International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 06 | June 2021 www.irjet.net p-ISSN: 2395-0072

Safety Gear Equipment Detection for Warehouse and Construction Sites Using YOLOv5

Shubh Mody[1], Hriday Mehta[2], Pragun Mantri[3], Bushra Ali[4], Ankit Khivasara[5]

[1,2,3,4]*Department of Electronics And Telecommunication Engineering, K. J. Somaiya College Of Engineering, Mumbai - 400077, Maharashtra, India*

[5]*Assistant Professor, Department of Electronics and Telecommunication Engineering, K. J. Somaiya College Of Engineering, Mumbai - 400077, Maharashtra, India*

Abstract - *With the advancements in technology made in the 21st century, object detection is a topic that has attracted a lot of attention in recent years. It is probably the most well-known term within the domain of computer vision and it encounters some really interesting problems. To gain a complete image understanding, we should not only concentrate on classifying different images but also try to precisely estimate the concepts and locations of objects contained in each image. This task is referred to as object detection. Object recognition is a collection of related tasks for identifying objects in digital photographs. Here, to perform object detection of different safety equipment, we have utilized a predefined framework/pretrained model as they are convenient for serving our purpose. The aforementioned framework we have used for object detection is – "YOLO" (You Only Look Once). It's a supremely fast and accurate framework that can process at 45 frames per second (in real-time) and also understands generalized object representation. We have initially given an image as input to YOLO and the framework then divides the input image into grids. Image classification and localization are applied on each grid. We have created a website for the deployment of our model using "Flask". It is a web application framework to compile modules and libraries which will also help the developer to write web applications without writing low-level code like thread management and protocols. We have also used Heroku as a platform to deploy and manage our web application. Heroku is a container-based cloud Platform as a Service (PaaS) that can be used to instantly extend applications with fully managed services. After the model deployment is done, we can ask for the users to upload images of their choice through the website. YOLO then predicts the bounding boxes and their corresponding class probabilities for objects (if any are found). The classes we have categorized the images into are listed as "hat", "vest", "goggles", "glove", "shoe".*

Key Words: Object Detection, Deep Learning, Convolution, Neural Networks, Flask, Heroku, and YOLO.

1. INTRODUCTION

In today's ever-growing world, technology is advancing at a phenomenal pace and it's showing no sign of slowing down. Over the years, technology has revolutionized our world and daily lives. The process of recognizing objects in videos and images is known as Object recognition. A human glancing at a picture will instantly acknowledge what the objects are and where they're located within the image. The ability to notice objects quickly combined with the knowledge of a person helps to create an accurate judgment concerning the nature of the item. A system that simulates the flexibility of the human sensory system to notice objects is something that scientists are researching. Fast and accurate are the two prerequisites for which an object detection algorithm is examined. Deep learning is a subfield of machine learning. We study a hierarchy of features and functions of the subject under study with the help of input data. Deep learning is similar to machine learning. It builds a hierarchy of features from top to bottom. The systematic and arranged features are easy to understand and explain to others as well. This method of learning can learn the features at any level stigmatically. The help of human-made techniques is not required. The best thing about deep learning is that all the models have deep architectures. Object recognition is to describe a collection of related computer vision tasks that involve activities like identifying objects in digital photographs. Image classification involves activities such as predicting the class of one object in an image. Object localization is referring to identifying the location of one or more objects in an image and drawing an abounding box around their extent. Object detection does work of combines these two tasks and localizes and classifies one or more objects in an image. When a user or practitioner refers to the term "object recognition", they often mean "object detection". It may be challenging for beginners to distinguish between different related computer vision tasks [1].

So, we can distinguish between these three computer vision tasks with this example: As such, we can distinguish between these three computer vision tasks:

Image Classification: Predict the type or class of an object in an image.

- **Input:** An image with a single object, such as a photograph.
- **Output:** A class label (e.g. one or more integers that are mapped to class labels).

Object Localization: Locate the presence of objects in an image and indicate their location with a bounding box.

- **Input**: An image with one or more objects, such as a photograph.

• **Output**: One or more bounding boxes (e.g. defined by a point, width, and height).

Object Detection: Locate the presence of objects with a bounding box and types or classes of the located objects in an image.

• **Input**: An image with one or more objects, such as a photograph.

• **Output**: One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.

From this breakdown, we can see that object recognition refers to a suite of challenging computer vision tasks [1][2].

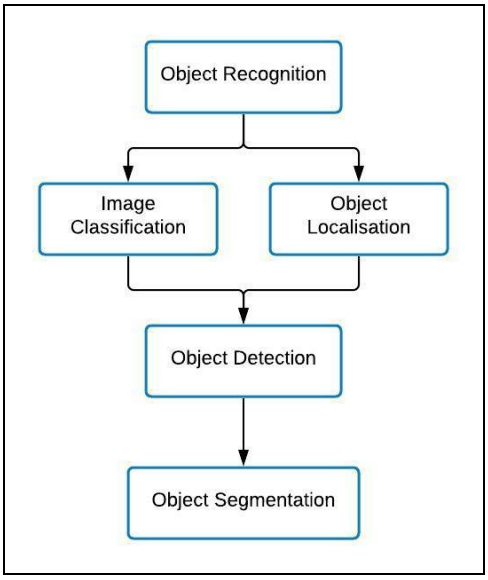


Fig - 1: Overview of Object Recognition Computer Vision Tasks

One further extension to this breakdown of computer vision tasks is *object segmentation*, also called "object instance segmentation" or "semantic segmentation," where instances of recognized objects are indicated by highlighting the specific pixels of the object instead of a coarse bounding box[2][3].

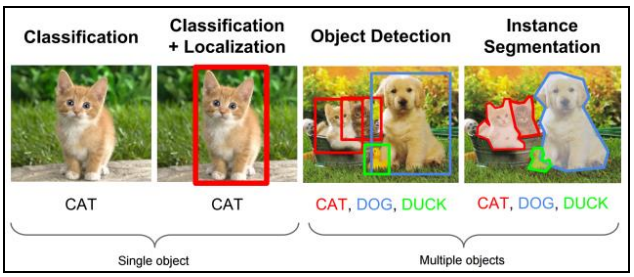


Fig - 2: Image Classification & Localization[3]

Image classification also involves assigning a class label to an image, whereas object localization involves drawing a bounding box around one or more objects in an image. Object detection is always more challenging and combines these two tasks and draws a bounding box around each object of interest in the image and assigns them a class label. Together, all these problems are referred to as object recognition.

Object detection is a computer vision technique that helps to identify and locate objects within an image or a video. In deep learning, object detection algorithms have been explored rapidly. With its increasing utilization, the future is bright in this domain. Some examples of object detection are: For a car to decide what to do next i.e accelerate, apply brakes, or turn, it needs to know where all the objects are around the car and this is achieved by object detection; Facial Recognition: Smartphone getting unlock by detecting the person's face; Identity verification through IRIS code; Ball tracking in sports; Medical imaging; Object counting; Robotics, etc. all use object detection. Object detection eases our work to a large extent and all the examples stated above are proof. In India which is known for poor safety measures for the labor workers and for those who are working in factories, construction sites, warehouses, etc., object detection can be a life savior. To safeguard the health of the workers, we have developed a model which will help us to ensure that the worker has worn the proper safety equipment or not. After the successful training of the model and comparing its various parameters, we have deployed our model for real-world application use with the help of flask we successfully created our WebPage to use that model. And finally, we deployed our webpage on Heroku.

2. LITERATURE REVIEW

Object detection is the identification of an object in an image along with its localization and classification. Software systems that can perform these tasks are called object detectors. Object Detectors have been making fast strides in accuracy, speed, and memory footprint. A decent argument can be made that object detectors have achieved close to human parity. Conversely, like any deep learning model, these detectors are still open to adversarial attacks and can misclassify objects if the image is adversarial. Work is being done to make object detectors and deep learning models, in general, more robust to these attacks. Accuracy, speed, and size will constantly be improved upon, but that is no longer the most pressing goal. Detectors have attained a respectable quality, allowing them to be put into production today. The goal now should be to make these models robust against hacks and ensure that this technology is being used responsibly[4][6][9].

The first Deep Learning object detector model was called the Over feat Network which used Convolutional Neural Networks (CNNs) along with a sliding window approach. It classified each part of the image as an object/non-object and subsequently combined the results to generate the final set of predictions. This method of using CNNs to solve detection led to new networks being introduced which pushed the state of the art even further.

There are currently two methods of constructing object detectors- the single-step approach and the two-step approach. The two-step approach has achieved better accuracy than the former whereas the single-step approach has been faster and shown higher memory efficiency. The single-step approach classifies objects in images along with their locations in a single step. The two-step approach on the other hand divides this process into two steps. The first step generates a set of regions in the image that have a high probability of being an object. The second step then performs the final detection and classification of objects by taking these regions as input. These two steps are named the Region Proposal Step and the Object Detection Step respectively. Alternatively, the single-step approach combines these two steps to directly predict the class probabilities and object locations. Object detector models have gone through various changes throughout the years since 2012. The first breakthrough in object detection was the RCNN which resulted in an improvement of nearly 30% over the previous state-of-the-art.

In 2015, the YOLO algorithm inherits the idea of overfeating, its speed of detection reaches 45 frames per second. P-ReLU activation function is adopted during model training. However, it has problems such as the accuracy of positioning and recall rate are dissatisfied. Besides, the detection for a very small object is ineffective. At last, generalization ability is relatively weak [2][5].

YOLOv2 [5] and YOLOv3 [6] algorithms were proposed on CVPR 2017, focusing on solving the poor recall rate and positioning accuracy. It uses DarkNet-19 as the feature extraction network and adds batch normalization as the pre-treatment. The original YOLO uses the full connection layer to directly predict the coordinates of the bounding box, while YOLOv2 refers to the idea of Faster R-CNN, introduces anchor mechanism, and calculates a better anchor template in the training set.

The operation of anchor boxes is applied to the convolutional layer to improve the prediction of the bounding boxes. Meanwhile, the positioning method with strong constraints is adopted to greatly improve the recall rate of the algorithm. Combined with the fine-grained features, the shallow feature and the deep feature are connected, which is helpful to the detection of small-size objects.

In 2020, YOLOv4 was released [7][4], which is a significant update to the YOLO family, with an increase in AP and FPS of 10% and 12% based on COCO datasets, respectively. On June 25, 2020, Ultralytics released YOLOv5 on Github, which performs excellently, especially the frame rate 140FPS of the YOLO v5s model is amazing.

In YOLOv5, four versions of the object detection network are given, namely, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. We have used all models in this project. YOLOv5s is the network with the minimum depth and the minimum width in the YOLOv5 series. The YOLOv5s is the fastest one, but the AP is relatively less accurate. However, this model is also a good choice if the focus of object detection is on larger targets and fewer complex scenarios, which are speed-oriented. The other three networks of the YOLOv5 series are based on YOLOv5s to continuously deepen and widen the network, the AP is also continuously improved, but the speed will become slower[12]. We have shown the results of all the models in this project but have used YOLOv5s while deploying it into the webpage because it is smaller in size. We will discuss all the models of YOLOv5 and the architecture of YOLO in the next section.

We also did a study about the plight of construction workers. In India, the construction industry is the second largest employer when compared to agriculture. Throughout the world, the construction area of civil engineering is one of the most hazardous industries. The number of fatal accidents taking place at the construction sites is quite alarming and the major cause was found to be falls of persons from height and through openings.

In the present scenario, the Indian construction industry is quite large and complex involving the latest technology as well as manpower. On a par with the development of the construction industry, drawbacks in terms of safety and health aspects are also witnessed.

In the past few decades, the need for safety awareness among construction industries was realized. This is due to the high cost associated with work-related injuries, worker's compensation, insurance premium, indirect costs of injuries, and litigation. Every year, a considerable amount of time is lost due to work-related health issues and site accidents [8]. There are several factors responsible for health problems and construction site accidents. From the result of Occupational Safety and Health Administration examination on the causes of construction fatalities, it was shown that 39.9% of fatalities in construction were caused by falls, 8.4% were struck by objects, 1.4% were caught in between incidents, and 8.5% were electrocution [8]. Thus, we decide to take the dataset which can help us to increase the measures for the safety of workers.

3. WORKING AND ARCHITECTURE OF YOLO MODEL

The YOLO model was first described by Joseph Redmon, et al. in the 2015 paper titled “You Only Look Once: Unified, Real-Time Object Detection.” Note that Ross Girshick, developer of R-CNN, was also an author and contributor to this work, then at Facebook AI Research. The approach involves a single neural network trained end to end that takes a photograph as input and predicts bounding boxes and class labels for each bounding box directly. The technique offers lower predictive accuracy (e.g. more localization errors), although operates at 45 frames per

second and up to 155 frames per second for a speed-optimized version of the model.

The model works by first splitting the input image into a grid of cells, where each cell is responsible for predicting a bounding box if the center of a bounding box falls within the cell. Each grid cell predicts a bounding box involving the x, y coordinate, and the width and height, and the confidence. A class prediction is also based on each cell.[2][5][6].

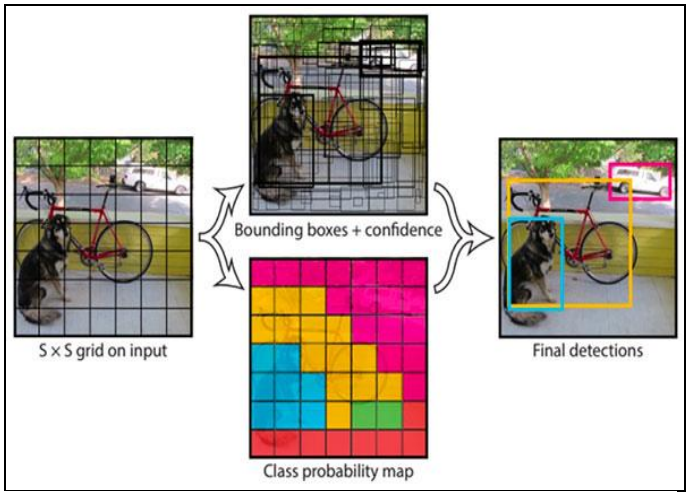


Fig - 3: YOLO model with 7x7 grid cell was applied on the input image. (Redmon, et al., 2016)

In order to implement object detection, each grid cell predicts *B* bounding boxes with their parameters and confidence scores for those boxes (*Figure 3.7*) (V Thatte, 2020). These confidence scores reflect the presence or absence of an object in the bounding box. The confidence score is defined as:

$$confidence\ score = p(Object) * IOU_{pred}^{truth}$$

with p(object) is the probability that there is an object inside the cell and IOU_{pred}^{truth} is intersection over union of prediction box and ground truth box. p(object) is in the range 0-1, so the confidence score is close to 0 if no object exists in that cell. Otherwise, the score equal to IOU_{pred}^{truth}.

Besides, each bounding box consists of 4 other parameters (x,y,w,h) corresponding to (center coordinate(x,y), width, height) of a bounding box fig - 3. Combining with the confidence score, each bounding box consists of 5 parameters.

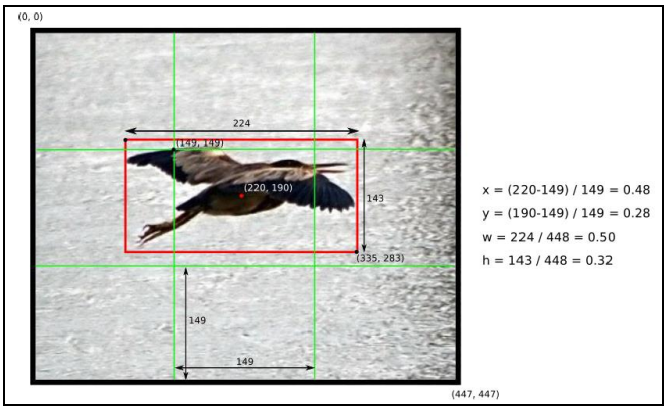


Fig - 4: Parameters for a bounding box in a 3x3 grid cell. (Menegaz, 2018)

The probability of an object predicted for each class in a grid cell denoted p(class$_i$ |Object) Probability values for the C class will produce C output for each grid cell. The B bounding box of the same grid cell shares a common set of predictions about the object's class, meaning that all bounding boxes in the same grid cell have the same class. As shown in Fig- 4, for example, there are 2 prediction boxes in the center cell. They have different parameters (x,y,w,h, confidence score). However, they have the same 3 prediction classes.

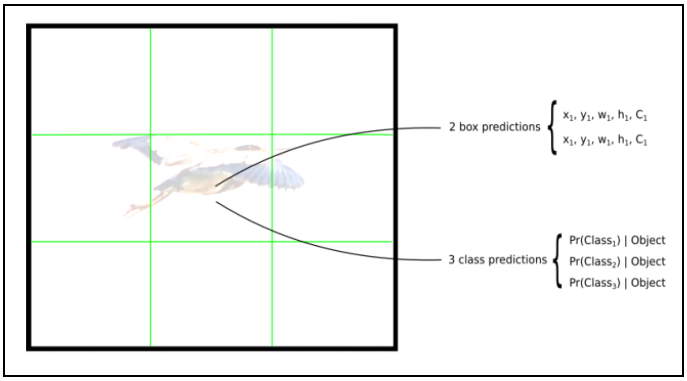


Fig - 5: The bounding boxes of the same grid cell share the set of prediction classes. (Menegaz, 2018)

Thus, the model has S x S grid cells for an image. Each cell predicts B bounding boxes which consist of 5 parameters and share prediction probabilities of C classes. The total YOLO output of model parameters will be S×S×(5∗B+C) (Menegaz, 2018). For example, evaluating the YOLO model on the famous COCO dataset which contains 80 classes and set each cell to predict 2 bounding boxes, the total output parameters are 7×7×(5∗2+80) [2][5][6][9].

The purpose of the YOLO algorithm is to detect an object by precisely predicting the bounding box containing that object and localize the object based on the bounding box coordinates. Therefore, predicted bounding box vectors correspond to output vector y^ and ground truth bounding box vectors correspond to vector label y. Vector label y and predicted vector y^ can be indicated as fig - 5 where the

purple cell does not have any object, the confidence score of bounding boxes in purple cell equal to 0, then all remain parameters will be ignored.

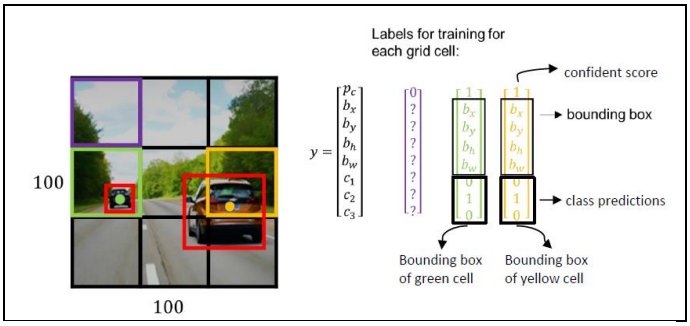


Fig - 6: Specifying label vector *y* in a YOLO model has 3x3 grid cells and predict object for 3 classes. (datahacker.rs, 2018)

Finally, YOLO applies Non-Maximum Suppression (NMS) to clean all bounding boxes which do not contain any object or contain the same object as other bounding boxes (*Fig- 6*). By picking a threshold value, NMS removes all the overlap bounding boxes which have intersection over union (IOU) values higher than the threshold value. [10](ODSC Science, 2018)

3.1 STRUCTURAL ANALYSIS OF YOLO MODEL

In YOLOv5, four versions of the object detection network are given, namely, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. YOLOv5s model was used in this project. YOLOv5s is the network with the minimum depth and the minimum width in the YOLOv5 series. The YOLOv5s is the fastest one, but the AP is relatively less accurate. However, this model is also a good choice if the focus of object detection is on larger targets and less complex scenarios, which are speed-oriented. Here for deployment purposes, we have used YOLOv5s. The other three networks of the YOLOv5 series are based on YOLOv5s to continuously deepen and widen the network, the AP is also continuously improved, but the speed will become slower[12].

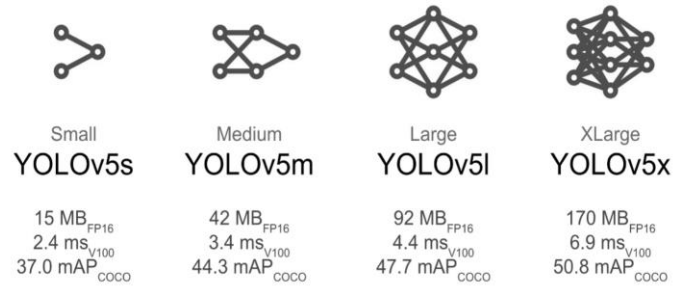


Fig – 7: Different models of YOLOv5[5]

The structure of YOLOv5 is very similar to that of YOLOv4, both use CSPDarkNet53 (i.e., Cross Stage Partial Network) as the Backbone net. PANET (i.e., Path Aggregation Network) and SPP (i.e., Space Pyramid Pooling) were employed as the Neck net.

The input of YOLOv5 adopts the same way of mosaic data enhancement as YOLOv4. Mosaic refers to the method of CutMix data enhancement [5]. However, CutMix only makes use of two images for splicing, while mosaic data enhancement utilizes four images, which are randomly scaled, cropped, and resized. In addition, YOLOv5 is also optimized in terms of adaptive image scaling.

The function of the backbone is to aggregate and combine different fine-grained images to form a convolutional neural network. The backbone of YOLOv5s adopts CSPNet, which makes use of gradient information in other CNN frameworks [5][13].

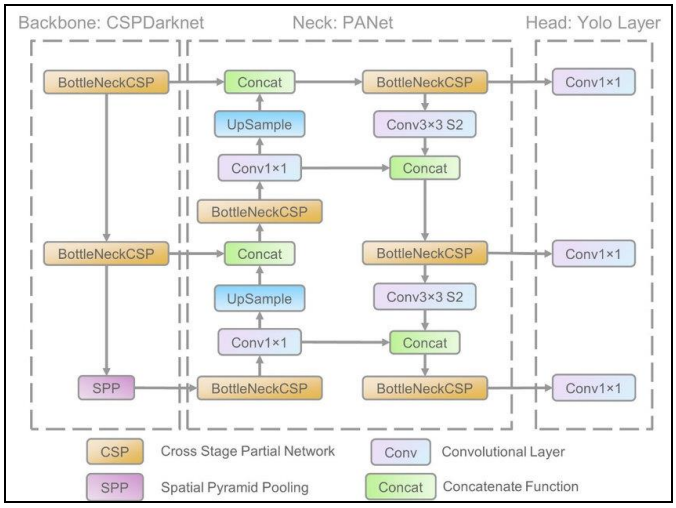


Fig – 8: Structural Architecture of YOLOv5[17]

Neck is a series of network layers that mix and combine image features and transfer them to the prediction layer, also known as Head. The Neck part of YOLOv5s adopts PANET [5]. Generating feature pyramids is the main function of the Neck. The feature pyramid enhances the detection of visual objects with multiple scales; thus, it recognizes the same object with various sizes and scales.

The Head of the model is mainly employed for final judgment. In the YOLOv5 model, the model Head is as same as the previous versions of YOLOv3 and YOLOv4. Each Head has (80 classes + 1 probability + 4 coordinates) × 3 anchor frames, 255 channels in total[6][7].

The loss function for object detection is generally composed of classification loss and recurrence loss of bounding boxes. GIOU is used as the loss function of bounding boxes in YOLOv5. In the postprocessing of object detection, NMS operation is usually needed for filtering visual objects. In YOLOv5, weighted NMS is accommodated. The selection of activation function is very important for deep neural networks. In YOLOv5, Leaky ReLU is used in the

middle/hidden layer, the sigmoid function is applied to the final detection layer[5]. YOLOv5 provided us with two optimization functions Adam and SGD. The default one is SGD. If we train a smaller dataset, Adam is a better choice.

4. PROJECT DESIGN

Object detection, in simple terms, is a method that is used to recognize and detect different objects present in an image or video and label them to classify these objects. It typically uses different algorithms to perform this recognition and localization of objects, and these algorithms utilize deep learning to generate meaningful results. The theory behind this technique of object detection is that it helps in the recognition, detection, and localization of multiple visual instances of objects in an image or a video. It provides a much better understanding of the object as a whole, rather than just basic object classification. Hence, this proves to be very advantageous when it comes down to increasing the safety of workers at construction sites. This method can be used to count the number of instances of unique objects and mark their precise locations, along with labeling. With time, the performance of this process has also improved significantly, helping us with real-time use cases as mentioned above. Hence, we can easily identify the different equipment responsible for the protection of workers. There are several ML and DL methods available for object detection, in this project we have adopted versions of v5 belonging to the YOLO family. The YOLO framework focuses on the entire image as a whole and predicts the bounding boxes, then calculates its class probabilities to label the boxes. The family of YOLO frameworks is very fast object detectors. The better the quality of training data we input, the more accurate and efficient the image recognition model is. The most important parameters while training a neural network model include size, quality, and quantity of images along with the number of channels. Aspect ratio, mean and standard deviation of images is also considered [6]. We also look at image scaling and the available data variations. Here we took our custom dataset which includes almost 3000 pictures of construction and factory workers and labeled it using labellmg. The classes we have categorized the images into are listed as “hat”, “vest”, “goggles”, “glove”, “shoe”. It is highly detrimental to those working at construction sites who risk their lives on a daily basis. Training such a model would be helpful to minimize the risk and guarantee maximum safety. Out project involved detecting objects of a custom dataset as well. One could also upload an image of his/her choice on our webpage and the output would result in that image being categorized as one of the classes of gear that workers wear. All things considered, it eventually provides us with an answer to the question: “What object is where and how much of it is there?”.

4.1 PROBLEM STATEMENT

The task is to detect the safety equipment of construction personnel. Humans can glance at an image and instantly know what objects are present but this is complicated for computers. This model is crucial as safety is dependent on this, and all the errors made by detection will propagate to the model. Following are some of the major challenges with detection in such conditions. Not only do we want to classify image objects but also to determine the objects’ positions at fast speeds. Limited data, class imbalance, and multiple spatial scales can also cause difficulties and may act as limitations restricting our model. We also need to ensure smooth deployment of our model on the webpage with the help of the Heroku platform.

4.2 BLOCK DIAGRAM

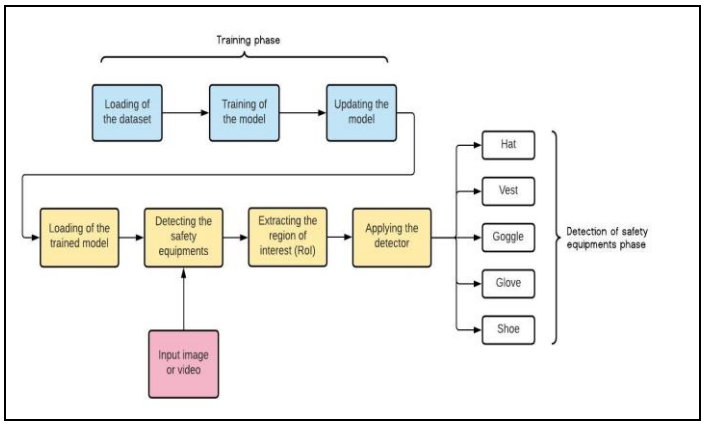


Fig – 9: Flow Chart of Project

4.3 OBJECTIVES

1. To learn about the different models of YOLOv5.
2. To label and train our custom dataset.
3. Analyze the results from the trained model.
4. Making a webpage using flask and deploying the model on it.
5. Finally, deploying our webpage with the model on Heroku and using it to detect safety equipment.

This project will be helpful to understand the YOLOv5 model in deep and its different types. We are also aiming of creating a model which we trained using a custom dataset. The dataset is based is on the safety equipment of factory, construction workers. This will help us to safeguard the safety of the workers by properly monitoring them. It can be used by both private and government organizations. We also deployed our model on a webpage which we have built using flask and atlas we deployed our webpage on Heroku.

5. IMPLEMENTATION

5.1 PREPARING THE DATASET AND LABELLING IT

We downloaded approximately 3000 images of a construction worker with safety equipment from dreamstime.com and online resources labeled it using labellmg. We have to select YOLO in the application and after clicking save a txt file is created for the same.

Fig – 10: Labelling the picture with labelImg

```
0 0.591875 0.147500 0.176250 0.158333
2 0.595625 0.290000 0.148750 0.090000
1 0.590625 0.734167 0.323750 0.531667
```

Fig - 11: The .txt file created after saving the above image

Open the label file to understand what labeling is. A sample file can look like Figure - 11. The format of the label file is

Class_ID_1 X_CENTER_NORM Y_CENTER_NORM WIDTH_NORM HEIGHT_NORM

Class_ID_2 X_CENTER_NORM Y_CENTER_NORM WIDTH_NORM HEIGHT_NORM

The class_id is the index number for different classes. The id of the first label will be 0 and an increasing integer after that. Note all the position attributes in the label file are not absolute but normalized.

X_CENTER_NORM = X_CENTER_ABS/IMAGE_WIDTH

Y_CENTER_NORM = Y_CENTER_ABS/IMAGE_HEIGHT

WIDTH_NORM = WIDTH_OF_LABEL_ABS/IMAGE_WIDTH

HEIGHT_NORM = HEIGHT_OF_LABEL_ABS/IMAGE_HEIGHT

The labeling of the images is now done and we can start training the YOLO now. Here we had divided our dataset id into 70% for training and 30% validation set.

5.2 MODEL CONFIGURATION

Ultralytics, Glenn Jocher also provides some sample YOLOv5 models built on previous theory. The YOLOv5 model on PyTorch will read these architectures from the yaml file and build it in the train.py file. This also makes it easier to configure the architecture depending on the different object detection problems.

The purpose of this project is to evaluate the performance of the YOLOv5 algorithm, so the original architecture will be used and will temporarily not configure or add other algorithms and optimization methods to the model.

Because this sample YOLOv5 architecture is used to train the COCO dataset, the number of classes defined is 80. For our dataset, the number of classes needs to be adjusted. Since the anchor box auto-learning has been integrated, the anchor box parameters can be ignored as default.

Weights *&* **Biases** is the best tools for machine learning. W&B is used for experiment tracking, dataset versioning, and collaborating on ML. Here we have used it for the live tracking of our dataset while training

5.3 TRAINING THE MODEL

Starting the training process by the following command as shown above.

The meaning of the following arguments:

- **train.py**: the model will be trained by this file.
- **img:** Here we are defining input image size. We are compressing the image to a smaller size to make the training process faster. After many experiments, many computer vision researchers agreed that the size 416×416 is the ideal size to use as input without losing much detail.
- **batch:** determining the batch size. The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters. The forwarding of thousands of images into the neural network at the same time makes the number of weights that the model learns at one time (one epoch) to increase a lot. Thus, the dataset is usually divided into multiple batches of *n* images and training batch by batch. The results of each batch are then saved to RAM and aggregated after the training for all batches is completed. Because the weights learned from the batches are stored in RAM, so the larger the number of batches, the more memory consumption will be consumed.
- **epochs:** Here we are defining the number of training epochs. The number of epochs is a hyperparameter that defines the number of times that the learning algorithm will work through the entire training dataset. One epoch means that each

sample in the training dataset has had an opportunity to update the internal model parameters. An epoch is responsible for learning all input images, in other words, training all input. Since the dataset is split into multiple batches, one epoch will be responsible for training all the batches. The number of epochs represents the number of times the model trains all the inputs and updates the weights to get closer to the ground truth labels.

- **data:** the path to data.yaml file containing the summary of the dataset. The model evaluation process is executed immediately after each epoch, so the model will also access the validation directory via the path in data.yaml file and use its contents for evaluation at that moment.
- **cfg:** specify our model configuration path. Based on the architecture defined in the model yaml file previously, this command line allows the train.py file to compile and build this architecture for training input images.
- **weights:** specify a path to weights. A pretrained weight can be used for saving training time. If it is left blank, the model will automatically initialize random weights for training.
- **name:** name of result folder. The model will be created in a directory containing all the results performed during training.
- **cache:** cache images for faster training

```
Epoch   gpu_mem       box       obj       cls     total   targets  img_size
92/99     1.86G   0.03644   0.06178  0.002908    0.1011        35       416:
           Class     Images     Targets         P         R      mAP@.5
             all   1.06e+03   3.6e+03      0.49     0.746     0.671

Epoch   gpu_mem       box       obj       cls     total   targets  img_size
93/99     1.86G   0.03634   0.06176  0.003035    0.1011        78       416:
           Class     Images     Targets         P         R      mAP@.5
             all   1.06e+03   3.6e+03     0.492     0.749     0.672

Epoch   gpu_mem       box       obj       cls     total   targets  img_size
94/99     1.86G   0.03597   0.06099  0.002797   0.09975        61       416:
           Class     Images     Targets         P         R      mAP@.5
             all   1.06e+03   3.6e+03     0.481     0.753     0.671

Epoch   gpu_mem       box       obj       cls     total   targets  img_size
95/99     1.86G   0.03629   0.06211  0.003008    0.1014        39       416:
           Class     Images     Targets         P         R      mAP@.5
             all   1.06e+03   3.6e+03     0.482     0.749     0.669

Epoch   gpu_mem       box       obj       cls     total   targets  img_size
96/99     1.86G   0.03608   0.06102  0.002721   0.09982        47       416:
           Class     Images     Targets         P         R      mAP@.5
             all   1.06e+03   3.6e+03     0.488     0.754      0.67

Epoch   gpu_mem       box       obj       cls     total   targets  img_size
97/99     1.86G   0.03637   0.06089  0.002736       0.1        30       416:
           Class     Images     Targets         P         R      mAP@.5
             all   1.06e+03   3.6e+03     0.497     0.751     0.678

Epoch   gpu_mem       box       obj       cls     total   targets  img_size
98/99     1.86G   0.03575   0.06091  0.002907   0.09957        61       416:
           Class     Images     Targets         P         R      mAP@.5
             all   1.06e+03   3.6e+03       0.5     0.748     0.678

Epoch   gpu_mem       box       obj       cls     total   targets  img_size
99/99     1.86G     0.036   0.06106  0.002987       0.1        72       416:
```

```
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95:
                 all       1059       3601      0.761      0.636      0.655      0.315
                 hat       1059       1252      0.907      0.871      0.895      0.487
                vest       1059       1262      0.847      0.876      0.872      0.491
             goggles       1059        148      0.616      0.534      0.548      0.232
               glove       1059        450      0.731       0.38      0.402      0.167
                shoe       1059        489      0.706      0.517       0.56      0.199
100 epochs completed in 0.940 hours.

Optimizer stripped from runs/train/yolov5s_results2/weights/last.pt, 14.8MB
Optimizer stripped from runs/train/yolov5s_results2/weights/best.pt, 14.8MB
CPU times: user 44.7 s, sys: 9.39 s, total: 54.1 s
Wall time: 57min 7s
```

Fig - 12: Training progress in 100 epochs

We can observe the various results and graphs from the add-in called TensorBoard that visualized the entire training process. We can also observe our training result on the dashboard of Weights & Biases. Here in the above figures, we have shown for YOLOv5s. The same steps are repeated for YOLOv5m, YOLOv5l, YOLOv5x.

5.4 DEPLOYMENT OF THE MODEL

As we can see model saves 2 weighting results as pt file. As shown in *Fig - 12,* last.pt file is the weight at the last epoch and best.pt file is the weight at the last epoch for the highest accuracy. The size of both files is only 14.8MB, making it very light to integrate into web or mobile application. That's why here we have used YOLOv5s for deploying purposes.

Hereby using Flask we have developed our Webpage which will take the image from the user and when we click on predict it will predict that the mention classes(object) are there or not in the image.

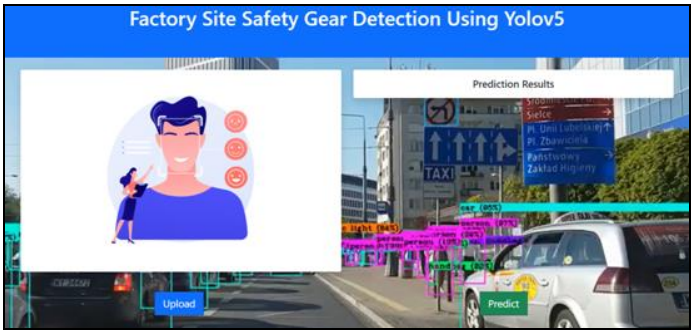


Fig - 13: Webpage using flask on which our trained model is deployed

Webpage On Heroku:

1. Log in to Heroku & Create the project
2. Choose Version control system
3. Link the repository with Heroku

Link: https://custom-object-detection.herokuapp.com/

6. RESULTS AND ANALYSIS

6.1 EVALUATION OF MODEL ON DIFFERENT PARAMETERS

To compare the performance between the two algorithms, we have used four parameters (Precision, Recall, F1 Score, mAP).

Mean Average Precision(mAP): To determine a score that compares the ground-truth bounding box to the detected box. The higher the score, the more accurate the model is in its detections.

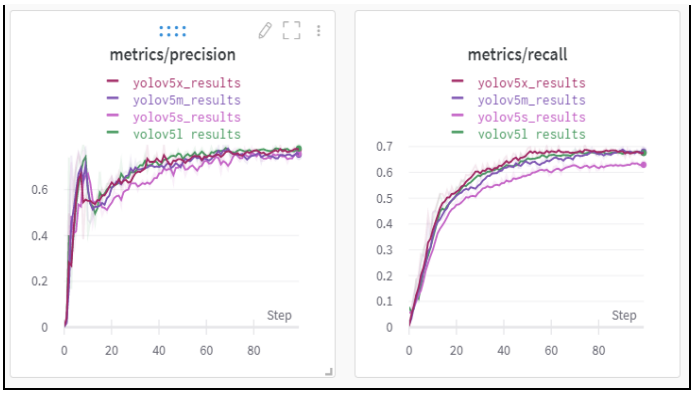
F_1 Score: To determine the test's accuracy.

Precision: To determine the ratio of correctly predicted positive observations to the total predicted positive observations.

Recall: To determine the ratio of correctly predicted positive observations to all observations in actual class

Model	Precision	Recall	F_1Score	mAP
YOLOv5s	0.761	0.636	0.690	0.655
YOLOv5m	0.781	0.658	0.721	0.679
YOLOv5l	0.785	0.681	0.735	0.681
YOLOv5x	0.789	0.688	0.738	0.689

Table 1: Parameters of all the model



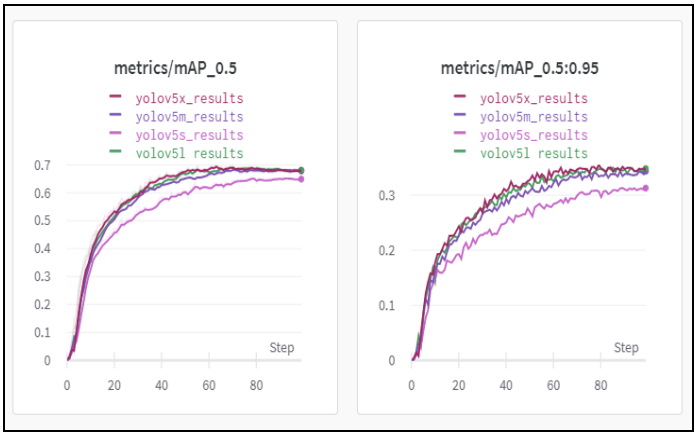


Fig - 14: Parameters Comparison of all models

From Table - 1 & Figure - 14 we can observe that YOLOv5s has the least precision, recall, F_1 Score, and mAP, YOLOv5x has the maximum. We observed that as the depth and width of the network i.e.YOLOv5 is increasing we are getting much better results. Increasing the depth increases the capacity of the model, while wider residual networks allow many multiplications to be computed in parallel. Thus helping us to increase its efficiency.

6.2 CONFUSION MATRIX

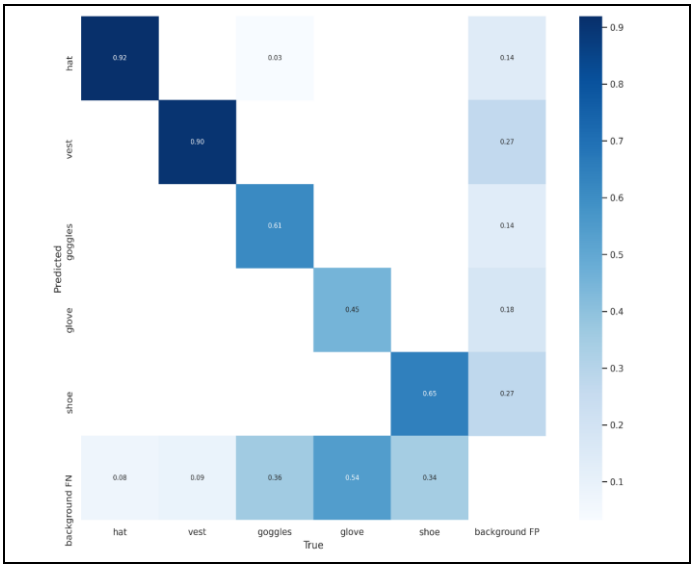


Fig - 15: Confusion matrix for YOLOv5s

6.3 COMPARISON OF LOSSES IN YOLOv5 MODELS BASED ON THEIR SIZE.

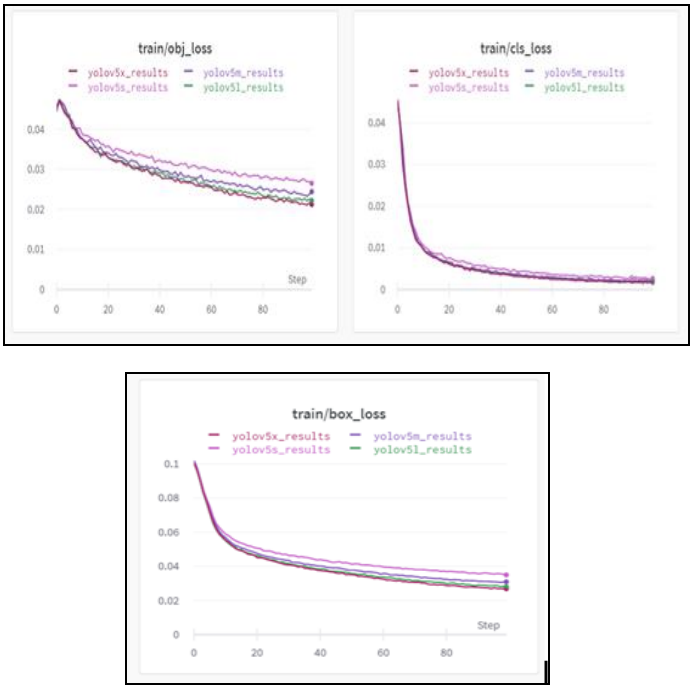
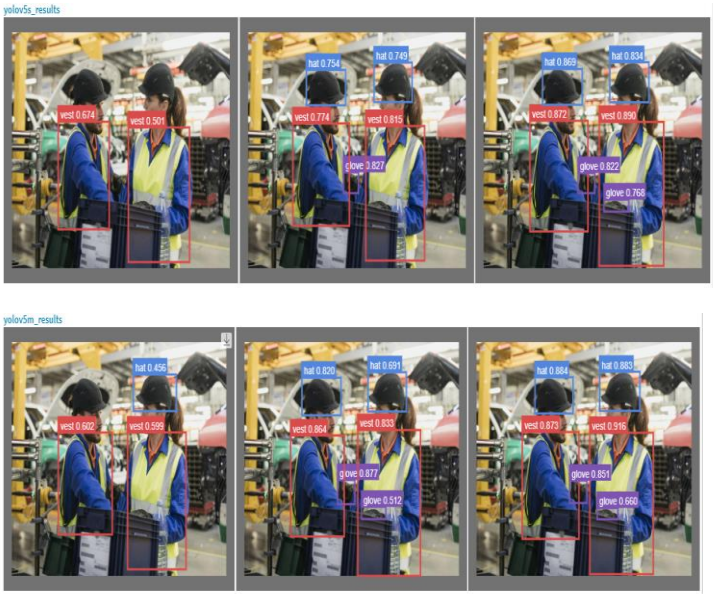


Fig - 16: Loss for all YOLOv5 model

Here we can observe that YOLOv5s has more loss, while YOLOv5x has the least. As the depth and width of the network are increasing we are getting much better results. Increasing the depth increases the capacity of the model, while wider residual networks allow many multiplications to be computed in parallel. Thus, helping us to increase its efficiency.

6.4 IMAGES AFTER DETECTION WITH BOUNDING BOXES



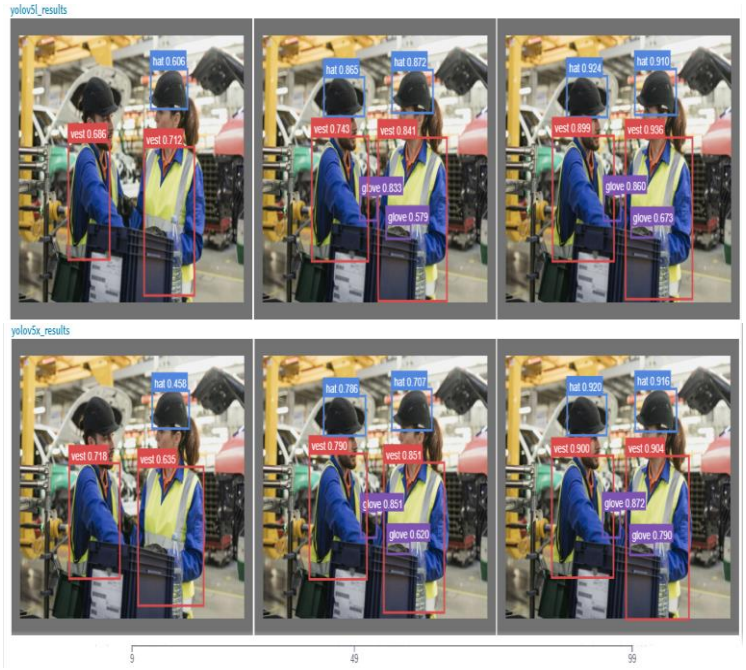


Fig - 17: Detection of the image after a different number of epochs (i.e. 9, 49 & 99) for all YOLOv5 model

In Fig – 17 we can see that as the number of epochs increases model is predicting more accurately this is because the number of epochs is a hyperparameter that defines the number of times that the learning algorithm will work through the entire training dataset. One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE. So if we increase the number of epochs we will get more accurate results.

6.5 DETECTION ON DEPLOYED WEBPAGE ON HEROKU

Figure - 18: Webpage after detection of the image

As we can see in Fig - 18 we have successfully deployed our model which, in this case, successfully detects the 'vest' and the 'hat' of the person wearing it.

7. CONCLUSIONS

To sum up, we got an overview of object detection and its implementation using deep learning. We realized the core idea of object detection and how it is implemented in the real world using various methods and specifically using deep learning. We went through the history of YOLOv5 models and also implemented them to check the accuracy of this architecture. We initially trained our detector by preparing a dataset for its input. Then we evaluated the model and visualize the training data. We also arranged the setup of our detector on a webpage. Through our model and the web application we have created, we proved to successfully justify our motive for the utmost safety and protection of workers. YOLOv5 is lightweight and extremely easy to use. YOLOv5 trains quickly, inferences, and performs well.

- We took a dataset of custom images to train the model using YOLOv5. This is also referred to as our training phase.
- We then made our inference based on the test images after seeing how our model predicts them. This comprises the testing phase.
- We saw that as the width and depth of the YOLOv5 model increases the efficiency also increases. The different YOLOv5 models we used are YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x These are models with increasing depth and width respectively.
- It was observed that we can obtain the results amazingly fast with some discrepancies.
- For the deployment of our model, we have used Heroku to make our web application.
- Even when a picture is uploaded on the webpage, our model successfully recognizes the objects present in the image only if it belongs to one of the 5 classes we have used to train our machine

REFERENCES

[1] A Gentle Introduction to Object Recognition With Deep Learning by Jason Brownlee on May 22, 2019, in Deep Learning for Computer Vision (Online) https://machinelearningmastery.com/object-recognition-with-deep-learning/#:~:text=Object%20detection%20is%20slow.&text=Fast%20R%2DCNN%20is%20proposed,a%20deep%20convolutional%20neural%20network

[2] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.2016.

[3] Review of Deep Learning Algorithms for Object Detection Arthur Ouaknine Feb 5, 2018 (Online)https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

[4] A Survey of Modern Object Detection Literature using Deep Learning arXiv:1808.07256v1 [cs.CV] 22 Aug 2018

[5] Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 7263–7271 (2017)

[6] Redmon, J., Farhadi, A.: YOLOv3: an incremental improvement. arXiv preprint arXiv:1804.02767 (2018)

[7] Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: YOLOv4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)

[8] S. Kanchana, P. Sivaprakash, Sebastian Joseph, "Studies on Labour Safety in Construction Sites", The Scientific World Journal, vol. 2015, Article ID 590810, 6 pages, 2015. https://doi.org/10.1155/2015/590810

[9] You Only Look Once: Unified, Real-Time Object Detection: Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi University of Washington arXiv:1506.02640v5 [cs.CV] 9 May 2016

[10] A Practical Guide to Object Detection using the Popular YOLO Framework PULKIT SHARMA, DECEMBER 6, 2018 (Online) https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framewor-python/#:~:text=YOLO%20is%20a%20state%2Dof,each%20grid%20contains%205%20boxes

[11] Object Detection with Deep Learning: A Review arXiv:1807.05511v2 [cs.CV] 16 Apr 2019

[12] YOLOv5 by Glenn-Jocher (Ultralytics) https://github.com/ultralytics/yolov5

[13] A Real-Time Apple Targets Detection Method for Picking Robot Based on Improved YOLOv5 B Yan, P Fan, X Lei, Z Liu, F Yang - Remote Sensing, 2021 - mdpi.com

[14] N. Avinash, *Neural Network Models in R*, 2019. [Online]. Available: www.datacamp.com/community/tutorials/neural-network-models-r

[15] B. Jason, A Gentle Introduction to Computer Vision, 2019. [Online]. Available:www.machinelearningmastery.com/what-is-computer-vision/

[16] M. Anukrati, *A Comprehensive Guide to Types of Neural Networks*, 2019. [Online]. Available: www.digitalvidya.com/blog/types-of-neural-networks/

[17] A Forest Fire Detection System Based on Ensemble Learning, February 2021, Research Gate