection is one of the most essential and challenging tasks in computer vision and deep learning. The main goal of object detection is to determine whether the image has an object from predefined categories and then to return the class and spatial location of that object. Researchers achieved a significant improvement in object detection in both speed and accuracy due to the ability to learn from raw pixels. There are three main stages in object detection: region proposal, feature extraction, and classification. The current state-of-art object detection algorithms are divided into two categories: two-stage and one-stage. The two-stage algorithms perform the first two stages separately, while the one-stage algorithms perform these two stages together. A two-stage algorithm like faster R-CNN is known for its superb accuracy, while the one-stage algorithms like YOLO and SSD are much faster than two-stage algorithms. Still, they lack accuracy, especially with a small object. This work targeted the accuracy, so the two-stage detection algorithms, faster R-CNN, were adopted as the basic structure for the detection network, evaluated under different weather conditions. The study implemented and tested the faster R-CNN with VGG16 as a feature extractor with images under differing weather conditions. First, the study trained the network under different training parameters to obtain the best detector. Then, the study tested and evaluated the two best detectors under different weather conditions. The results show that the accuracy of the detector is affected differently under different conditions, and more complex environments result in greater inaccuracy.

Keywords: Object detection, YOLO Algorithm, Faster R-CNN

1. Introduction

Object detection, as a subdivision of computer vision, is considered to be one of the difficult computer visions tasks. It deals with detecting instances of predefined categories in images. Object detection constitutes an essential and significant task in a wide range of applications like robot vision, pedestrian and face detection, security, digitalizing texts, intelligent video surveillance, automotive safety, and advanced driving assistant systems (ADAS). The objective of object detection is to determine the existence of any instances of an object from predefined categories (e.g., person, cat, car) in an image and return the class and spatial location (via bounding box) of that object.

Object detection is a challenging task, mainly due to the complexity of the background. The detector attempts to detect objects in the foreground and eliminate the background. The complexity of the background most affects the accuracy, and weather conditions tend to increase the complexity of the background. The weather conditions may also affect the lighting and the clarity of the image; therefore, these affect the detection accuracy.

Recently, object detection gained significant improvement by employing deep learning technologies. Deep Learning, as a powerful methodology for learning feature representation artificially in the image, led to notable development in object detection. The accuracy of the object detection algorithm provides a significant reason to choose which algorithm to use. Faster R-CNN [1] is known for its accuracy, but further evaluation must be applied to test its accuracy under different conditions. The purpose of this work is to evaluate the accuracy of faster R-CNN under different weather conditions.

Researchers focused on a hand-crafted method to extract a low-level feature to detect pedestrians by designing a manual algorithm. Recently, the researchers initially combined the hand-crafted method with a deep convolutional network to take advantage of the development of deep learning. Development in the feature extraction stage for the detection of the human has been in place since Dalal presented the Histogram of Oriented

E-mail: ali.almajed@gmail.com, fred_lacy@subr.edu, yasser_ismail@subr.edu

Gradients (HOG) to detect pedestrian [2]. To improve HOG, which computes multi-resolution image features explicitly, Pitor approximated these features via extrapolation and called this method aggregated channel features (ACF) [3]. ACF used boost decision trees with orthogonal (single feature) splits, while Locally Decorrelated Channel Features (LDCF) used decision trees with oblique (multi-feature) splits, which were more effective given the highly correlated data; Nam proposed this approach [4].

The paper is organized as follows, in section 2, a literature review is elaborated. Object detection algorithms are elaborated in section 3. Methodology and Implemented Algorithm is discussed in detail in section 4. The training procedure is described in section 5. Simulation and implementation are discussed in section 6. A conclusion will be drawn in section 7.

2. LITERATURE REVIEW

Recently, Sermant employed the convolutional neural network (ConvNet) in pedestrian detection, showing that ConvNet – with a few twists – yields competitive results. [5]. To further benefit from deep learning for pedestrian detection, Tome et al. [6] proposed a new architecture by analyzing and optimizing each step of the detection pipeline and called the pipeline (DeepPed). Object detection presents three main stages: region proposals, feature extraction, and classification, as seen in Figure 1.

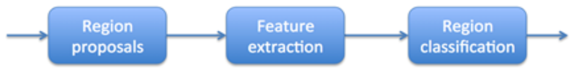


Figure 1. A common pipeline for object detection.

A. Region Proposals

The sliding window (Figure 2) is a traditional region proposals technique. The technique works by running a window from right to left and top to bottom to thoroughly search for a RoI, by the use of a different size of the window to detect an object at a different viewing location. However, the sliding window tends to generate a huge number of region proposals.

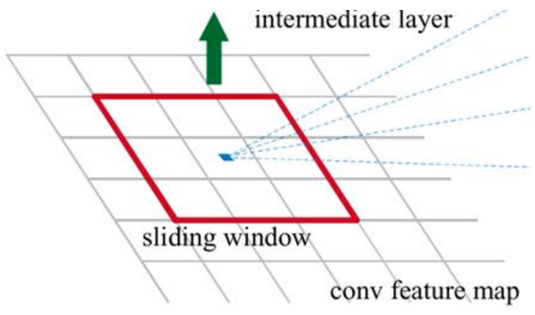


Figure 2. Sliding Window [7].

The other method used in the region proposals is Selective search [8]. This alternate method combines the strength of both the exhaustive search and segmentation methods to generate a region of proposals. The method applies the segmentation as a selective search that returns a small set of objects location as compared to the sliding window. Figure 3 shows the outputs of the selective search method. There exist three goals of the selective search algorithm: accounts for all object scales holds a diverse set of techniques to deal with all cases and displays a reasonably fast speed. Forming the basis of the selective search takes a hierarchical and bottom-up grouping algorithm. Then, the selective search algorithm uses four complementary similarity criteria to deal with all cases: color, texture, size, and fill.

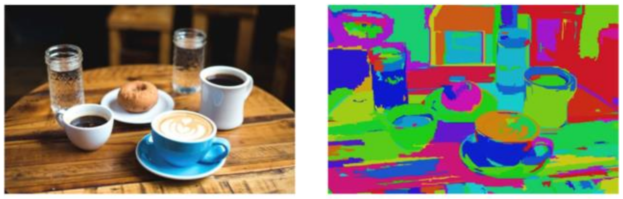


Figure 3. The image in the left is the input image, while the image in the right is the output of the selective search method [9].

The algorithm performed in two steps: first, it added all bounding boxes that corresponded with the segmented parts to the list of proposed regions. Second, the algorithm applies a bottom-up grouping of the adjacent segments based on similarity. All these measures are within the range [0, 1], which serves to facilitate the combinations of these measures. The last complementary is the sum of all of the four measures (Equation 7). The First complementary measure which measures color similarity is $s_{colour}(r_i, r_j)$ (Equation 1), which uses one-dimensional color histograms for each color channel using 25 pins. This leads to a color histogram $c_i = \{c_i^1, \dots, c_i^n\}$ for each region r_i with dimensionality n = 75, when the three color channels are used. The color histograms are normalized using the L_1 norm. The similarity is measured using the histogram intersection:

$$s_{colour}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k) \quad (1)$$

The color histograms can be efficiently propagated through the hierarchy by:

$$C_t = \frac{size(r_i) \times C_i + size(r_j) \times C_j}{size(r_i) + size(r_j)} \quad (2)$$

The size of a resulting region is simply the sum of its constituents:

$$size(r_t) = size(r_i) + size(r_j) \quad (3)$$

The second complementary measure which measures texture similarity is $s_{texture}(r_i, r_j)$ (Equation 4). The histogram is extracted using a bin of size 10 for each orientation for each color channel. This led to a texture histogram $Ti = \{t_i^1, \cdots, t_i^n\}$ for each region ri with dimensionality $n = 240$, using three color channels. The study used the L_1 norm to normalize the texture histograms. The histogram intersection measured the similarity:

$$s_{texture}(r_i, r_j) = \sum_{k=1}^{n} min(t_i^k, t_j^k) \qquad (4)$$

The research applied an efficient propagation of the texture histograms through the hierarchy in the same manner as the color histograms.

The third complementary measure, encouraging small regions to merge early is $s_{size}(r_i, r_j)$(Equation 5). This measure forced regions in S to be of similar sizes throughout the algorithm – $s_{size}(r_i, r_j)$ – defined as the fraction of the image that r_i and r_j jointly occupy:

$$s_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)} \qquad (5)$$

Size (im) denotes the size of the image in pixels

The fourth complementary measure measured how well the region r_i and r_j fit into each other, is $s_{fill}(r_i, r_j)$ (Equation 6). The notion intent is to fill gaps to avoid any holes. Only the regions' size and the containing boxes were used to keep the measure fast. Specifically, BB_{ij} is defined to be the tight bounding box around r_i and r_j. Now $s_{fill}(r_i, r_j) =$ becomes the fraction of the image contained in BB_{ij} which is not covered by the regions of r_i and r_j:

$$s_{fill}(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)} \qquad (6)$$

The final similarity measure is a combination of the above four:

$$s(r_i, r_j) = a_1 s_{color}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j) \qquad (7)$$

where $ai \in \{0,1\}$ denoted whether the similarity measure is used or not. As we aim to diversify our strategies, we do not consider any weighted similarities.

B. *Feature Extraction*

The second stage in object detection is feature extraction. The first breakthrough in the object detection was in 2005 when Dalal et al. [2] proposed the Histogram of oriented gradients (HOG). Dalal, accompanied by his team, detected a pedestrian with reasonable accuracy at that time. The HOG descriptor will read an image as an array of size n. Then, HOG used the distribution of the direction of gradients as features to determine whether there is a pedestrian. By approximation of multi-resolution features via extrapolation from nearby scales, Pitor et al. [3] improved HOG by combining the descriptor with channel features. This method is called the aggregated channel feature (ACF). By utilizing the high correlation, the study proposed an efficient feature transform to remove correlation in local neighborhoods; this approach was called the local decorrelated channel feature (LDCF) [4]. LDCF uses decision trees with oblique splits because the feature can be more productive with correlated features, such as when the topology of the resulting classifier matches the natural topology of the data.

2012 was the first year for the Convolutional Neural Network (CNN) to induce attention in object detection, once Kirzhevsky [10] used the CNN to win the ImageNet Large Scale Visual Recognition (ILSVR) 2012 competition. Using CNN, Kirzhevsky dropped the classification error from 26% to 15%. CNN takes as an input an image with a fixed size and then processes that image through different layers to extract a feature map. By employing a differentiable function, every layer carries the ability to perform a transformation from one volume to another.

Simo et al. [11] attempted to improve the content by increasing the depth of the network through the use of an architecture that employed very small convolutional filters with the size of (3 × 3), astride 1, the same padding, and by max-pooling a layer of 2 × 2 filters of stride 2. As a result, Visual Geometry Group-16 (VGG16) did not incorporate a large number of hyperparameters. The researchers used this architecture (Figure 4) to win the (ILSVR) competition in 2014 in the localization and classification tracks, respectively. As shown in Figure 4, VGG16 follows the same arrangement of convolutional and max-pooling layers consistently throughout the architecture. The last 3 layers construct 2 Fully Connected (FC) layers, followed by softmax function for output. VGG16 has 16 layers with weights, and is considered to be an outstanding model architecture, composed of a large network with around 138 million parameters.

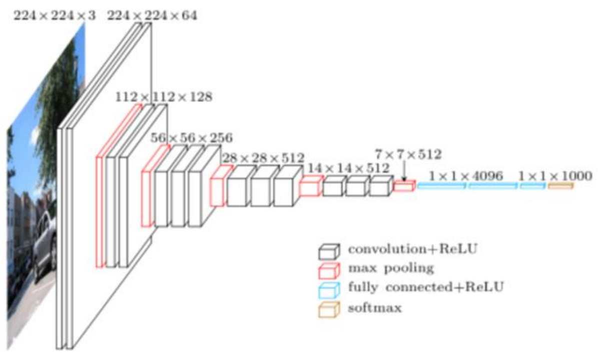


Figure 4. VGG16 Architecture.

3. Detection Algorithms

The detection algorithms divide into two categories: two-stage detectors and one-stage detectors [11]. Two-stage detectors perform the detection task in two stages. First, this category generates regions of interest (RoI) that may have an object and then extracts a feature map form those RoI's. The most popular approach in the two-stage detection algorithm is a faster R-CNN. On the other hand, the one-stage detection algorithm performs both steps together. The most popular one-stage detection algorithms are YOLO and single-shut-detector (SSD). YOLO and SSD eliminate the region proposals stage and perform the region proposals and feature extraction in one step.

One of the most popular two-stage detection algorithms is Faster R-CNN. Girshick et al. proposed a region-based convolutional neural network (R-CNN) [12] as an inspiration, due to the breakthrough in object detection using CNN. R-CNN combines the region-based method with a convolutional network which applies a high-capacity CNN to bottom-up region proposals, using selective search as an external region proposal method to extract about 2000 RoI. Regions are resized and then fed to the CNN network for feature extraction. The last step is to feed the patches to the linear Support Vector Machine (SVM) to predict the category of each patch. R-CNN does not share computation; instead, it performs a deep ConvNet forward pass for every RoI. It obtains a superb accuracy; however, it is very slow and computationally costly.

Figure 5 shows the system overview of R-CNN: (1) The network takes an input image with a fixed size, (2) uses the selective search method to generate around 2000 proposals, (3) uses the CNN to extract features for each proposal, and (4) uses linear SVM for classification [12].

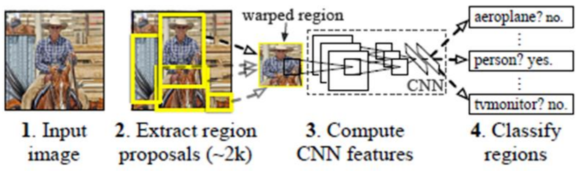


Figure 5. Object Detection System Overview [12].

Girshick proposed fast R-CNN [7] as an improvement of R-CNN, after recognizing the drawbacks. R-CNN detection is slower because it extracts the feature for every proposal. Moreover, the training is done in a multi-stage pipeline and is expensive in both space and time, extracting the feature of the entire input image using CNN. Also, it uses the selective search as an external feature extractor to generate a RoI, which when combined with the corresponding feature map, forms patches for object detection. Figure 6 shows the architecture of Faster R-CNN; first, it uses selective search to generate RoI, and CNN to extract the feature map from the input image. Then, the model utilizes an RoI pooling layer to combine RoI and feature maps. Finally, the model applies softmax and abounding boxes regressor to output the class and bounding box offset.

Faster R-CNN extracts the feature map from the input image, rather than extracting the feature from each RoI, similar to R-CNN – thus exhibiting why it is much faster than R-CNN.

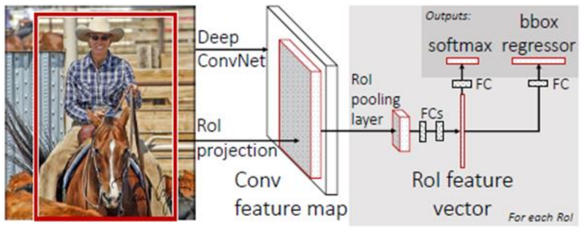


Figure 6. The Architecture of Fast R-CNN [7].

Fast R-CNN uses ROI pooling to wrap the patches to a fixed size, then feeds the patches to a fully connected layer for classification and localization. ROI pooling reduces the feature maps into an identical size by splitting the input feature map into a fixed number (k) of roughly equal regions, and then by applying max pooling on every region. Therefore, the output of ROI pooling is always a fixed number (k), regardless of the size of the input. Lastly, fast R-CNN employs the softmax function for classification and probability bounding box regressor for localization. One of the essential capabilities of fast R-CNN is that it trains all network weights with back-propagation. [7]

Faster R-CNN [1] replaces the selective search method, the slowest part of the fast R-CNN, by Region Proposal Network (RPN) to generate RoI. RPN shows a much less computational cost when compared to selective search (external), and also shares most of the computation with an object detection network. Besides, RPN ranks region boxes (called anchors) and then proposes the ones most likely containing objects (Figure 7). Anchor boxes handle the variations in aspects of ratio and scale of objects. The RPN may be trained end-to-end by stochastic gradient descent (SGD) and back-propagation.

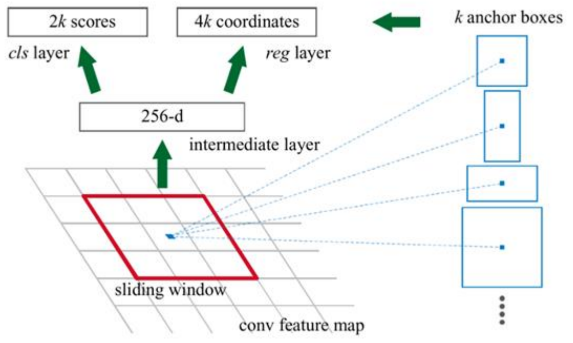


Figure 7. Region Proposal Network (RPN) [1].

Figure 8 shows high-level diagrams of the two-stage detection algorithm frameworks. The diagrams indicate the main differences between the region-based detection algorithms. R-CNN extracts the features from each RoI generated by the selective search method, which causes the process to become very slow. Fast R-CNN, on the other hand, extracts the feature map directly from the input image to speed up the process. Then it uses the RoI pooling layer to combine ROIs with feature maps. Since the selective search method is the slowest part of fast R-CNN, faster R-CNN replaces it with a Region Proposal Network (RPN).

The second category of the detection algorithm is a one-stage detection algorithm. One of the most popular algorithms in this category is YOLO [13]. YOLO deals with object detection as a regression problem to predict bounding boxes and class probabilities at once from the full image. Figure 9 shows how YOLO works; it divides every image into an SXS grid, and every grid cell predicts B bounding boxes and confidence scores for those boxes. The accuracy of the bounding boxes is reflected by the confidence score, as well as whether there is an object or not. The confidence score reflects how likely the box has an object, and how accurately it thinks the box is a boundary box. If no other object-located confidence score is zero, the finding is equal to the intersection over union (IOU) between the predicted box and the ground truth box. Each prediction box has five elements: x,y,w,h, and a confidence score. Figure 10 shows the architecture of YOLO: The network has 24 convolutional layers followed by two fully connected layers. and uses a reduction layer of size 1 × 1 to reduce the feature space from preceding layers [13].

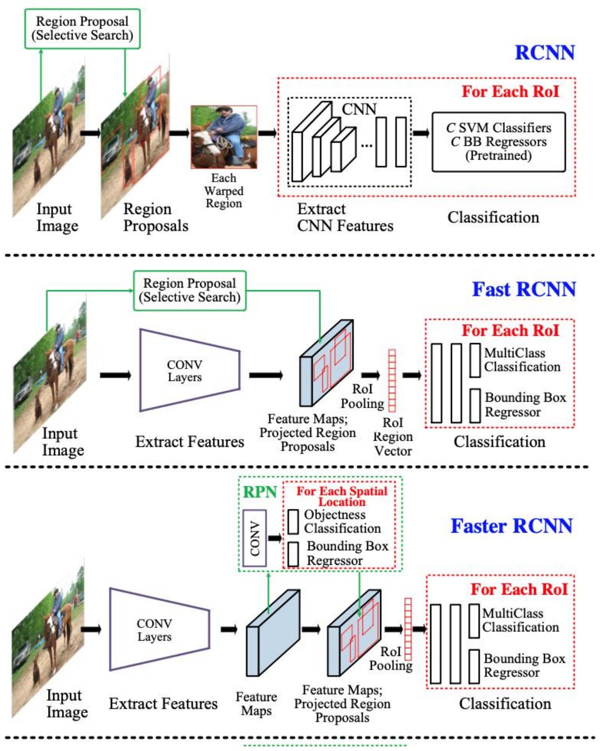


Figure 8. High-level diagrams of the two-stage detection algorithms frameworks.

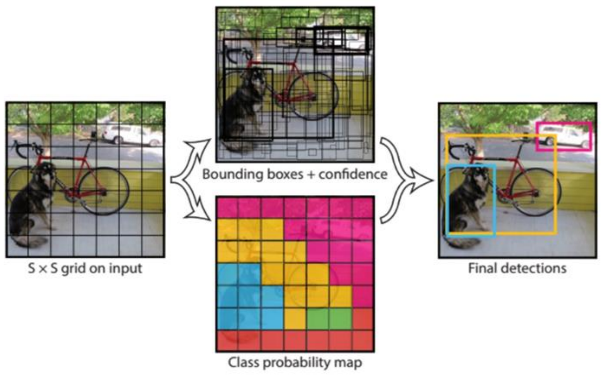


Figure 9.YOLO divides the image into SXS for each grid, and each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities [13].

Liu proposed that YOLOv2 [14] remains focused on improving the recall and localization of YOLO while maintaining the classification accuracy. The researcher simplifies the network and then makes the representation simpler to learn. YOLOv2 uses a high-resolution classification network that enhances accuracy by replacing the fully connected layer with an anchor box to predict bounding boxes, which increase the predicted boxes per image. These replace the classification model VGG-16 with a new proposed model, Darknet-19, that displays 19 convolutional layers and five max-pooling layers.

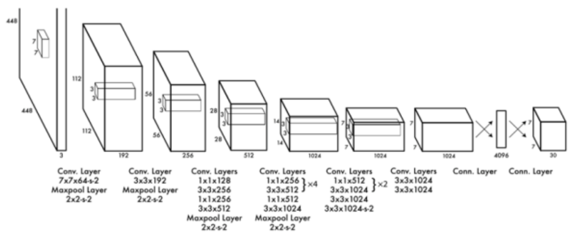

Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Figure 10. YOLO Architecture [13].

The other most popular one-stage algorithm is a single shot detector (SSD) [15]. SSD is a technique that uses a single, deep, neural network for detecting objects in images. SSD eliminates region proposals and a subsequent pixel or feature resampling stages, thus encapsulating all computation into a single network. This approach makes it simple, relative to other techniques that require region proposals. By employing small convolutional filters applied to feature maps, SSD predicts box offsets and category scores for a set of default bounding boxes. To achieve high detection accuracy, SSD produces predictions of different scales from feature maps and explicitly separates predictions by an aspect ratio. Figure 11 shows the architecture of SSD, which uses VGG-16 as a base network, then adds several feature layers. These layers predict the offset of different aspect ratios and scales and their associated confidences to the default box.

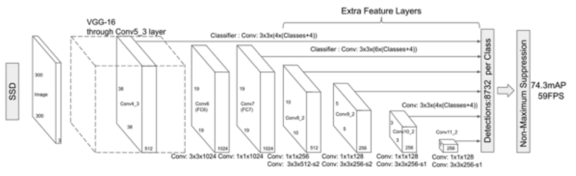


Figure 11. SSD Architecture [15].

Figure 12 shows the high-level diagrams of both YOLO and SSD. YOLO predicts detection, directly using a small set of candidate regions, employing fully connected layers at the top of the network for classification. On the other hand, SSD uses multiple scales at the top of the network to perform detection by operating on multiple convolutional feature maps; each predicts category scores and box offsets for bounding boxes of appropriate size.

Two-stage detectors use RPN to generate nearly 300 proposals to glean the best performance. The overall speed decreases because each region must pass through convolutional layers and fully connected layers for classification to fine-tune the bounding boxes. The ideas based on this approach gives one of the best performances in Common Objects in Context (COCO) detection challenge, even though these do not suit the real-time application [1].

Generally, one-stage detectors are not as accurate as two-stage detectors, even though the one-stage detectors are faster [17]. YOLO is extremely fast, but it is not as accurate as of the two-stage detectors. Moreover, YOLO struggles with small objects, because it only predicts one type of object in one grid [13]. Another disadvantage of The YOLO algorithm is that it uses a feature map solely on a single scale. SSD, on the other hand, considers prediction from various feature maps, instead of one, thus improving the accuracy over YOLO. Yet SSD still presents a lower performance, as compared to two-stage detectors [18].

The two-stage methods reached supremacy over the best performing object detection of deep, convolutional, neural networks [18].

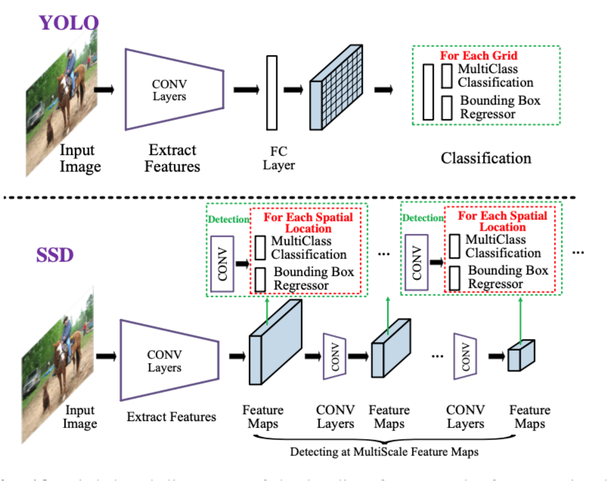


Figure 12. High-level Diagrams of the one-stage detection algorithm frameworks [16].

4. Methodology and Implemented Algorithm

To obtain the best result from the detection algorithm that is used in this work, the research applied the following methodology. First, the study preprocessed the training and testing data by labeling the class and bounding boxes in each image. Second, the study determined the optimal size of the training data set by training and then tested the model with four different dataset sizes and with three different number of anchor boxes (3,6,9) (Please see Table 1).

Then, the study used the optimal size of the dataset to further tune the training parameters, training the base model with a different number of anchors from 3 to 9. Finally, the study evaluated the best models showing the best accuracy results under different weather conditions. The weather conditions were showers, rain, snow, nighttime, high pedestrian traffic, and heavy background.

TABLE 1. THE SIZES OF THE DATASETS USED TO DETERMINE THE OPTIMAL SIZE OF THE TRAINING DATA, EACH DATASET WAS TRAINED WITH FOUR DIFFERENT ANCHOR BOXES.

Set Number	I				II			
Set Size (images)	40				60			
Number of Epochs	20				20			
Number of Anchors	3	6	9	12	3	6	9	12
Model Name	I-20-3	I-20-6	I-20-9	I-20-12	I-20-3	I-20-6	I-20-9	I-20-12
Set Number	III				IV			
Set Size (images)	80				100			
Number of Epochs	20				20			
Number of Anchors	3	6	9	12	3	6	9	12
Model Name	I-20-3	I-20-6	I-20-9	I-20-12	I-20-3	I-20-6	I-20-9	I-20-12

A. Implemented Algorithm

The algorithm divided into four steps: loading the dataset, creating a faster R-CNN detection network, training, and evaluation. The first step was loading the dataset. This step involved loading the ground truth data for training and testing. The ground truth data presented image information together with the classes and the spatial location of bounding boxes. The study then created datastores for loading the image, as well as label data for training and evaluation.

The second step was Creating faster R-CNN Detection Network. This step involved preprocessing the training data, which included image resizing, data augmentation, and choosing the network parameters. The preprocessing of the training data included image resizing and augmentation. The resizing of the image was required since the input layer expected the input image to be the same size. The study used the data augmentation to increase the number of training images artificially, as well as to also reduce overfitting.

The faster R-CNN parameters included a selection of the feature extraction CNN, feature layer, number of anchor boxes, and number of classes to be detected. The third step was to choose the training options that contribute more to the accuracy of training network that involved maximum epochs, min-Batch sizes, and learning rates.

The last step after training the network was evaluating the resulted trained detector. In this step, the trained detector using the trained data by comparing the prediction with the ground truth data. The output showed the average precision (AP) of the comparison.

B. Data Preprocessing

<u>Dataset</u>

The research obtained images used in this work from Penn-Fudan Database [20]. The process took images from campus and urban streets, and during the daytime as well, with the pedestrian shown in a straight-up position. The images divided into five sets of different sizes. The training sets numbered four: the first set consisted of 40 images. The second set showed 40 images from the first set, plus an additional 20 images. Each set would encompass all the images from the previous set, plus 20 additional images. The last set was the testing set, which displayed uniquely 20 images for testing.

<u>Ground Truth Labeling</u>

The study labeled all the sets of images with the class and bounding boxes, manually using an Image labeler application from MATLAB. The research applied the ground truth data of the labeled images, together with other images, for the training and testing of the model. This work labeled the images with a class identified as "pedestrian."

<u>Evaluating Metrics</u>

The accuracy of the prediction of both the class and bounding boxes is evaluated using average precision (AP). Over the last years, AP was the most commonly used evaluation metrics in object detection [19]. AP may be defined as the average precision under different recalls.

$$\text{AP} = \int_0^1 p(r)\ dr \qquad (8)$$

The precision is the total true positive over the total of true positive and false positive, or the ratio of true object detections to the total number of objects that the classifier predicted.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \qquad (9)$$

Recall is the true positive over the total of a true positive and false negative or the ratio of true object detections to the total number of objects in the data set.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \qquad (10)$$

Intersection over Union (IoU) threshold is used to measure the object localization accuracy. IoU is the overlap of a predicted versus ground truth bounding box for an object.

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \qquad (11)$$

Determining the optimal training samples

To determine the optimal number of images for the training dataset, the network trained with four sets of images (40, 60, 80, and 100) and a different number of anchor boxes (3,6, 9 and 12). All the models (See Table1) trained with the SGD as with a momentum of 0.9, a learning rate of 0.001, a mini-batch size of 2, and 20 epochs. Then, the study tested and evaluated all the resulted detectors with the same testing dataset that showed 20 images and evaluated with a mean AP.

Tuning Training Parameters

The study used the training set with 100 images as a dataset to train the models with many anchor boxes between 3 and 9 and maintained the same training parameters.

The research increased the number of epochs gradually by 10 with each number of anchors. All the trained networks with less than 80% average precision were eliminated. Finally, all the remaining networks were trained with more epochs until the precision decreased.

Evaluation of the best model under different weather Conditions

The study tested and evaluated the best-obtained models under different weather conditions. The conditions were showers, rain, snow, nighttime, high pedestrian traffic, and a heavy background. The research added the effects of showers, rain, snow, and nighttime to the images artificially. To consider an image under high pedestrian traffic, it should have at least three pedestrians. The Heavy Background category has images showing many objects in the background. Figure 13 shows examples of different weather conditions that are used in this paper.

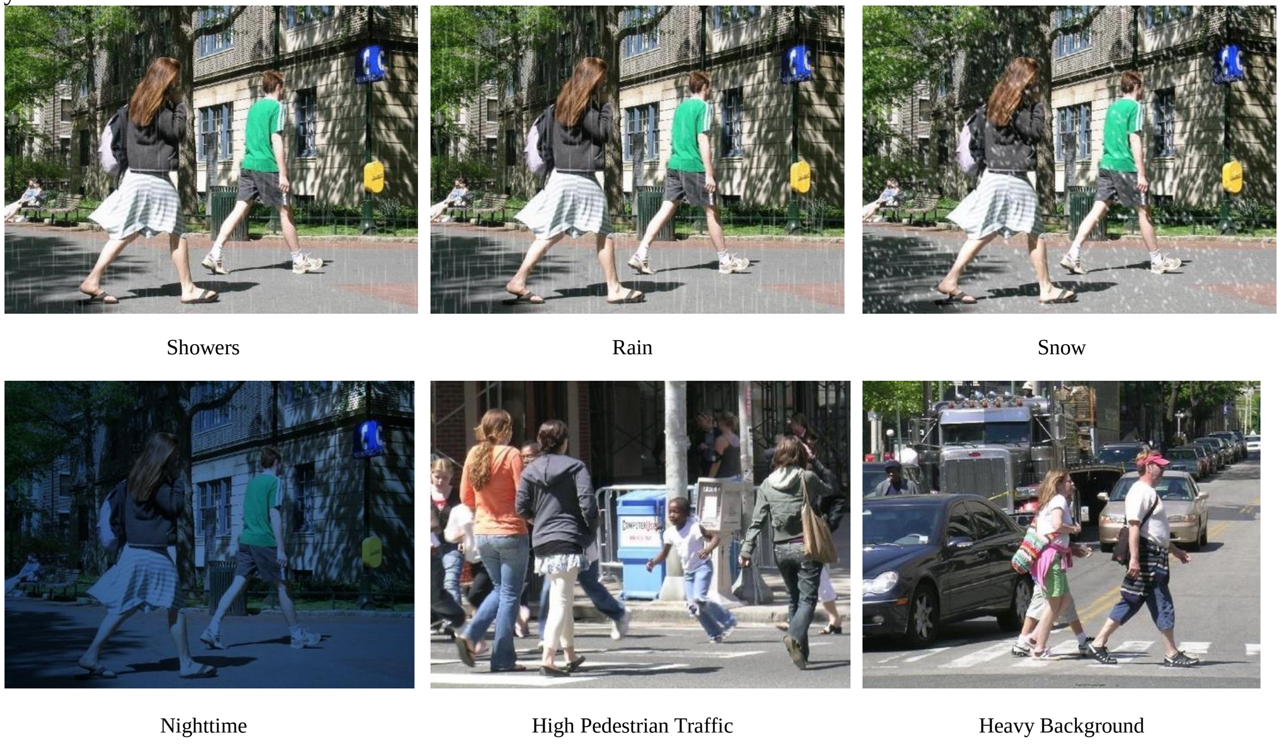


Figure 13. Examples of image categories used for evaluation. The original images were obtained from [20].

5. TRAINING

A. CNN Layers

To understand how to train a CNN, we should first understand the main layers used to build a CNN. The main layers used to build the CNN architecture are the Convolutional layer (conv), the Rectified Linear Unit (ReLU) layer, the Pool layer, and the Fully connected layer (FC) (Please refer to Figure 14).

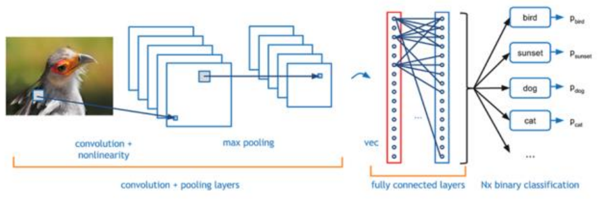


Figure 14. CNN Architecture Layers.[21]

The convolutional layer was the first layer in the Convolutional Neural Network (CNN). This layer required an image with size [W × H × D], where W, H, and D are width, height, and depth respectively, and then run a filter (kernel) with size [k × k × D], where k is filter size, to do a product multiplication with the receptive field – starting from the right corner and moving one pixel at a time (when stride=1). The output of the conv layer exhibited a feature map with size [(W - k+1) × (H+1-k) × D].

The Rectified Linear Unit (ReLU) represented the activation function. The ReLU increased the nonlinear properties of the model and also the overall network, applying an elementwise activation function f(x) = max(x,0) (Figure 15). The size of the feature map was not changed.

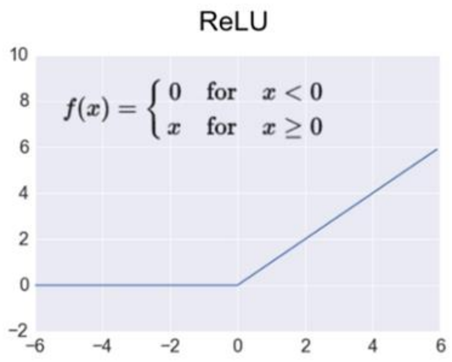


Figure 15. The rectifier Linear Function is used to increase the nonlinearity of the network [21].

The Pooling Layer was a down-sampling layer, which applied a filter (normally 2 × 2, and stride =2) to the input volume and outputted the maximum (max pooling), or average (average pooling). The output spatial dimension reduced to half with 2 × 2 filters (Figure 16).

The Fully connected Layer (FC) drew the input volume from the preceding layer and determined which feature correlated to a particular class. The output was an N-dimensional vector containing an N number of classes from which the program will choose.

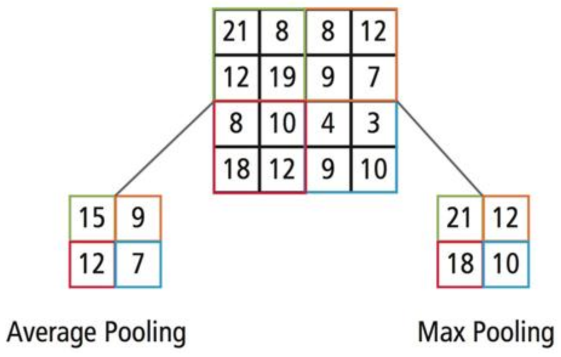


Figure 16. Average and Max Pooling [22].

B. What is Training?

The goal of the training was to optimize the weights within the model by solving the optimization problem. This goal may be achieved by employing an optimizer to minimize the loss function to be as close to zero as possible. There were three well-known optimizers: Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMSP), and Adam. The study used the SGD with a momentum optimizer in this work because SGD is the most known optimizer in the deep learning tasks. An SGD algorithm updated the weight of the network, with every iteration using the back-propagation algorithm.

C. Training Process

The training incorporated four steps: forward pass, loss function, backward pass, and weight update. The first step was to pass an image array of numbers throughout the entire network. The image array would then pass from the input layer through all the hidden layers until reaching the output layer.

The loss function (12) calculated the error by comparing the prediction and the true label of the image. The input to the loss function was the output of the forward pass and ground truth data. The overall loss was the classification loss and regression loss.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}\,(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}\,(t_i, t_i^*) \quad (12)$$

$$\begin{aligned} t_x &= (x - x_a)/\,w_a\,, & t_y &= (y - y_a)/\,h_a\,, \\ t_w &= \log(\frac{w}{w_a})\,, & t_h &= \log(\frac{h}{h_a})\,, \\ t_x^* &= (x^* - x_a)/\,w_a\,, & t_y^* &= (y^* - y_a)/\,h\,, \quad (13) \\ t_w^* &= \log(\frac{w^*}{w_a})\,, & t_h^* &= \log(\frac{h^*}{h_a})\,, \end{aligned}$$

$$L_{loc}(t^u, v) = \sum_{i\in\{x,y,w,h\}} smooth_{L_1}\,(t_i^u - v_i), \quad (14)$$

in Which $smooth_{L_1}\,(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$

In equation 12, i is the index of an anchor in a mini-batch, the number of training samples present in a single batch, and p_i is the predicted probability of anchor i being an object. The ground-truth label p_i is 0 if the anchor is negative and is 1 if the anchor is positive; t_i is a vector representing the 4 parameterized coordinates (Equation 13) of the predicted bounding box; t_i is also that of the ground-truth box associated with a positive anchor. The classification loss L_{cls} represents log loss over two classes (object vs. not object). For the regression loss, we used L_{reg} $(t_i\,, t_i^*)$ = R$(t_i\ - t_i^*)$, where R is the robust loss function (smooth L_1) in Equation 14. The term $p_i^* L_{reg}$ means the regression loss is activated only for positive anchors (p_i^*=

1) and is disabled otherwise (p_i^*= 0). The outputs of the cls and reg layers consist of {p_i} and {t_i}, respectively. N_{cls} and N_{reg} are normalized and weighted by a balancing parameter λs. The cls term in Equation 12 is normalized by the mini-batch size, and the reg term is normalized by the number of anchor locations.

After that, the backward pass determined which weight contributed more to the loss to reduce it through utilizing optimization. Put another way, the backward pass is the process of counting changes in weights.

The last step is updating the weight. First, a set of arbitrary weights used to initialize the network. After each epoch, a single pass of the data through the model, the weights updated by computing the gradient (dL/dW) of the loss function with respect to each of the weights that were set. Then, the new weight (W) is the current weight (Wi), subtracted by the gradient $W = Wi - \frac{dL}{dW}$.

6. SIMULATION AND IMPLEMENTATION

A. Determining the optimal training samples

Table 2 presents the result of training the base network, with a different set of training images and a different number of anchor boxes. This table reflects the result of evaluating all the models trained and tested with different training images size (40, 60, 80, and 100), and also with a different number of anchor boxes (3, 6, 9, 12). The result listed in the table includes the mean Intersection over Union (IoU) and Average Precision (AP) as an evaluation metric of the model accuracy.

Figure 17 shows the result of each training set; the result indicates the AP (y-axis) as an evaluation measure of the accuracy and the number of anchors (x-axis). Figure 17-A evidences the result of evaluating 40 images of training dataset models, with 3,6,9 and 12 anchors. The best accuracy result with 40 images training set models obtained 0.75 when trained with 6 anchors. Figure 17-B shows the result of evaluating 60 images training dataset models when trained with the same number of anchors. The best accuracy result was obtained 0.83 when trained with 9 anchors. Figure 17-C shows the result of evaluating 80 images training dataset models when trained with the same number of anchors. The best accuracy result was obtained 0.83 when trained with 6 anchors. Figure 17-D shows the result of evaluating 100 images training dataset models when trained with the same number of anchors. The best accuracy result was obtained 0.83 when trained with 3, 6, and 12 anchors.

The models trained with 100 images dataset show the best overall AP results; therefore, these were chosen for further tuning with training parameters.

TABLE 2. THE RESULT OF TRAINING THE BASE NETWORK WITH A SET OF 40,60,80 AND 100 IMAGES AND ALSO THE NUMBER OF ANCHORS WAS CHANGED FROM 3,6,9,12. THIS TABLE HAS THE RESULTED AVERAGE PRECISION (AP), AND ALSO THE MEAN INTERSECTION OVER THE UNION.

Sr.	Model	Training Images	Anchor Boxes	mean IoU	AP	Sr.	Model	Training Images	Anchor Boxes	mean IoU	AP
1	I-20-3	40	3	0.71	0.60	9	III-20-3	80	3	0.66	0.77
2	I-20-6		6	0.80	0.75	10	III-20-6		6	0.79	0.83
3	I-20-9		9	0.85	0.65	11	III-20-9		9	0.84	0.70
4	I-20-12		12	0.88	0.74	12	III-20-12		12	0.86	0.81
5	II-20-3	60	3	0.70	0.81	13	IV-20-3	100	3	0.70	0.83
6	II-20-6		6	0.81	0.78	14	IV-20-6		6	0.80	0.83
7	II-20-9		9	0.84	0.83	15	IV-20-9		9	0.77	0.82
8	II-20-12		12	0.86	0.82	16	IV-20-12		12	0.86	0.83

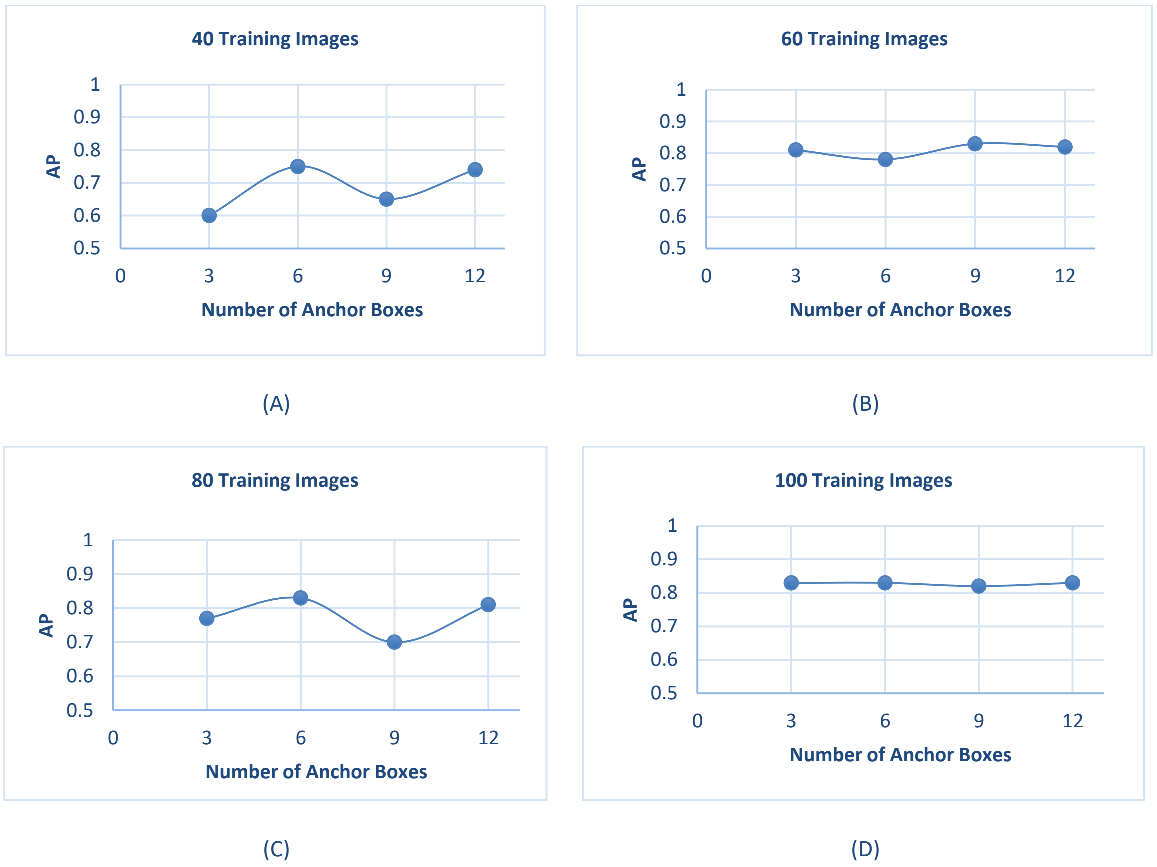


Figure 17. The result of training the base network with a different number of images, the plot of the accuracy versus the number of anchors with each set.

B. *Tuning Training Parameters*

The study trained and evaluated models with 100 images training datasets with more of a range in the number of anchor boxes from 3 to 9 to determine the optimal number of anchors, but kept the other training parameters fixed. This step helped to tune the base network to get the best model.

The number of epochs increased gradually with each number of anchors. The research eliminated all the trained networks with less than 80% average precision. Then all the remaining networks were trained with more epochs until there was no improvement in the accuracy.

Table 3 presents the result of training and the model with 100 images dataset, showing many anchors from 3 to 9. The model (IV-20-4), trained with 20 epochs and 4 anchors, obtained an AP result of 0.81. Since AP is more than 0.8, the model, trained with 30 epochs, gave a result of 0.83. Therefore, we trained the model further with 40 epochs, yet attained the same result of 0.83. So

this study can identify that the best result from the model with 4 anchors showed when the model trained with 30 epochs. The research applied the same method with all numbers of anchors. The comparison between all numbers of anchors and the AP result is shown in Figure 18. This figure shows not only the AP result but also the number of epochs used to reach this result for all number of anchors. The best results obtained were 0.86 when model IV-30-7 and IV-30-8 trained with 7 and 8 anchors and 30 epochs.

C. *Testing and evaluating*

The tuning of the training parameters provides the best accuracy result, which is 86% with two models trained with 7 and 8 anchors: Models IV-30-7 and IV-30-8, where both were trained with 30 epochs (Table 3). Generally, the result obtained from Model IV-30-7 is better than the second model IV-30-8 when tested under different conditions.

The resulting detectors from these two models tested under different weather conditions: showers, heavy rain, snow, nighttime, and heavy background. Generally, Model IV-30-7 performs better than the other model except when evaluated with showers and high traffic, showing only a 1% difference. Figure 19 shows the result of testing both models under different conditions.

Table 4 shows drops of the average precision (AP) for both Models IV-30-7 and IV-30-8 when tested under different weather conditions. The accuracy dropped with 3% and 4% when tested under showers and heavy rain, but dropped by 8% under snow weather since snow affects the background more than rain. The nighttime had a slight drop (1%) to the accuracy, but both the high traffic and the heavy background were the most dropped in terms of accuracy.

The conclusion is that the more complex the background, the more the loss in the accuracy. The snow affects the complexity of the background more than the rain, which is why the drop in the accuracy increases with the snow. Since nighttime only changes the brightness of the image and does not change the background, there was only a slight drop in accuracy. When evaluating a model with a complex background, such as high traffic and heavy background, the accuracy drops considerably.

TABLE 3. THE RESULT OF THE TRAINED NETWORK WITH 100 IMAGES AND DIFFERENT NUMBER OF ANCHOR BOXES AND EPOCHS, THE BEST ACCURACY RESULT OF 0.86 WAS OBTAINED WITH MODEL IV-30-7/8

Model	Anchor Boxes	epochs	AP	Model	Anchor Boxes	epochs	AP
IV-20-3	3	20	0.83	IV-20-7	7	20	0.85
IV-30-3		30	0.81	IV-30-7		30	0.86
IV-40-3		40	0.76	IV-40-7		40	0.85
IV-20-4	4	20	0.81	IV-20-8	8	20	0.83
IV-30-4		30	0.83	IV-30-8		30	0.86
IV-40-4		40	0.83	IV-40-8		40	0.81
IV-20-5	5	20	0.77	IV-20-9	9	20	0.82
IV-20-6	6	20	0.78	IV-20-9		30	0.83
				IV-40-9		40	0.83

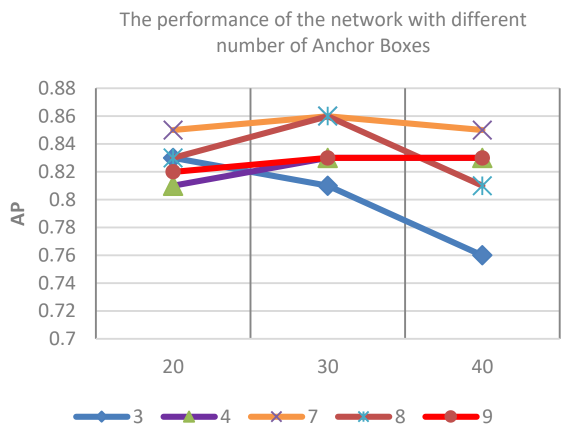
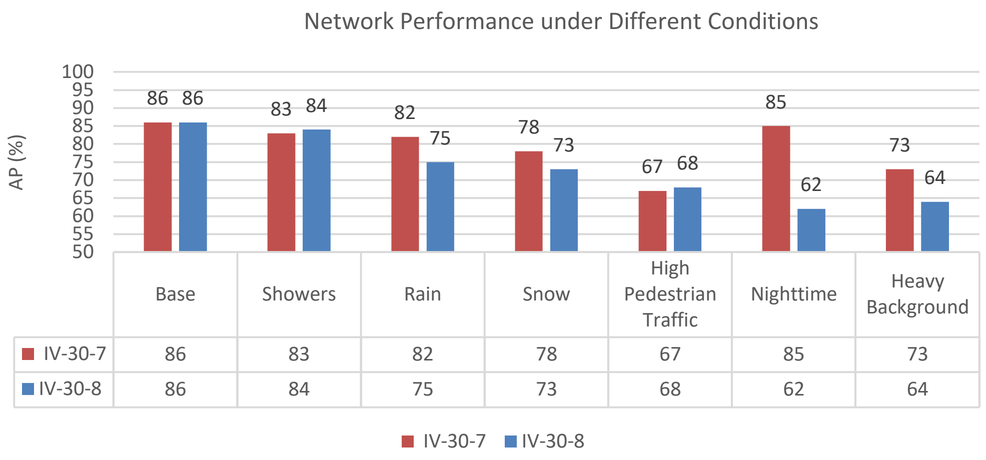


Figure 18. The accuracy of the network with a different number of anchor boxes. This graph shows the plot of average precision versus each number of boxes

D. Implementation Details

The base network was implemented in a CPU with a processor of 3.1 GHz Dual-Core Intel Core i5 and 8 GB 2133 MHZ LPDDR memory. The software used is MATLAB R2019b (student license#40861480) with both image processing and computer vision toolbox, as well as machine learning and a deep learning toolbox. The faster R-CNN [1] structure was implemented with a pretrained VGG16 [11] for feature extraction.

Figure 20 shows some of the examples of output images from the detector. Each image has a rectangular box around each pedestrian. Also, each box will have a confidence score associated with each box. Figure 22-A shows some examples of the output image when the original images tested with the detector, accompanied by 7 anchor boxes. Furthermore, Figure 22-B/C/D/E/F shows examples of the output images from all other categories used in this work. It is worth mentioning that the number associated with the bounding box, in Figure 22, indicates the confidence of the detector regarding the accuracy of the detection. The close the number to “1”, the more confidence that the object shows as “Pedestrian.”

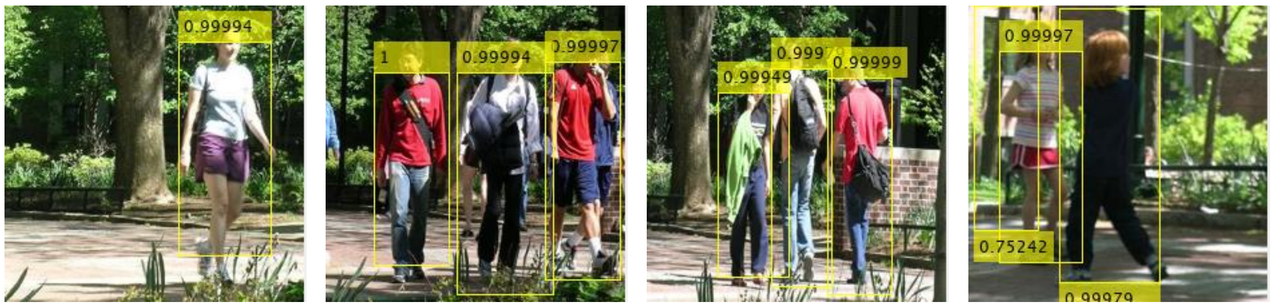


	Base	Showers	Rain	Snow	High Pedestrian Traffic	Nighttime	Heavy Background
🟥 IV-30-7	86	83	82	78	67	85	73
🟦 IV-30-8	86	84	75	73	68	62	64

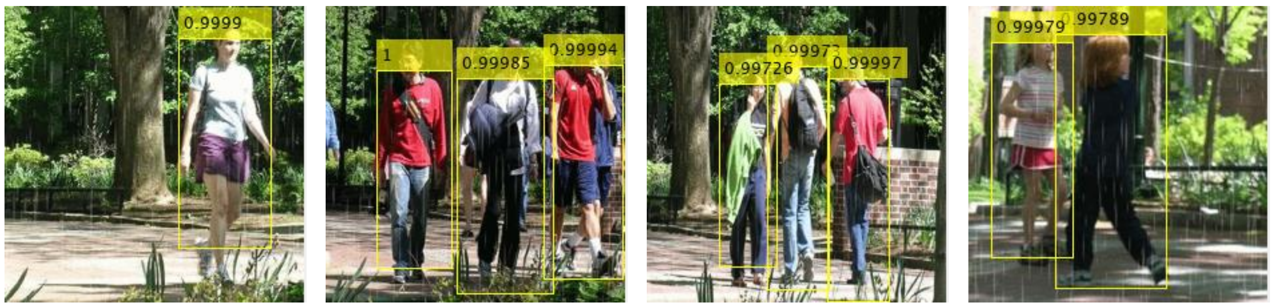
Figure 19. Network Performance of Model IV-30-7 and IV-30-8 when they tested and evaluated under showers, rain, snow, high pedestrian traffic, and heavy background Conditions.

TABLE 4. THIS TABLE SHOWS THE DROP IN THE ACCURACY AFTER TESTING THE MODEL WITH DIFFERENT WEATHER CONDITIONS.

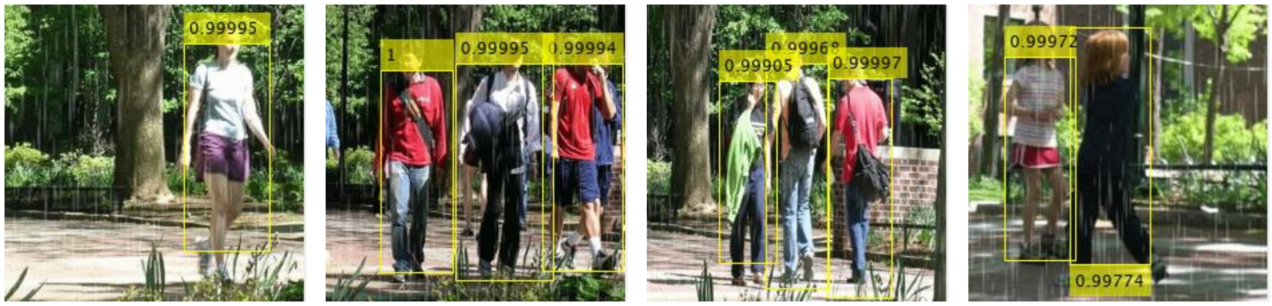
Model		**Showers**	**Heavy Rain**	**Snow**	**High Traffic**	**Nighttime**	**Heavy Background**
IV-30-7	**AP (%)**	83	82	78	67	85	73
	Drop	3	4	8	19	1	13
IV-30-8	**AP (%)**	84	75	73	68	62	65
	Drop	2	11	13	18	24	21



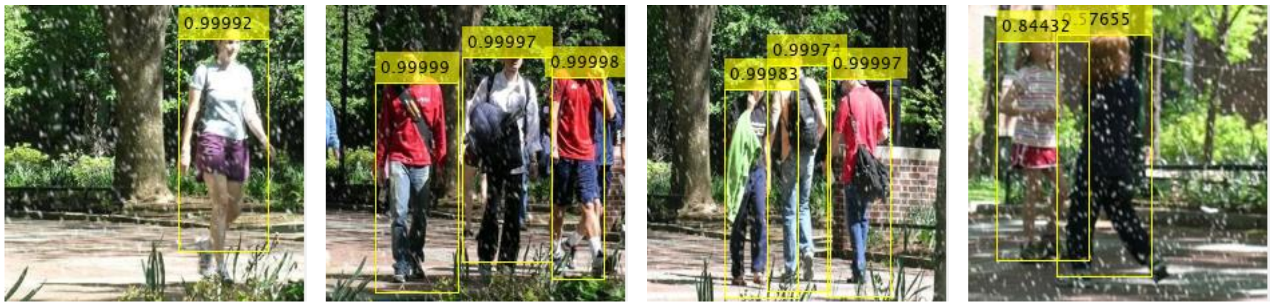
(A) Original (no affect)



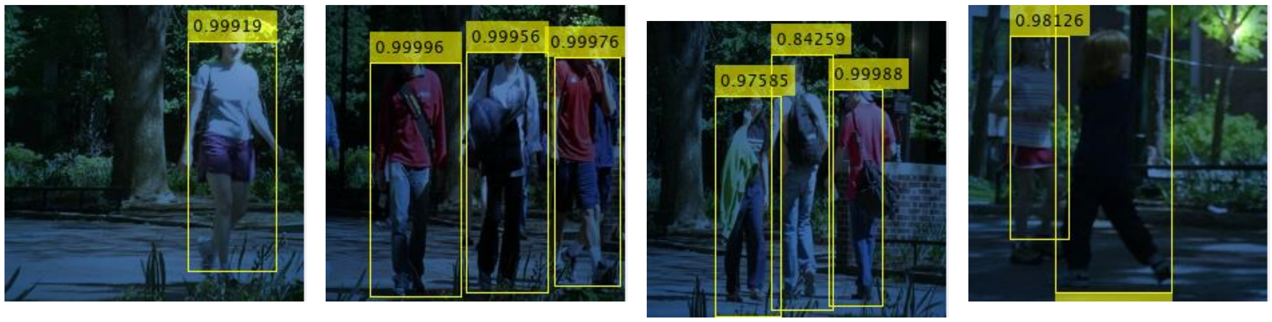
(B) Shower



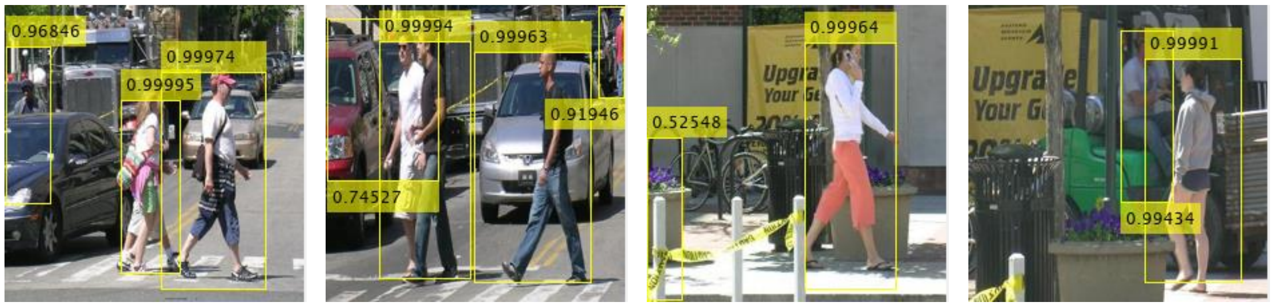
(C) Rain



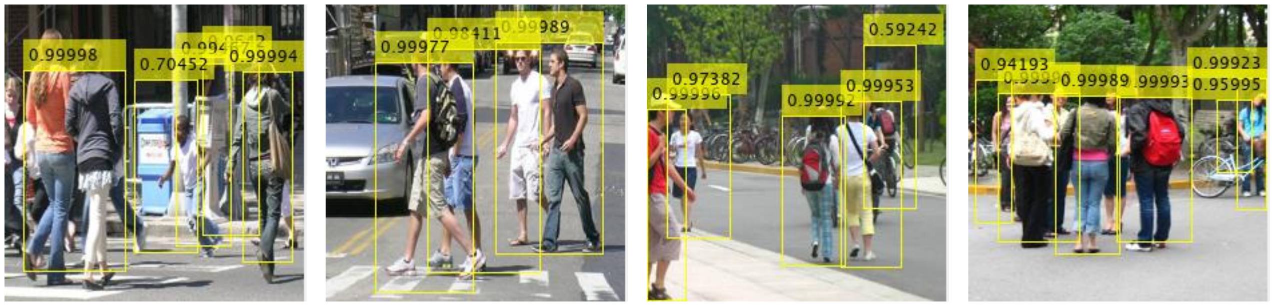
(D) Snow



(E) Nighttime



(F) Heavy Background



(G) High Pedestrian Traffic

Figure 20. Examples of detector output images, the number at the top of the rectangle is the probability that this rectangle has an object.

7. Conclusion and Future Work

In this paper, the study evaluates the accuracy of the state-of-art smart object detection algorithm, faster R-CNN, under different weather conditions. The research implemented faster R-CNN with vgg16, tuning and evaluating the parameters for the best result. In the first chapter, the study explored the object detection methods employed with CNN to automatically detect and classify object detection. The study explained the methods used for generating RoI, then considered the convolutional neural networks that are vgg16. Finally, the study explored the state-of-art object detection algorithms in both two-stage and one-stage algorithms. Determining the optimal number of images for the training dataset was the first step in this work. Then the study tuned the training parameters tuned to obtain the best setup for the base model that in turn led to a more accurate detector. The research implemented a two-stage detection algorithm, faster R-CNN with VGG16, as a base model. Finally, the best two results in detectors were used to evaluate the performance of the model under different conditions. The final results after testing the two models with the different sets showed each model representing one weather condition. The accuracy was not decreased when the model tested for rain and snow weather conditions. Yet the accuracy decreased as the background became more complex, i.e., the study tested the model with sets of high traffic and heavy background images. In general, the results obtained showed good accuracy, even when subjected to different weather conditions. The more the weather affected the background, the more the accuracy decreased.

Acknowledgment

The authors would like to thank Electrical Engineering Department at Southern University and A&M College, Baton Rouge, USA for the great support provided in full to finalize this work.

References

[1] R. Shaoqing, K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in Advances in neural information processing systems, 2015.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), 2005.

[3] P. Dollár, R. Appel, S. Belongie and P. Perona, "Fast feature pyramids for object detection," IEEE transactions on pattern analysis and machine intelligence, vol. 36, no. 8, pp. 1532-1545, 2014.

[4] W. Nam, P. Dollár and J. H. Han, "Local decorrelation for improved pedestrian detection," in Advances in neural information processing systems, 2014.

[5] P. Sermanet, K. Kavukcuoglu, S. Chintala and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2013.

[6] D. Tomè, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi and S. Tubaro, "Deep convolutional neural networks for pedestrian detection.," Signal processing: image communication, no. 47, pp. 482-489, 2016.

[7] R. Girshick, "Fast r-cnn," in Proceedings of the IEEE international conference on computer vision, 2015.

[8] J. R. Uijlings, K. E. V. De Sande, T. Gevers and A. W. Smeulders, "Selective search for object recognition," International journal of computer vision, vol. 104, no. 2, pp. 154-171, 2013.

[9] V. S. Chandel, "Learn Open CV," [Online]. Available: https://www.learnopencv.com/selective-search-for-object-detection-cpp-python/. [Accessed 17 March 2020].

[10] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," 2012.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition.," in arXiv preprint arXiv:1409.1556, 2014.

[12] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587, 2014.

[13] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

[14] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szeged, S. Reed, C.-Y. Fu and A. C. Berg, "Ssd: Single shot multibox detector," in European conference on computer vision, Springer, Cham, 2016.

[16] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu and M. Pietikäinen, "Deep learning for generic object detection: A survey.," International journal of computer vision, vol. 128, no. 2, pp. 261-318, 2018.

[17] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi and I. Fischer, "Speed/accuracy trade-offs for modern convolutional object detectors.," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017.

[18] S. Agarwal, J. O. Du Terrail and F. Jurie, "Recent advances in object detection in the age of deep convolutional neural networks.," arXiv preprint arXiv, no. 1809.03193, 2018.

[19] Z. Zou, Z. Shi, Y. Guo and J. Ye, "Object detection in 20 years: A survey.," arXiv preprint arXiv:1905.05055, 2019.

[20] "Penn-Fudan Database for Pedestrian Detection and Segmentation," [Online]. Available: https://www.cis.upenn.edu/~jshi/ped_html/. [Accessed 1 March 2020].

[21] [Online]. Available: https://towardsdatascience.com/deep-learning-2-f81ebe632d5c. [Accessed 18 March 2020].

[22] [Online]. Available: https://medium.com/@Aj.Cheng/convolutional-neural-network-d9f69e473feb. [Accessed 2 March 2020].

Mr. Ali Al Majed is an instrumentation and control engineer (I&C). He received his B.Sc. degree in Systems Engineering from King Fahd University of Petroleum and Minerals – Saudi Arabia, in 2004. He received his MEng in Electrical Engineering from Southern University and A&M College at Baton Rouge – USA, in 2020. In August 2004. He joined Honeywell Turki Arabia Ltd – Saudi Arabia as an assistant Engineer. In May 2005, he joined Dar Al-Riyadh – Saudi Arabia as a Jr. Instrumentation Engineer. In June 2007, he joined Snamprogetti SA Ltd – Saudi Arabia as Switch-over Instrumentation and Automation Supervisor. In Feb 2009, he joined Sami Saif Industrial – Saudi Arabia Services as a Warranty Control Engineer. In Feb 2010, he joined KBR-AMCDE– Saudi Arabia as Instrumentation and Control Engineer. In May 2013, he joined SNC-Lavalin Fayez Engineering as Instrumentation and Control Engineer. He has the opportunity to gain experience from working with international companies such as Snamprogetti, KBR, and SNC-LAVALIN. He has experience in the design of instrumentation and control systems (such as DCS, ESD, VMS, and TMS). His experience ranges in all levels of FEED design, detail engineering, FAT, SAT, construction, pre-commissioning, commissioning, and implementation of the control systems.

Dr. Fred Lacy received the B.S.E.E. degree and Ph.D. degree in electrical engineering from Howard University, Washington, DC, in 1987 and 1993, respectively, and the M.S.E. degree from Johns Hopkins University, Baltimore, MD, in 1989. He was a Postdoctoral Fellow in the Bioengineering Department, University of California, San Diego, for four years, where he performed research in the area of biosensors. He was with the US Food and Drug Administration, where he performed medical device reviews. In 2002, he joined the Electrical Engineering Department, Southern University and A&M College, Baton Rouge, LA, where he is engaged in research and teaches courses in solid-state electronics, electrical and electronic circuits, and electronics-based sensors.

Dr. Yasser Ismail received his BS. degree in Electronics & Communications Engineering from Mansoura University - Egypt, in 1999. He received his MS in Electrical Communications from Mansoura University - Egypt, in 2002. Dr. Ismail received his MSc and PhD degrees in Computer Engineering from University of Louisiana at Lafayette - USA in 2007 and 2010 and subsequently joined Umm Al-Qura University - Kingdom of Saudi Arabia as an assistant professor. In Fall 2012, he joined University of Bahrain - Kingdom of Bahrain as a Computer Engineering Assistant Professor. In Fall 2016, Dr. Ismail Joined both Electronics & Communications Engineering Department - Mansoura University - Egypt and Zewail City of Science and Technology - University of Science and Technology - Zewail City - Egypt as an assistant professor. Dr. Ismail is currently working as an assistant professor in the Electrical Engineering Department, Southern University and A&M College - Baton Rouge - Louisiana - USA. His area of expertise is Digital Video Processing Algorithms/Architectures levels, Internet of Things (IoT), VLSI and FPGA Design (Low-Power and High-Speed Performance Embedded Systems), automotive transportation, Robotics, RFID, and Wireless and Digital Communication Systems. He has published two books, two book chapters, and more than 35 articles in related journals and conferences. Dr. Ismail served as a reviewer for several conferences and journals, including IEEE ICIP, IEEE GCCCE, IEEE ICECS, IEEE MWSCAS, IEEE ISCAS, IEEE SIPS, IJCDS, Springer, Elsevier, IEEE Transactions on VLSI, IEEE Transaction on Circuit and System for Video Technology (TCSVT), and IEEE Transactions on Image Processing. He severed in the technical committees of IEEE ISCAS 2007, IEEE ICECS 2013, MobiApps 2016, IEEE Virtual World Forum on Internet of Things (WF-IoT 2020), and IEEE MWSCAS 2018, 2019, and 2020 conferences. He was invited to serve as a lead guest editor for a special issue in mobile information systems – Hindawi publishing corporation September 2016. Dr. Ismail served as a PI and Co-PI for several funded grants from NSF and other international fund agencies. Additionally, Dr. Ismail served as a member of many Editorial Boards 2018-present.