Technical Report:
A YOLOv3 FRAMEWORK FOR DETECTING PERSONAL PROTECTIVE EQUIPMENT

Anh Nguyen

***Abstract*— In this paper, we propose a YOLOv3 based framework for detecting types of personal protective equipment (PPE) used in constructions. In particular, we firstly u the transfer learning method with the pre-trained model which was trained on the COCO dataset. Additionally, we will provide the algorithm which will verifying the types of protective equipment the worker wearing. Finally, we will evaluate the detection performance of the model which is trained with our customized dataset with two types of common standard: COCO standard and PASCAL VOC standard.**

***Keywords*—Object detection, convolution neural network, one-stage detector strategy, deep learning.**

I. INTRODUCTION

It is fact that various object detection researches have leveraged deep learning techniques, especially Convolution Neural Networks (CNN). Generally, deep learning models used for object detection can be split into two main groups. The first group firstly extract regions, which contain target objects. Then, detected regions are fed into a classifier for detecting the objects. This group is referred to as region-based detector. There are two well-known algorithms in this type of algorithms: Fast R-CNN and Faster R-CNN. With Fast R-CNN, the input image having a bunch of region proposals passes the feature extraction model (e.g VGG-16) to get the feature map. Each region on the feature map which corresponds to a region proposal on the input image will continue passing the RoI (Regions of Interest) pooling layer to have the fixed size to be able to pass the fully connected layers and give a feature vector. Finally, this vector will be fed into one fully connected layer to classify and another fully connected layer to estimate the bounding box. With Faster R-CNN [1], a CNN model will be used to generate the feature map from the input image, then this feature map is passed in to two module: the first module is a Region Proposal Network (RPN) to get region proposals, the second module will use these proposals of RPN to capture the regions on the shared feature map which is fed into RPN to reshape them and make classification and bounding box regression, this operation is the same as Fast R-CNN. Compared to Fast R-CNN, Faster R-CNN is more effective in both terms of detection performance and time consuming, results on PASCAL VOC2007 shows that Faster R-CNN with VGG-16 backbone get 73.2 mAP score with 7 FPS, while Fast R-CNN get 66.9 mAP with around 3 FPS.

The second approach aims to detect regions and objects simultaneously, referred to as one-stage detectors. the presentation for this type of algorithms are YOLO (You Only Look Once) [2] [3] [4] and SSD (Single Shot Detectors).

In terms of YOLO, it has many different versions with the first YOLO version (we can call it YOLOv1), it uses a single CNN architecture having 24 convolutional layers followed by 2 fully connected layers to give the feature map. On each cell of the final feature map which is also the detection map, consist of the offsets for bounding box scales and the classification score of a type of object that the cell may contain, the shape of the detection map is $S \times S \times (B \times 5 + C)$, where S is the width and height, B is the number of bounding boxes which is estimated per a cell and C is the number of classes. YOLOv1 achieves 63.4 mAP score on PASCAL VOC 2007 with the processing rate is 45 FPS, it outperforms Faster R-CNN with VGG-16 backbone which only has 7 FPS. With SSD [5], it has multiple detection on multiple scales of feature maps. To make predictions on each feature map, it uses the 3×3 filter to slide over all locations of the feature map. For one feature map having the size $m \times n$, each grid cell will give 4 offsets, c class scores for one out of k default bounding boxes thus it will need $(4 + c) \times k$ 3×3 filters applied for a cell on the feature map $m \times n$. SSD shows impressive effectiveness of using multiple scales of feature maps to make prediction, it achieves 74.3 mAP score on PASCAL VOC2007 test set with VGG-16 backbone and the input resolution as 300. Furthermore, by using this network configuration, SSD can operate at 46 FPS.

Although SSD is good as time consuming and shows good performance with the dataset which contains large objects such as PASCAL VOC dataset, SSD struggles with small objects. With VGG-16 and the input resolution as 300, on the COCO test-dev 2015 as small objects, SSD only gets 5.5 AP_S while Faster R-CNN gets 7.7 AP_S. By using meta CNN architecture such Inception RestNet V2 proves that the heavy backbone network does not improve the performance of SSD in detecting small objects [6]. The less performance with small objects can be explained by the lost information about location as deeper layers. Deeper layers have strong semantic, this is good for classification but having strong semantic also causes the loss in location information. Therefore, combining the earlier feature map with the deeper feature map can help detect small objects to be better. This is proved in [7] by using Feature Pyramid Networks (FPN) to give output that makes performance of

Faster R-CNN in small objects to be improved, at 18.2 AP_S on COCO test set. Inspired by FPN, in the version 3 of YOLO framework (called YOLOv3) with Darknet-53 architecture, YOLOv3 achieves 18.3 AP_S score on COCO test set and 57.9 AP_{50}, this outweighs the SSD531 with ResNet-101. Moreover, YOLOv3 only needs around 51 ms for time inference while Faster R-CNN using Resnet-101 backbone combined with FPN takes 172 ms and SSD531 takes 125 ms. This make the YOLO frameworks more preferable in many application in practice.

In terms of real-time detection algorithms, YOLO framework is one of the prior selections, this is because it not only has fast rate but also keeps the detection performance really competitive at low thresholds with other complex models such as Faster R-CNN with TDM. Additionally the performance in detecting small objects is pretty high with the most of two stage detectors. Nowadays, YOLO framework is optimized with C++ libraries like TensorRT to run better with NVIDIA graphics process units, this makes YOLO more suitable for the embedded kits having GPUs such as Jetson nano.

II. Datasets and tasks proposed

A. Propose a dataset of personal protective equipment

We reuse two published dataset: crowd-sourced dataset [8] and hard hat and safety vest dataset which is available on Kaggle competition. Fig. (1a) and (1b) present some samples in both datasets, to create our own dataset.

With the dataset which is obtained from Kaggle website, we remove negative images which only contain backgrounds and we have the Table I that lists the number of objects per a class for each datasets.

Moreover, to help the training process become faster, we apply the transfer learning method with the pre-train model is trained with COCO dataset (Common Object in Context dataset) [9] which is provided by COCO challenge and containing 330K images with 80 general object categories. The pre-train model may generate general features of objects and the model will not learn to extract features in new dataset from beginning which will save time for training process. With transfer learning method, it allow to solve the problem about the lack of the dataset for training the deep learning model that we have mentioned.

TABLE I: The number of instances per a class and the image totals of two datasets

Class	Crowd-sourced dataset	Kaggle dataset
Hat	1646	2585
Vest	42	2144
Worker	2497	0
Image total	785	2186

B. The task definition

Task 1, with the YOLOv3 model, we will train and test the recognization on three types of objects such as Vest, Harthat and Worker. Then we will evaluate the performance

(a) The crowded-source images

(b) The images from Kaggle website

Fig. 1: Illustration of dataset

of the model with two popular standard metrics that are given by COCO challenge [9] and PASCAL VOC challenge [10]. Additionally, we will split the original dataset in to 5-folds for cross validation.

Task 2, to check type of the equipments which are being wore by workers. In case of using two types of PPE: Hardhat (H) and safety vest (V), when considering a worker (W), there are 4 situations: Worker without PPE (W), Worker having Hardhat (WH), Worker having Safety vest (WV) and Worker having both Hardhat and Safety vest (WHV). There are many methods that uses the deep learning model to check the types of PPE are wearing by the worker. Within this paper, our proposed method of using YOLOv3 model for detecting and recognising the 3 types of object named Hardhat, Safety Vest and Worker. Next steps, we will use the verifying algorithm to determine what types of PPE the workers are wearing.

C. Evaluation metrics

To evaluate, we firstly compute the intersection over union (IOU) between the ground truth and the estimation bounding box as Eq. (1):

$$IOU = \frac{area(B_{pred} \cap B_{gt})}{area(B_{pred} \cup B_{gt})}. \quad (1)$$

Next, we determine True Positive, False Positive and False Negative as following:

- True Positive (TP): A true detection, if IOU $\geqslant$ threshold.
- False Positive (FP): A false detection, if IOU $<$ threshold.
- False Negative (FN): The bounding box is not detected.

Given the TP, FP, and FN, **Precision** which is used to measure the ratio of correctly predicted results and total positive predicted results is computed by Eq. (2)

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

Meanwhile, **Recall**, which demonstrates the sensitivity of the model with the correct predictions on the total number of positive ground truths, is computed by Eq. (3)

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

For computing Average Precision (AP), the method called all-points interpolation method is used and it can be considered the area under curve (AUC) of precision/recall curve [11]. This method is applied to both the COCO standard and the PASCAL VOC standard. The formula is showed in Eq. (4).

$$\begin{aligned} AP &= \sum (r_{n+1} - r_n) p_{interp}(r_{n+1}) \\ p_{interp}(r_{n+1}) &= \max_{\tilde{r}:r \geqslant r_{n+1}} p(\tilde{r}) \end{aligned} \tag{4}$$

The definition of Average Recall (AR) over IOU $\in [0.5, 1]$ [12] is illustrated in Eq. (5).

$$AR = 2 \int_{0.5}^{1.} recall(o) do \tag{5}$$

$recall(o)$ is the recall that is correspond to the IOU o. To compute with discrete values Eq. (5) is changed into Eq. (6).

$$AR = \frac{2}{G} \sum_{i=1}^{G} \max(IOU_i - 0.5, 0) \tag{6}$$

Where G is the number of ground truths.

AR is the simple method revealing clearly the detection performance of the model. This is because it has the relatively large correlation with the detection performance [13].

1) Metrics based on COCO standard: COCO standard metrics contain 12 metrics showed in Table II and the size of object areas are defined in Table III. In COCO standard, the AP and AR is computed over all the categories resulting in there is no the difference between AP and mAP or AR and mAP. In other word, in COCO standard, it will implement the evaluation with object class and no object class.

TABLE II: COCO standard metrics

Metric	IOU	Area	maxDets	Score
AP	$0.5 : 0.05 : 0.95$	all	100	$AP^{.5:.95}$
AP	0.5	all	100	$AP^{.5}$
AP	0.75	all	100	$AP^{.75}$
AP	$0.5 : 0.05 : 0.95$	small	100	AP^{small}
AP	$0.5 : 0.05 : 0.95$	medium	100	AP^{medium}
AP	$0.5 : 0.05 : .95$	large	100	AP^{large}
AR	$0.5 : 0.05 : .95$	all	1	$AR^{max=1}$
AR	$0.5 : 0.05 : .95$	all	10	$AR^{max=10}$
AR	$0.5 : 0.05 : .95$	all	100	$AR^{max=100}$
AR	$0.5 : 0.05 : .95$	small	100	AR^{small}
AR	$0.5 : 0.05 : .95$	medium	100	AR^{medium}
AR	$0.5 : 0.05 : .95$	large	100	AR^{large}

TABLE III: Size definition in COCO standard

Object Area	Pixels
Small	Under 32^2
Medium	Between 32^2 and 96^2
Large	Above 96^2

2) Metrics based on PASCAL VOC standard: In PASCAL VOC standard, the AP is computed per classes then the mean average precision (mAP) is averaged AP over classes. Additionally, The IOU threshold in PASCAL VOC metric is chosen the fix threshold which is 0.5.

TABLE IV: PASCAL VOC standard metrics

Object	$\mathbf{AP}^{IOU=.5}$
Hat	AP_h
Vest	AP_v
Worker	AP_w
mAP	$\frac{AP_h + AP_v + AP_w}{3}$

III. The method explanations

A. YOLOv3 algorithm explanation

YOLOv3 (You Only Look Once version 3) is one of the most popular algorithm used in solving object detection problems. YOLOv3 is a type of one stage detectors which implement both localization and classification via only one CNN model. YOLOv3 has some advantages such as: be more accuracy and be suitable for small objects. The reason makes YOLOv3 better accuracy than original YOLO is that YOLOv3 exploits the similar architecture as Feature-Pyramid Network [14] to give multiple outputs. Meanwhile, the original YOLO makes the ouputs on only one scale of the feature map. The output of YOLOv3 will be a feature map and for each of cells, given one prediction which is illustrated in Fig. 2 is a vector having the shape as following: $[p_0, t_x, t_y, t_w, t_h, p_1, ..., p_N]^T$, in which for N classes, we have N probabilities on each classes, one confident score p_0 which give the probability of the object can be contained in the the grid cell in both YOLO and YOLOv3, p_0 is defined as: $P(\text{Object}) \times \text{IOU}_{\text{pred}}^{\text{truth}}$ [2]. To make the confident score having the value in interval of $[0, 1]$, YOLOv3 uses the sigmoid function [4] to give the result. At test time, the confident score will be re-defined as Eq. (7):

$$\text{confident score} = p_i * p_0 \tag{7}$$

Where p_i with $i \neq 0$ is the probability of a specific class.

In addition, sigmoid function are used at the output feature maps to generate other probabilities of classes. This is because when applying the sigmoid function to get the probabilities, it means the condition probability: $P(\text{Class}_i|\text{Object})$ [2], this will tell us that how much chance this object belongs to the specific class. Using the sigmoid function and the threshold which the condition probabilities have to be greater than it in order to be assigned the class to the object allows an object to have many class names. Having more than one class name on an object regularly occurs in object detection problems, for example: an object is woman also can be the object of person class. Therefore, applying sigmoid function help us keep all classes which the object may belong to.

Otherwise, t_x, t_y, tw, t_h are the offsets of bounding boxes, the actual bounding boxes are determined as Eq. (8):

$$\begin{aligned} b_x &= \sigma(t_x) + C_x \\ b_y &= \sigma(t_y) + C_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \end{aligned} \tag{8}$$

Where (C_x, C_y) are the coordination of the grid cells, (p_w, p_h) are the width and the height of anchor boxes, respectively. The anchor boxes are considered initial bounding boxes and the actual boxes will be resized based on these bounding boxes. Therefore selecting anchor boxes is really important to achieve the high performance of the model. For selecting the anchor boxes, the most common method is that uses the K-means cluster algorithm (noting that the initial points equal the multiplication of the number of anchor boxes per output feature maps to the number of output feature maps).

In order to make an output feature map, YOLOv3 applies the 1×1 convolution layer [15] to the final feature map. The shape of this output feature map is determined as following: $S \times S \times (5 + N) * 3$. The depth of the output is $(5 + N) * 3$, where 5 is 4 bounding box offsets which are generated from the model to determine the actual bounding boxes of the objects and one confident score p_0 of theses actual bounding boxes, N is number of classes and 3 is number of anchor boxes per a feature map in YOLOv3 model.

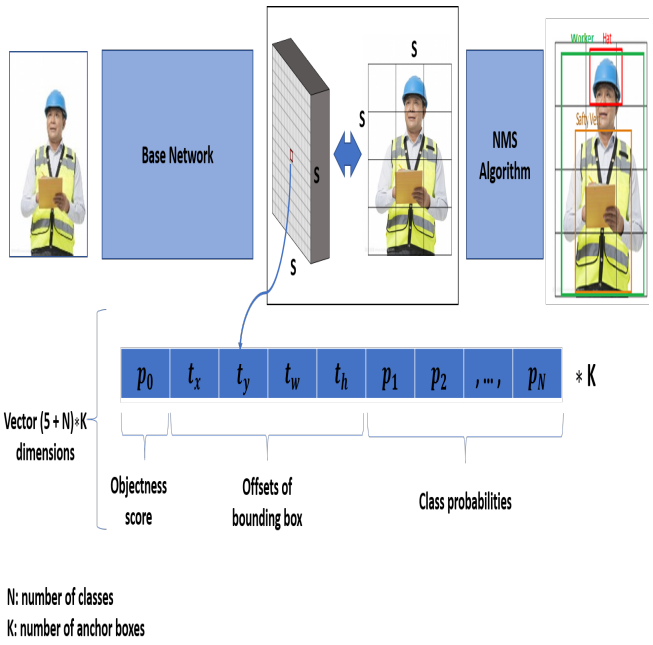

Fig. 2: The illustration of YOLO algorithm

B. Darknet-53 architecture

The Darknet-53 architecture using the residual blocks with the 1×1 convolutional layer followed by the 3×3 layer. Using residual blocks help to prevent the model from the vanishing gradient effect the 1×1 layers are used to summarize the previous feature maps and then these layers are restored by the 3×3 layers. Otherwise, to accelerate the training process after each convolutions in the entire YOLOv3 model after every convolutions, the batch nomalization layers are then applied to prevent the variant shift during training [16] and therefore we achieve the architecture of residual blocks [17][4] with the structure as showing in Fig. 4:

	Type	Filters	Size	Ouput
	Convolutional	32	3 x 3	256 x 256
	Convolutional	64	3 x 3/2	128 x 128
1x	Convolutional	32	1 x 1	
	Convolutional	64	3 x 3	
	Residual			128 x 128
	Convolutional	128	3 x 3/2	64 x 64
2x	Convolutional	64	1 x 1	
	Convolutional	128	3 x 3	
	Residual			64 x 64
	Convolutional	256	3 x 3/2	32 x 32
8x	Convolutional	128	1 x 1	
	Convolutional	256	3 x 3	
	Residual			32 x 32
	Convolutional	512	3 x 3/2	16 x 16
8x	Convolutional	256	1 x 1	
	Convolutional	512	3 x 3	
	Residual			16 x 16
	Convolutional	1024	3 x 3/2	8 x 8
4x	Convolutional	512	1 x 1	
	Convolutional	1024	3 x 3	
	Residual			8 x 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 3: Darknet-53 architecture

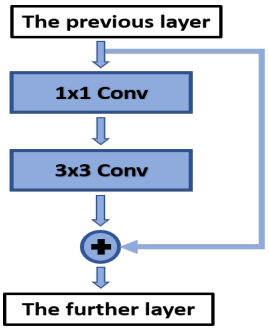

Fig. 4: The residual block in YOLOv3

C. Multiple scale predictions in YOLOv3

The output of YOLOv3 is made from 3 scales of feature maps. Since the input image has the size of 416×416, the model generates the scales of feature maps with the different sizes: 13×13, 26×26 and 52×52.

The first output is made by the final output layer at the end of darknet-53 model (layer 52^{nd}) going through other 6 conv layers, then applying the 1×1 convolutional layer to the layer 58^{th} in order to generate the output feature map with the size of 13×13. The layer 59^{th} is applied another convolutional layer to upsample 2 times, then it is concatenated with the feature map of the layer 43^{rd}. The second output is made by the layer 60^{th} (just upsampling and concatenating with layer 43^{rd}) going through 6 conv layers then being applied the 1×1 kernels,and the second ouput have the size of 26×26. The final prediction is generated by the same way which applied for the second output, the feature map 59^{th} is applied another convolutional layer with stride 2 to upsample 2 times after that it will be concatenated with the layer 26^{th} and go through some conv layers,then applying the 1×1 convolutional layer to make the output.

Predicting on the multiple scales of feature maps make the detection more effective. This is because in object detection, objects appear in the variety of scales and ratio aspects.

Therefore, generating the outputs on different scales of feature maps enhance the ability of the detection for the model. Moreover, downsampling then upsampling the feature maps may lead to the lost of the spatial information. Therefore, in order to solve this problem, the results of from previous are concatenated after upsampling process. Additionally, the resolution of the input image is high to help segment data better so that it supports YOLOv3 to detect effectively small objects.

In YOLOv3 model, the entire convolutions use the Leaky ReLu [18] as the activation function excepting for the output layers, the fomular of the activation function, showed in Eq. (9).

$$f(x) = \begin{cases} \alpha x, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{9}$$

The entire structure of the YOLOv3 architecture is illustrated in Fig. 5. It will make three scales for the output and not at all these bounding boxes are useful, most of them are the duplicates. Therefore, to keep the best bounding boxes and remove redundant bounding boxes, the Non-maximum Suppression is used in the test stage to remove these duplicate boxes. Additionally, The output feature maps which are low resolution will take over to detect objects having the big shape while the higher resolution feature map can detect the objects with the small shape

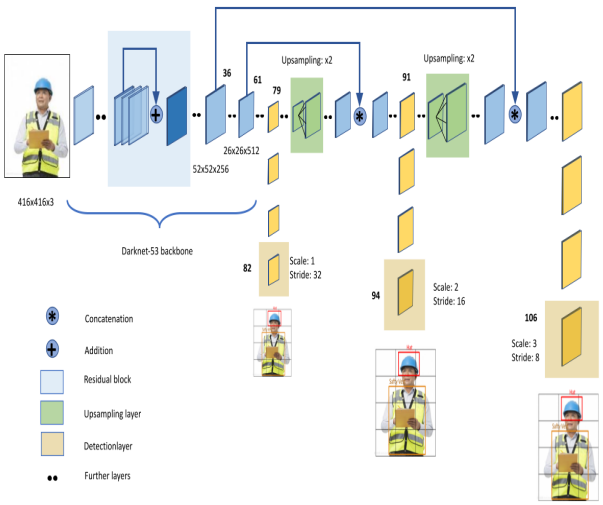

Fig. 5: The YOLOv3 architecture

D. Loss function

The loss function in YOLOV3 is showed in the Eq. (10). It includes three terms: localization loss, classification loss and confident loss and they are simultaneously minimized during the training step.

$$\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 \\
&+\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned} \tag{10}$$

To implement the process of maximizing the average precision (noting that this precision depends on the bounding box estimations and the reducetion in the impact of the cells which do not contain objects), in the loss function, λ_{coord} should be greater compared to the rest elements and the λ_{noobj} should be set as small weight. In training process, we set $\lambda_{coord} = 5$ and $\lambda_{noobj} = 0.5$.

E. The system explaination

First, YOLOv3 model use Non-Maximum Suppression (NMS) remove the duplicate bounding boxes [19]. Next, we will verify what types of PPE which the worker is wearing. There are many ways to check such as: neuron network, decision tree,... Our method is proposed as the use the IOU (Intersection Over Union) between each types of PPE with the bounding box of the worker. Specifically, if the value of IOU when classifying the specific PPE with the worker is greater than the threshold, it will be verified to belong to this worker. The entire algorithm to verify is presented in algorithm (1).

Finally, the whole system will be illustrated in the Fig. 6:

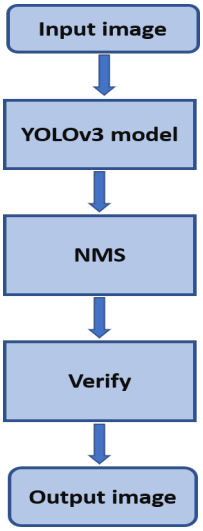

Fig. 6: The flowchart for the system

Algorithm 1: Structure to verify PPE equipment

Result: S is the set of PPE types that W_i is wearing
Set $S = \emptyset$;
for *each worker detected in* W_i **do**
  for *each detected box* $P \in \{H, V, ...\}$ **do**
    Compute the area of box;
    Compute the IOU;
    if $IOU \geqslant threshold$ **then**
      Insert P in the set S, i.e., $P \in S$;
      Break and go to the next type of PPE;
    end
  end
end

Fig. 7: The illustration for the DA

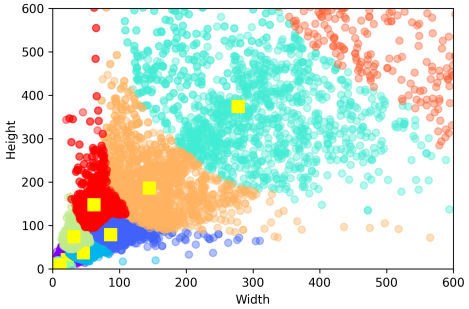

Fig. 8: The illustration of anchor boxes

IV. Experimental Settting

A. Data Augmentation

Data Augmentation (DA) is a kind of regularization that help to prevent the deep learning model from the overfitting effect during testing [20] and therefore it plays an important role in many deep learning models. For image processing, DA can generate new input image via many transformations, such as: flipping, distorting, replacing, etc. For objects in constructions, we choose effects: flipping, distorting, replacing, resizing for the images. This is because in construction environment, we often deal with the lighting problems and the multiple of workers' pose and therefore if the model just learn the original images, this can lead to the low performance. Every image before being fed into the model for training will be resized, placed and distorted. Next, the flipping effect is then applied with the probability is 0.5. Distorting the images will randomly changing the factors such as: hue, saturation and value in the HSV color space with the percentage intervals showed in Table V and some samples for original the image after going through DA step are show in Fig. 7.

TABLE V: THE CHANGING INTERVALS OF FACTORS IN HSV COLOR SPACE

Factor	Changing interval
Hue	$(0; \pi)$ rad
Saturation	$(0; 33)$ %
Value	$(0; 33)$ %

B. Anchor Box Setting

The chosen anchor boxes created by applying the K-mean clusters algorithm in training set are showed in Fig. 8 and the sizes of anchor boxes for each scale of output feature maps are show in the Table VI. The accuracy of those bounding boxes is 63.52%

C. Training Process Setting

The training process includes two stages, in the first stage, only output layer is trained and other layer is freezed to utilize the general feature which was generated by training the model with the huge dataset. we set the model trained with the batch size of 50 and the learning rate is 0.001 during 50 epoches, the best model will be saved for the second phase.

In the second phase of training, the entire network will be unfreezed and trained with a very small learning rate. The main purpose in the second stage is to make the model more adaptive with the dataset and the weights of kernels of the model should not change so much, this is the reason is that the learning rate should be very small. In this step, we set the learning rate as 0.0001 with the batch size is 16.

During training the deep learning model, one of the popular issuses which often occur that is overfitting effect. The popular method to solve this problem is to adds the dropout layers immediately behind each convolutional layers [21], but in our model, using dropout layers will not be effective by two reasons. First, in the first phase of training, the model was freezed all layers except for the output layers whose weights was set trainable and therefore there is no place to put the dropout layers. Second, during the second phase, the model was unfreezed, but we expect the model change slightly to be more adaptive to the data and adding the dropout layers which in turn make the network update its weights significantly. Since using the dropout method to tackle the overfitting problem due to two mentioned reasons, therefore we choose the early stopping method to preserve

TABLE VI: The anchor box sizes on each scales

Scale	Size
13×13	$(11 \times 9); (22 \times 21); (32 \times 74)$
26×26	$(46 \times 37); (62 \times 148); (87 \times 79)$
52×52	$(145 \times 186); (278 \times 374); (579 \times 671)$

the good result in training and we set that after 10 epoches if the validation loss is not improved, the training process will be finished.

Another method helping to improve the performance and reducing in the training time is the learning rate schedule. After 3 epoches, if the validation loss is not better, the current learning rate will be multiplied by 0.1.

V. Experimental Results and Discussion

A. Experimental Results

The results conducted with COCO standard are showed in Table VII. With AP metrics, these show that the AP scores reduce significantly with higher IOU thresholds and objects which have small size. Furthermore, with COCO standard metrics, the sensitive level of the model which is reflected via AR scores shows the low performance with the images having the less amount of objects and small objects.

TABLE VII: COCO standard results

Metric	IOU	Area	maxDets	Score
AP	$0.5 : 0.05 : 0.95$	all	100	0.37
AP	0.5	all	100	0.66
AP	0.75	all	100	0.37
AP	$0.5 : 0.05 : 0.95$	small	100	0.07
AP	$0.5 : 0.05 : 0.95$	medium	100	0.28
AP	$0.5 : 0.05 : 0.95$	large	100	0.52
AR	$0.5 : 0.05 : 0.95$	all	1	0.25
AR	$0.5 : 0.05 : 0.95$	all	10	0.43
AR	$0.5 : 0.05 : 0.95$	all	100	0.43
AR	$0.5 : 0.05 : 0.95$	small	100	0.10
AR	$0.5 : 0.05 : 0.95$	medium	100	0.34
AR	$0.5 : 0.05 : 0.95$	large	100	0.58

Table VIII shows PASCAl VOC standard metrics and it gives more detail of evaluation for each class so that we can realise the performance of the model to each types of object. It is notable that the value of mAP in Table VIII is equal to the value of AP with IOU $= 0.5$ in Tabel VII. This can be explained by when computing AP with the fix threshold 0.5, COCO standard used the same method as AP computing method in PASCAL VOC standard, but in COCO standard, AP is computed with all types of objects and in PASCAL standard is computed on each class and the mAP is average over AP of each class thus resulting in the quite similar results can occur. Otherwise, in Tabel VIII, it shows the detection performance on 3 classes is quite balancing.

TABLE VIII: PASCAL VOC standard results

Object	$\mathbf{AP}^{IOU=.5}$
Hat	0.68
Vest	0.66
Worker	0.65
mAP	0.66

B. Discussion

For COCO standard metrics, looking at Table VII, it shows that this YOLOv3 model can operate well with the large objects. And in Table VIII, this shows the balance of the detection performance of the model over classes and therefore we can ignore the concern related whether the model have the same performance on all 3 classes.

With the analysis on both standards, when combining the model with the verifying algorithm, this PPE system should apply with the close distance to give the high performance.

VI. Conclusion

This paper presents our work of applying YOLOv3 algorithm to recognise the types of objects like the hardhat, the safety vest and the worker, then we propose the method to verify the types of personal protective equipment which workers are wearing. For training YOLOv3, we apply the transfer learning technique which is useful for training the model on the customizing dataset. Finally, we show how to evaluate the performance of YOLOv3 model on two common standard: COCO standard and PASCAL VOC standard.

References

[1] Ross Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[3] Joseph Redmon and Ali Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[4] Joseph Redmon and Ali Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[6] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al., "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7310–7311.

[7] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[8] Nipun D Nath, Amir H Behzadan, and Stephanie G Paal, "Deep learning for site safety: Real-time detection of personal protective equipment," *Automation in Construction*, vol. 112, pp. 103085, 2020.

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[11] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237–242.

[12] Rafael Padilla, Wesley L Passos, Thadeu LB Dias, Sergio L Netto, and Eduardo AB da Silva, "A comparative analysis of object detection metrics with a companion open-source toolkit," *Electronics*, vol. 10, no. 3, pp. 279, 2021.

[13] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele, "What makes for effective detection proposals?," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 4, pp. 814–830, 2015.

[14] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[15] Min Lin, Qiang Chen, and Shuicheng Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[16] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[19] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis, "Soft-nms–improving object detection with one line of code," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5561–5569.

[20] Luis Perez and Jason Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.