Hard Hat Detection Using YOLOv5

Ashish Kumar Singh(2019BCS-010)*, Gurjot Singh(2019BCS-021)*, Vibhor Sharma(2019BCS-070)*
Dept. of Computer Science and Engineering
Atal Bihari Vajpayee Indian Institute of Information Technology and Management
Gwalior, India
aks15o8o2001@gmail.com, gurjotsingh21003@gmail.com, vibhors352@gmail.com

Abstract—**In the United States in 2019, there were 1061 fatalities and many more injuries at construction sites. This was a 5.3 % increase from 2018. The same number in India was around 13000 fatalities. A significant number of people died as a result of falls, slips, and trips, as well as being struck by falling objects which are often caused due to ignorance towards wearing safety equipment. The Occupational Safety and Health Administration (OSHA) delivers Fall Prevention and OSHA-10 training to construction employees in order to reduce fatalities at construction sites due to these types of accidents. Furthermore, safety professionals keep an eye on whether workers are appropriately using personal protective equipment (PPE). Construction deaths have reduced by 2% annually since 1994, according to data; yet, the owners are still dissatisfied with this result. According to many researches, falls are the leading cause of construction fatalities. According to one study, half of the fatalities from falls were caused by workers who either did not use PPEs or did not utilize them effectively. Furthermore, studies have shown that correct usage of hard helmets can prevent fatalities from falls, slips, stumbles, and being struck by falling objects. This research created and evaluated a hard-hat detection programme that employs image processing techniques and deep learning to determine whether or not employees are wearing safety helmets. This research should help authorities to monitor the safety of workers at construction sites by automating the process that will reduce the number of accidents**

Index Terms—**PPE,hard-hat detection,image processing,deep learning.**

I. Introduction

Construction is one of the most dangerous industries where day to day work requires workers to perform hazardous tasks which pose injuries and fatalities to the workers. A significant portion of these fatalities are on account of incidents such as falls, slips, and trips, as well as being struck by falling objects. In years around 2012-13 nearly 34% of total deaths in the US in construction were due to falls as per Occupational Safety and Health Administration (OSHA); this number was as big as 50% in 1980-90s.
Workers fall from great heights and smash their heads on hard floors in the majority of fall accidents. According to an investigation report, half of the fall accidents occurred at a height of less than 3 m; additionally, 57 percent of the fall incidents occurred on ladders, rooftops, construction sites, platforms, or scaffolding. Construction accidents vary in severity, and even a ”minor” mishap can have long-term consequences. The impact of traumatic brain injuries on the skull can be categorized. Even if there is no fracture, a TBI can result in internal bleeding, degenerative brain disorders, and death. Another consideration, especially in the case of falls, is the sort of impact that produced the injury. Straight-on collisions with the head cause linear acceleration (motion in a straight line) and are linked to conditions including skull fractures. Rotational acceleration is caused by impacts at an angle, such as when a person lands on the side of their head. Hard helmets are made to withstand shock, item penetration, and contact with electrical hazards. Half of all fatalities from falls and a considerable number of fatalities from slips, trips, and being struck by falling objects may be reduced if employees wore hard hats appropriately. According to the findings of one investigation into the frequency of construction fatalities and the use of personal protective equipment (PPEs), 47.3 percent of fatally injured victims either did not use PPEs or did not use them appropriately.
Therefore a need arises to provide innovation and automation into the safety system. The safety requirements surrounding hard hats may be enforced more effectively if the safety engineers could watch the workers in real time using video streaming from the job site. This could lower the number of accidents. This research created a technique for identifying workers who do not wear safety helmets on construction sites.

II. Literature Review

A. Convolutional Neural Network

A CNN [1], or convolutional neural network, or convnets, is a deep learning neural network designed to analyze structured arrays of data. CNNs are good at detecting features such as lines, gradients, circles, and even eyes and faces in input image. This makes them pretty useful in computer vision tasks. The main advantage of CNNs over Regular Neural Networks is that they detect important features in image without any human intervention. Also the number of parameters that are learned during the training process in CNNs are comparatively less compared to regular NN. Regular NN in spite of being computationally more expensive than CNNs are not efficient for computer vision tasks.
A CNN consists of mainly 4 types of layers: **Input layer**, **Convolution layer**, **Activation layer** and **Pooling layer**.

1) Input layer: This layer contains the image as raw input.

2) Convolution layer: This layer gives the output by calculating the dot product between the convolution filter and image patches. The dimensions of the output layer are reliant upon the height and width of the input, height, width and depth of the convolution layer, padding and stride length.

3) Activation layer: This layer applies an activation function to every element of the output of the Convolution layer. Some examples of activation functions include Sigmoid Activation function, ReLu Activation function and tanh.

4) Pooling layer: This layer is used to reduce the volume in a CNN which helps reduce memory usage and makes the computation fast. There are usually two types of pooling layers: Max Pool and Average Pool.
The dimensions of the output after applying pooling depends upon the height and width of the input, height, width and depth of the pooling layer, padding and stride length.
Max Pool takes the maximum of all the elements over an image patch for a given pool layer size.
Average Pool takes the average of all the elements over an image patch for a given pool layer size.

These layers can be made to pile up on top of each other in different combinations to form different Convolution Neural Network Architectures(for example, ResNets [2], Inception Network [3]).

B. *EfficientDet*

EfficientDet architecture [4] is an object detecting algorithm first written by Google Brain. EfficientDet is based on EfficientNet, a convolutional neural network for classification that has been pre-trained on the ImageNet picture database.
EfficientDet pools and mixes different granularities of the image to create features that are then transmitted through a NAS-FPN feature fusion layer. The NAS-FPN combines a variety of features at various granularities and sends them to the detection head, which predicts bounding boxes and class labels.The paper [4] carefully explores the tradeoffs in scaling and object detection model. A comparison is shown in 3.

C. *SSD (Single Shot Detector)*

SSD is another popular algorithm used in object detection. It provides high speed as well as accuracy using relatively low resolution images. High detection accuracy is accomplished by utilizing multiple boxes or filters of various sizes and aspect ratios.
Wei Liu [5] presents this approach; to achieve high detection accuracy, prediction for the bounding boxes for different objects in the image is done by multiple feature maps of different sizes that represent multiple scales rather than a single feature map. For feature extraction VGG-16 base network is used as it is standard CNN architecture for image classification. To this base network additional convolutional layers are added for detection. The size of the layers decreases on moving towards the end of the network.
As shown in 1 in the input image a bounding box is formed for each object present in that image. These boxes of different aspect ratio are evaluated by convolutional layers. This is done to find the default box that overlaps with the bounding box containing objects. Two default boxes are shown below 2.
These boxes that are containing objects are considered as positive boxes and the rest are ignored. Δcx, Δcy, h and w, represent the offsets from the center of the default box and its height and width.

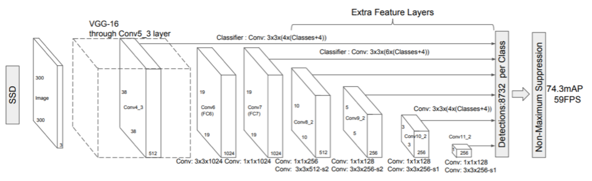
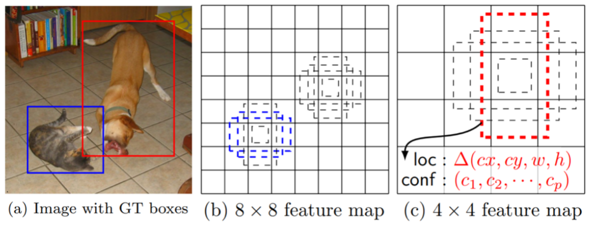


Fig. 1. SSD Architecture



loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$
(a) Image with GT boxes
(b) 8×8 feature map
(c) 4×4 feature map

Fig. 2. SSD Bounding Box

D. *YOLO*

YOLO [6] stands for **You Only Look Once**. YOLO models are well known for giving high performance while also requiring less space for its operation, making them perfect candidates for real time object detection scenarios. This method processes the entire image with a single neural network, then divides it into parts and predicts bounding boxes and probabilities for each component. The predicted probability is used to weigh these bounding boxes. The predictions are made after only one forward run through the neural network. After non-max suppression, it provides discovered items.

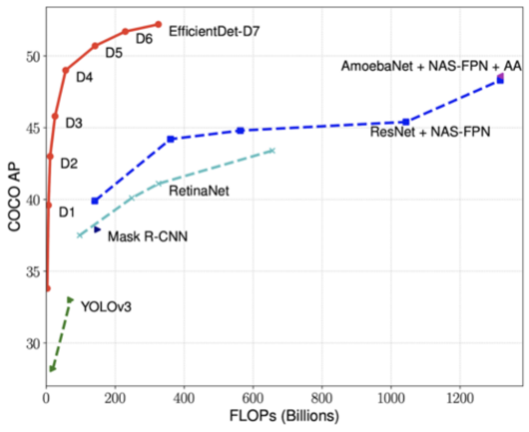


Fig. 3. Comparison between different object detection algorithms

III. Summary

YOLO, EfficientDet and Single Shot Detection are pioneer steps in the field of Object Detection. They all provide with real time object detection with newer versions of these techniques providing better Speed vs Accuracy tradeoff than the older versions.
However, these techniques are incomplete in addressing the camouflage problem. They also fail to separately segregate objects which are visually similar and are closer to each other. Also, these techniques do not yield the same results for datasets containing images taken from irregular angles and having low lighting.

IV. Methodology

We have used the YOLOv5(small) model architecture for training our safety helmet detector. The **Model Architecture** has been described in A, B and C subsections.

A. Backbone

In convolutional neural networks, especially with respect to the object detection tasks, backbones refers to the part of the network that is responsible for feature extraction. It is a part of the network that sees the input. Feature extraction is used to reduce a variable sized image to a fixed set of visual features.

1) CSPDarknet53: 5 shows CSPDarknet53 architecture. This has been used for feature extraction. In this section the working of CSPDarknet53 is explored.
Conv n*f*f, stride s, is nothing but a Convolution layer with filter size as f, number of filters n and stride length s.
CSPBlock X num in CSPDarknet53 represents that num CSPBlocks are stacked in the CSPDarknet53 one after each other.
In the CSPBlock where the Base Layer(height*width*channels) gets separated into Part 1 and Part 2, where the data in the base layer is divided into two equal parts, one part going to Part 1(height*width*(channels/2)) and the other part going to Part 2(height*width*(channels/2)).
Add in CSPBlock is used to add two different inputs of the same dimensions element by element.
Mish is an activation function used in Convolution Neural Networks.

$$f(x) = x.tanh(softplus(x)) \tag{1}$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2}$$

$$softplus(x) = ln(1 + e^x) \tag{3}$$

$$\begin{aligned} tanh(softplus(x)) &= \frac{e^{softplus(x)} - e^{-softplus(x)}}{e^{softplus(x)} + e^{-softplus(x)}} \\ &= \frac{e^{ln(1+e^x)} - e^{-ln(1+e^x)}}{e^{ln(1+e^x)} + e^{-ln(1+e^x)}} \end{aligned} \tag{4}$$

$$\begin{aligned} tanh(softplus(x)) &= \frac{1 + e^x - \frac{1}{1+e^x}}{1 + e^x + \frac{1}{1+e^x}} \\ &= \frac{(1 + e^x)^2 - 1}{(1 + e^x)^2 + 1} \end{aligned} \tag{5}$$

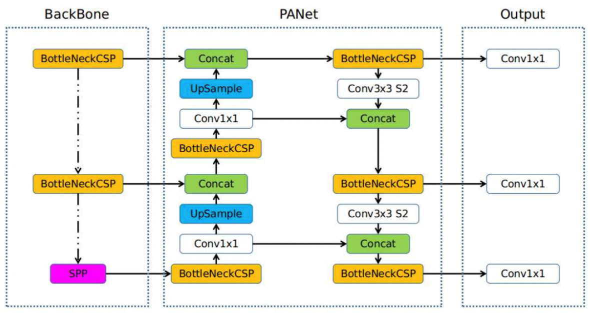


Fig. 4. Overview of YOLOv5 Model Architecture

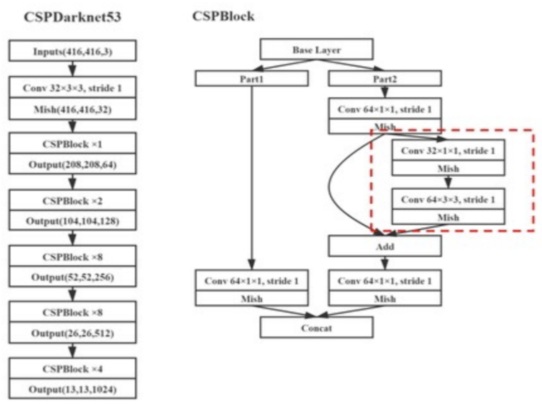


Fig. 5. Overview of CSPDarknet53 Architecture

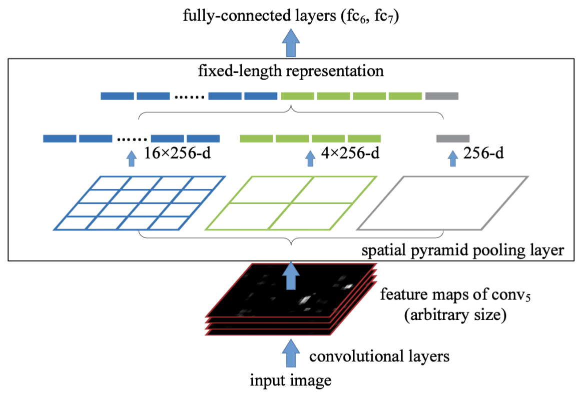
$$f(x) = x.\frac{(1 + e^x)^2 - 1}{(1 + e^x)^2 + 1} \tag{6}$$

Concat is used to stack two inputs of the same dimensions one over another.

B. Neck

Neck is used for creating a pyramid feature in the input image. It helps in determining the scaling factor of observed items of similar nature but different scales.

1) Spatial Pyramid Pooling(SPP): We have the features map at the output of the convolution neural networks, which are features generated by our various filters. To put it another way, we can have a filter that can identify circular geometric patterns and build a feature map that highlights these shapes while retaining the shape's position in the image. Whatever the size of our feature maps, the Spatial Pyramid Pooling Layer allows us to generate fixed size features. It uses pooling layers, such as Max Pooling, to produce different representations of our feature maps in order to generate a fixed size. Each feature map is pooled to become one value. This gives us a vector of size 1256.After that each feature map is pooled to have 4 and 16 values to give us vector of sizes 4256 and 16256 respectively.The 3 vectors obtained above are then concatenated to form a fixed size vector which will be the input of fully connected network.



fully-connected layers (fc6, fc7)
fixed-length representation
16×256-d
4×256-d
256-d
spatial pyramid pooling layer
feature maps of conv5 (arbitrary size)
convolutional layers
input image

Fig. 6. Overview of SPP

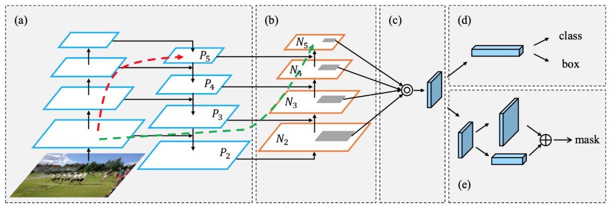


Fig. 7. Overview of PaNet

2) PANet: PANet stands for Path Aggregation Network.It is an architecture that allows better propagation of layer information from bottom to top or top to bottom.The neck's components normally flow up and down between layers, connecting just the final few layers of the convolutional network. In the figure 7 the first layer's information is added to layer P5 (red arrow) and propagated to layer N5 (green arrow). This is a shortcut for getting low-level data to the top. In original PaNet, the current layer and information from a previous layer is added together to form a new vector.In our Yolo implementation to add information to top layers modified version of PaNet is used where the new vector is created by concatenating the input and the vector from a previous layer.

C. Head

Head is the top of a neural network. It is the layer in the neural network that is responsible for generating the final output.

1) YOLOv3 Head: YOLOv3 has a multi-head detection system. At 3 different particular instances in our neural network a prediction is made..

D. Loss Function

$$Loss = \lambda_1 L_{loc} + \lambda_2 L_{cls} + \lambda_3 L_{obj}$$

The YOLOv5 loss consists of 3 parts:

1) ***Location Loss*** (L_{loc}) : **IOU** (Intersection over union) as shown in 10 is defined as:

$$IoU = \frac{AreaofIntersection}{AreaofUnion}$$

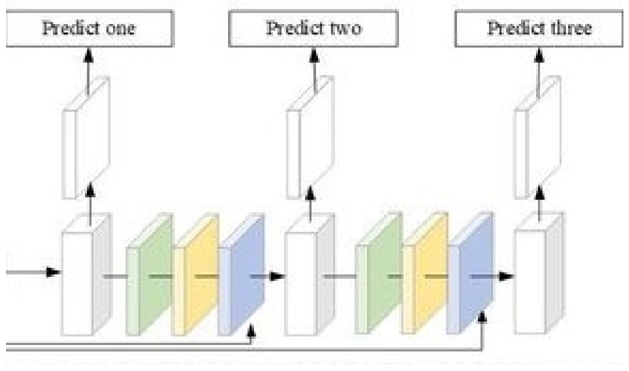


Fig. 8. YOLOv3 head

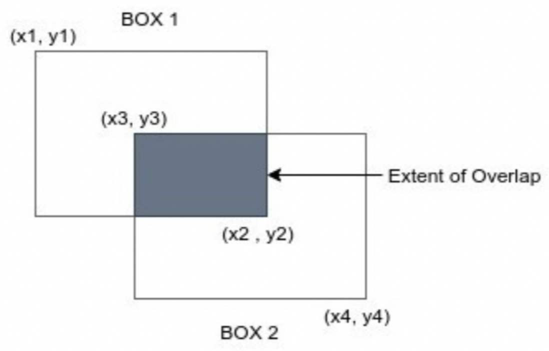


Fig. 9. IoU

For this particular scenario, BOX 1 and BOX 2 can be considered as predicted Bounding Box and ground truth Bounding Box.

$$L_{IoU} = IoU$$

GIoU(Generalised IoU Loss) maximizes the ground truth and anticipated bounding box overlap area. For non-overlapping scenarios, it increases the predicted box's size to overlap with the target box by moving gradually towards the target box.

$$L_{GIoU} = 1 - IoU + \frac{|C - AreaofUnion|}{|C|}$$

where C is the area of the smallest box covering the predicted and ground truth bounding boxes.

DIoU(Distance IoU Loss)

$$L_{DIoU} = 1 - IoU + \frac{d^2}{C^2}$$

here d is the distance between the center points of ground truth bounding box and predicted bounding box and C is the diagonal length of the smallest enclosing box covering both the boxes.

CIoU Complete IoU takes three crucial geometric factors, namely overlap area, center point distance, and aspect ratio,

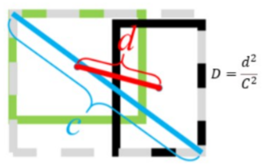


Fig. 10. DIoU

should be considered when calculating a good loss for bounding box regression. CIoU Loss is calculated by adding all three:

$$L_{CIoU} = 1 - IoU + \frac{d^2}{c^2} + \alpha\nu$$

here d is the distance between the center points of ground truth bounding box and predicted bounding box and C is the diagonal length of the smallest enclosing box covering both the boxes as mentioned in DIoU above.

$$\nu = \frac{4}{\pi^2}(arctan(\frac{w^{gt}}{h^{gt}}) - arctan(\frac{w}{h}))^2$$

where w^{gt} and h^{gt} is the width and height of ground truth bounding box and w and h is the width and height of the predicted bounding box.

$$\alpha = \frac{\nu}{1 - IoU + \nu}$$

$L_{loc} = L_{CIoU}$

*2) **Classes Loss (L_{cls})** :*

$$L_{cls} = \sum_{i=0}^{S^2} \chi_i \sum_{c \epsilon classes} (BCE(p_i(c), p_i c))$$

$$BCE(x, y) = -xlog(y) - (1 - x)log(1 - y)$$

here S^2 represents the total number of grid cells our input image is divided into. χ_i is 1 if an object appears in cell i otherwise 0.
$p_i(c)$ denotes the conditional class probability for class c in cell i.

$$ccp \equiv P_r(Class_i|object)$$

$P_r(Class_i|object)$ is the probability an object belongs to class i given an object is present and ccp is the conditional class probability.

*3) **Objectness Loss (L_{obj})** :*

$$L_{obj} = \sum_{i=0}^{S^2}\sum_{j=0}^{B}(\chi_{ij}^{obj} * BCE(C_i, C_i')) +$$
$$\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}(\chi_{ij}^{noobj} * BCE(C_i, C_i') * (1 - \chi_{ij}^{obj}))$$

where

$$BCE(x, y) = -xlog(y) - (1 - x)log(1 - y)$$

here S^2 represents the total number of grid cells our input image is divided into and B denotes the total number of bounding boxes.
χ_{ij}^{obj} is 1 if jth boundary box in cell i has an object else 0.
C_i denotes the box confidence score of box j in cell i.

$$bc \equiv P_r(object) \cdot IoU$$

$P_r(object)$ is the probability the box contains an object and bc is the box confidence score.

E. Training

We performed training for 100 epochs on a GPU Hardware Accelerator and took batch size as 100.For every epoch we keep track of precision, recall, mAP@0.5 and mAP@0.5:0.95 on our training dataset.
True Positive is when the model predicts the positive class correctly.
False Positive is when the model predicts the positive class incorrectly.
False Negative is when the model predicts the negative class incorrectly.
True Negative is when the model predicts the negative class correctly.
Precision is a measure of how accurate our predictions are and **Recall** is a measure of how well we find out all the positives.

$$Precision = \frac{TP}{TP + FP} \quad (7)$$

$$Recall = \frac{TP}{TP + FN} \quad (8)$$

True Positives, False Positives, True Negatives and False Negatives are calculated using IoU for a given IoU threshold value. This means that for every IoU threshold value we will have different values for Precision and Recall.
AP(Average Precision) for a class is calculated by finding out the area under the precision recall curve for a given class label.
mAP(mean Average Precision) is calculated by finding out the average of AP over all the classes.

$$mAP = \sum_{c \epsilon classes} \frac{(AP_c)}{totalnumberofclasses} \quad (9)$$

mAP@0.5 means mAP calculated for an IoU threshold value of 0.5.
mAP@0.5:0.95 means average mAP calculated over different IoU threshold values(0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95).

V. Experimental Section

A. Dataset

We took 4750 images of workers in a construction site along with their annotation in Pascal VOC Format. 70% of those images were used for training, 20% for validation and 10% for testing. We also performed a preprocessing step by stretching

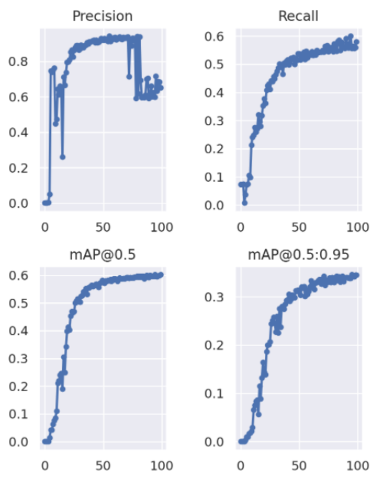


Fig. 11. Precision, Recall, map@0.5, map@0.5:0.95 values on our training dataset from 0 to 100 epochs

our image to a specific dimensions(416*416 in our case) so as to make sure that all the input images in our input data are of the same dimensions for our model.

B. Results

Before we started training our model mAP@0.5 value on our training dataset was 0.000417. After training for mAP@0.5 score on training dataset improved to 0.667. Our validation and training dataset gave a mAP@0.5 score of 0.6 and 0.602 respectively. Training our entire model took 2.319 hours.

Sets	Precision	Recall	mAP@0.5	mAP@0.5:0.95
train	0.661	0.658	0.667	0.382
val	0.587	0.568	0.6	0.345
test	0.932	0.557	0.602	0.353

TABLE I
PRECISION, RECALL, MAP@0.5, MAP@0.5:0.95 VALUES ON OUR TRAINING, VALIDATION AND TEST DATASET FOR OUR TRAINED MODEL.

VI. CONCLUSION

YOLOv5 even though hasn't been officially published like previous versions of YOLO can be used for our object detection tasks and can be fine tuned to our specific application by training its weights with images of our use case instances. The model could be trained further on a larger dataset and for more epochs to give better results in real world scenarios. There is also a possibility for exploration on whether doing so(training our model for more epochs and on larger and diverse dataset) would address the problems which have been left unanswered by previous Object Detection Techniques with respect to our task.

REFERENCES

[1] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[4] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2019. [Online]. Available: https://arxiv.org/abs/1911.09070

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37.

[6] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.