Object Detection Model for Web-Based Physical Distancing Detector Using Deep Learning

Andry Chowanda[1]*, Ananda Kevin Refaldo Sariputra[2], and Ricardo Gunawan Prananto[3]
[1−3]Computer Science Department, School of Computer Science, Bina Nusantara University
Jakarta, Indonesia 11480
Email: [1]achowanda@binus.edu, [2]ananda.sariputra@binus.ac.id, [3]ricardo.prananto@binus.ac.id

***Abstract*—The pandemic has changed the way people interact with each other in the public setting. As a result, social distancing has been implemented in public society to reduce the virus's spread. Automatically detecting social distancing is paramount in reducing menial manual tasks. There are several methods to detect social distance in public, and one is through a surveillance camera. However, detecting social distance through a camera is not an easy task. Problems, such as lighting, occlusion, and camera resolution, can occur during detection. The research aims to develop a physical distancing detector system that is adjusted to work with Indonesian rules and conditions, especially in Jakarta, using deep learning (i.e., YOLOv4 architecture with the Darknet framework) and the CrowdHuman dataset. The detection is done by reading the source video, detecting the distance between individuals, and determining the crowd of individuals close to each other. In order to accomplish the detection, the training is done with CSPDarknet53 and VGG16 backbone in YOLOv4 and YOLOv4 Tiny architecture using various hyperparameters in the training process. Several explorations are made in the research to find the best combination of architectures and fine-tune them. The research successfully detects crowds at the 16th training, with mAP50 of 71.59% (74.04% AP50) and 16.2 Frame per Second (FPS) displayed on the web. The input size is essential for determining the model's accuracy and speed. The model can be implemented in a web-based application.**

***Index Terms*—Object Detection, Web-based Application, Physical Distancing Detector, Deep Learning**

I. Introduction

At the end of 2019, a newly born virus called SARS-CoV-2 from Wuhan City, China, that caused pneumonia, spread quickly around the world [1]. In 2020, the world severed a disease caused by SARS-CoV-2 that became a global pandemic called Corona Virus Disease 19 (COVID-19). Around 214 countries reported confirmed COVID-19 cases and deaths [2]. Hence, World Health Organization (WHO) recommended that all countries applied social distancing, including full city lockdowns [3]. It was known as Pembatasan Social Skala Besar (PSBB – Large Scale Social Restriction) in Indonesia.

Then, in 2021, the pandemic still was not over, and the SARS-CoV-2 virus had new variants due to mutations [4]. WHO immediately instructed all citizens worldwide to apply a new solution called physical distancing. In Indonesia, physical distancing was regulated in the Decree of the Minister of Health of the Republic of Indonesia No. HK.01.07/MENKES/382/2020 about Health Protocol, which regulated and suggested people to reduce mobilization, applied a minimum of 1 meter of physical distancing, always wore a face mask, and avoided crowds [5].

The research aims to create a physical distancing detector, a solution based on artificial intelligence using deep learning with YOLOv4 architecture and CSPDarknet53 backbone options to detect physical distancing violations and detect crowds in a video. Object detection is used to detect physical distancing violations. The object detection architecture used the most when the research is written is YOLOv4, which has a balanced accuracy level and speed that makes YOLOv4 state of the art [6]. YOLOv4 uses some methods called head, neck, and backbone. One of the backbones used in YOLOv4 is CSPDarknet53, which has multiple components such as Cross-Stage-Partial-connections (CSP) connection and residual layer that helps feature extracting better by reducing necessary computation times [7]. Meanwhile, Visual Geometry Group (VGG)16 is also one of the other feature extractors that can be used as the model's backbone [8]. VGG16 only has a convolution process and pooling, making it better in accuracy and speed than the other

Received: June 30, 2022; received in revised form: Dec. 05, 2022; accepted: Dec. 05, 2022; available online: April 29, 2024.
*Corresponding Author

backbones, such as ResNet101 and ZF-Net [9].

The research objective is to help officers and law enforcement to observe the situation and tighten the implementation of physical distancing in their areas, such as buildings or places with surveillance cameras like Closed-Circuit Television (CCTV). The physical distancing detector can detect physical distancing violations from a real-time video source such as CCTV or USB cameras. In the future, it can support IP cameras as well. Furthermore, besides real-time videos, it can also detect from a selected video file, such as recorded CCTV footage or other videos.

A. Object Detection with Deep Learning

Deep learning is a potent tool for learning patterns from unstructured data [10]. Object detection is one of the tasks that can be solved using deep learning. Object detection acts as a combination of image classification and object localization. The aim of building an object detection system is to teach machines to understand and recognize the content in the images as humans do. This task will take an image as input and generate one or more bounding boxes with a class label assigned to each bounding box. This algorithm can handle multi-class classification and localization, as well as objects that have similarities with more than one class [11].

In addition, some literature has implemented a Convolutional Neural Network (CNN). CNN is similar to traditional Artificial Neural Network (ANN), consisting of neurons that develop independently through learning. Each neuron will receive input and perform a computational function (e.g., a scalar product followed by a non-linear function). The only significant difference between CNN and ANN is that CNN is more often used in the field of pattern recognition in images. This difference makes it possible to encode image-specific features into the architecture, making the network more suitable for image-focused tasks-while reducing the parameters required to set up the model. CNN consists of three types of layers. They are a convolutional layer, pooling layer, and fully-connected layer, respectively.

Some powerful frameworks can be implemented to build an object detection system in real time, such as the Darknet framework [12]. The Darknet framework is an open-source neural network written in C and CUDA. Darknet supports both CPU and GPU computing. In addition, Darknet supports CNN and Recurrent Neural Network (RNN) (Redmon, Darknet: Open-Source Neural Networks in C, 2013-2016).

Another framework for object detection is YOLO, which stands for You Only Look Once, an object detection architecture. YOLO is a state-of-the-art architecture for currently available models for object detection. This architecture is optimal in terms of performance and accuracy in detecting objects compared to other architectures, which are generally optimal in only one area, be it performance or accuracy. YOLO now consists of four versions, whereas in YOLO version 4 or YOLOv4, the backbone used is CSPDarknet53. As for the neck, the SPP and PAN are used. As for the head, the previous version of the YOLO architecture is used, namely YOLOv3 [6]. The backbone is the first part of an architecture that extracts the features contained in an image. Meanwhile, the head contains a detector that runs the object detection process in an image. This detector can be a dense prediction layer commonly used for one-stage detectors or a sparse prediction layer that functions as a two-stage detector. YOLO detector works as follows. YOLO divides the image into S×S grids. The S×S grids is used to predict the bounding box and the confidence score for the boxes. The conditional class probabilities are also predicted using the S×S grids, which contain objects. After that, the output of the process is summarized and used as an inference result of the image, consisting of the labels and coordinate positions of the detected objects according to the input image. YOLOv4 consists of three-level detectors, each optimal level for detected objects in small, medium, and large sizes [6].

B. Physical Distancing Detector System

A physical distancing detector is a tool to detect people who violate physical distancing rules given by the WHO to tighten physical distancing practices and reduce the spread of COVID-19. It can be done by detecting people using AI trained on images of people and then implementing it into an application such that it can be used to monitor physical distancing in real time. Some similar researchers have been working on detecting people to detect physical distancing, such as using Landing AI. The application made by Landing AI detects and marks people's locations that are too close to each other based on the recommendation of social distancing. This detection consists of calibrating the Point of View (POV), and detecting and measuring length. However, it has some drawbacks that it cannot cluster the people who are too close. Another drawback is that it uses Faster R-CNN, which is pretty slow for real-time detection to detect social distancing. There are also a few other studies with a similar goal [13–15]. Those researchers aim to create a tool to monitor physical distancing. Most physical distancing detector systems are deployed in a real time. However, several problems occur, such as the position of the crowd, the lighting, the camera resolution, and some objects (e.g., trees, building and other objects) that cover the

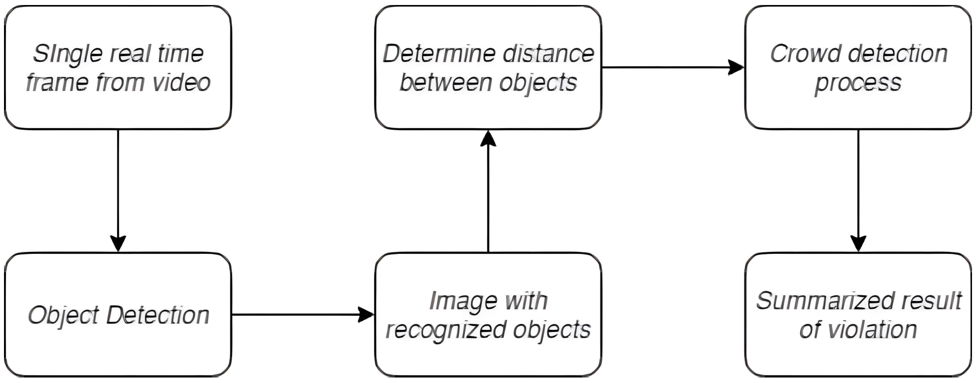

SIngle real time frame from video
Object Detection
Image with recognized objects
Determine distance between objects
Crowd detection process
Summarized result of violation

Fig. 1. Proposed methods.

person to be detected. The implementation of physical distancing detector systems can be seen in many areas, such as building and construction [16, 17], schools [18] and other public settings [19, 20].

II. Research Method

Figure 1 illustrates the proposed method in the research. In this proposed system, a single frame in each second from a video selected as the input is extracted and given to the model to be applied to object detection. This model uses one of the chosen algorithms, which will be explained later. The model then detects and produces output consisting of images with labelled objects and coordinates of the detected objects. The following steps calculate these coordinates and cluster the detected objects using the physical distancing algorithm, which consists of determining the distance between the objects, clustering the objects, and giving statistics based on those data.

A. Architectures and Experimental Settings

YOLOv4 is chosen as the detector to achieve real-time detection that is fast enough to assist officers. Regarding speed and accuracy, YOLOv4 outperforms the other detectors in performance and balance. For example, YOLOv4 achieves 64.9% with 31 Frame per Second (FPS) on an input size of 512 pixels [6]. Tested on other research, YOLOv3 surpasses its competitors, such as Faster R-CNN with only 3 FPS, 96.9% mAP and SSD with 10 FPS, 69.1% mAP. Meanwhile, YOLOv3 gains 23 FPS, 84.6% mAP [13]. YOLOv4 is the result of improvement from YOLOv3. Hence, it has better performance, and YOLOv4 is selected as the detector for the research. This architecture uses the same head as YOLOv3’s head. Researchers of YOLOv4 have chosen CSPDarknet53 as the optimum backbone to be paired with the detector part of YOLOv4. This backbone is also used as one of the tested backbones for experiments.

The research also explores VGG16 since VGG16 have incredible accuracy with good speed. In a previous study, VGG16 yields less accuracy by 1.1% but has 30 more FPS than the model using ResNet101 as the backbone. Therefore, VGG16 holds a little less accuracy while maintaining much faster than its competitor in those studies [9].

YOLOv4 Tiny architecture, the mini or small version of YOLOv4, is also explored in the research. It holds very great speed while sacrificing not too much accuracy. Tested on the same input size of 416 pixels, YOLOv4 grants 443 FPS while YOLOv4 can only achieve 82 FPS. It is tested on GeForce RTX 2080 Ti. On the other device, such as GeForce RTX 1080 Ti, YOLOv4 tiny achieves almost 375 FPS with 40.2% AP50, while YOLOv4 achieves around 48 FPS with 64.9% AP50. Since the goal is to have a fast enough and accurate model to detect people, this model is also selected as one of the methods used in the experiment. These backbones act as the feature extractor, extracting the features from the feature maps through some processes of convolution and pooling through the network. These features then are brought to the detector to be detected. This method itself may vary based on the detector used.

The aim is to see the effects of hyperparameters and variables used in training. So, the researchers experiment on such hyperparameters to see which give the best result for the current task. First, the experiment utilizes the available training method. It consists of training from scratch, transfer learning, and fine-tuning. The training is done from scratch because the YOLOv4 pretrained model is trained on the Common Objects in Context (COCO) dataset with 80 classes. This condition can cause the model to be not optimal for detecting people. Also, transfer learning and fine-tuning use a pretrained model. The difference between them is the freeze of the feature extractor part. In transfer learning, the feature extractor’s weights are frozen, while fine-tuning is not. The freezing preserves features learned from the previous scenario that models are trained on different but similar datasets. So, both have the upper hand and aim to see a better result in the research. As for the hyperparameters, there are plenty of options to choose from. The hyperparameters are learning rate, input size, steps, scales, dropout rate, and optimizers. The research experiments with the input size, steps and scales, and the number of iterations done. Input size is chosen as one of the main factors determining model accuracy. The number of iterations is also explored because the hyperparameter will change the weights as weights are only updated if iterations are done.

As for the steps, the effect of changing the learning rate is expected to be analyzed as the training continues. It is also expected to see if such an adjustment will make the gradient explode or vanish, which can be seen in the model’s accuracy during training. The architectures used in the research will be trained on the Crowdhuman dataset [21]. This dataset consists of only person class and mainly focuses on detecting people. So, this dataset is used instead of others, which usually contain other classes. This dataset annotates full body or full box, visible body or visible box, and head. The annotation’s full body and head parts will split the dataset into two classes (head and person) and be trained on the model. Hence, the trained model can detect the person and head.

B. Object Detection & Physical Distancing Algorithm

Many object detectors are researched in the community. Some popular ones are YOLOv4, Single Shot Detector (SSD), and Faster R-CNN. YOLOv4 and SSD are one-stage detectors, while Faster R-CNN is a two-stage detector. YOLOv4 uses multiple-scale detection, which helps to detect objects of different sizes. YOLOv4 also uses additional methods to help the model perform better, such as a bag of freebies and specials. YOLOv4 is trained on the COCO dataset. Then, the system will receive detected objects’ labels and position coordinates from the model. The coordinates representing the bounding boxes will be used to find the midpoint of the boxes. Then, the distance between the midpoint of every possible $(2000 * n)$ combinations of two people will be calculated based on the centre of the bounding box using the Euclidean formula, where d_x= x_1-x_2, d_y= y_1-y_2. The d_x and d_y consecutively are the diagonal, x-axis and y-axis distance, and x_1, x_2, y_1, and y_2 are the coordinates of the center of the bounding box. This formula has been used in similar works.

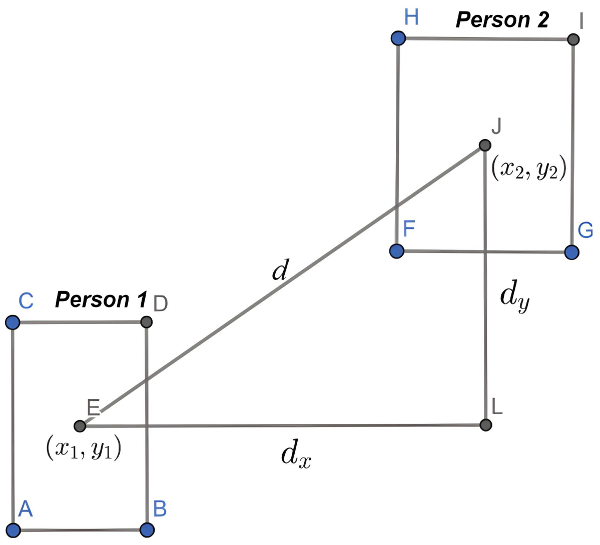


Fig. 2. Distance calculation simulation.

$$d = \sqrt{d_x^2 + d_y^2}.$$

Figure 2 illustrates the person’s distance calculation simulation. The d distance value from the previous calculation will be filtered based on several pixels, which act as the minimal distance required for the physical distancing of 1 meter in pixel, which is d based on Fig. 2. Let say this is the y value. Then, the people who are not distant from others under this y value will be filtered out and considered safe. This y value will be obtained via a scenario, and it is only valid on videos with the same conditions.

The people who are too close to each other will then be clustered according to this principle. A person will be considered a part of the same cluster or group if he or she is connected to at least one person who is also part of a cluster. The process of clustering follows this algorithm. For each combination of two people from all detected people, if the first or second person has been marked as at risk, it will search whether one has been in any cluster of all detected people. If the first person is part of a group, it will add the second person to the same group and mark them as a non-new group, vice versa for the second person. If none are part of any cluster, it will create a new cluster with both as part of the group. After that, it adds the group to the list of groups. This algorithm generates a list of clusters with people’s identifiers. It also provides the number of people in one cluster. The information that this algorithm provides will then be

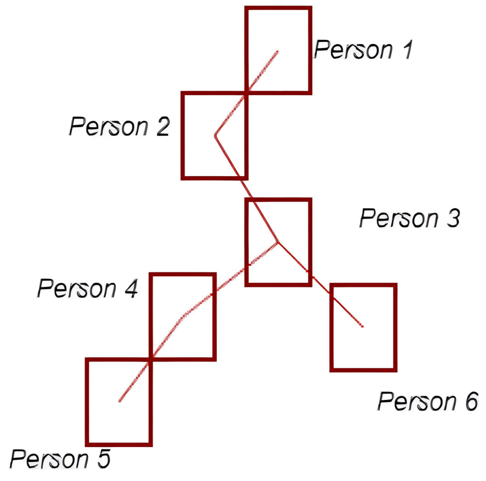


Fig. 3. Crowd calculation simulation.

used to display statistics of physical distancing activity on the video. Figure 3 illustrates the simulation of the crowd calculation algorithm.

III. Results and Discussion

The research uses the Crowdhuman dataset [21], which consists of training, validation, and testing sets. The research only uses training for the training part and validation to calculate the model’s accuracy for object detection. This dataset contains 15,000 images for the training set and 4,370 for the validation set. The dataset provides indoor and outdoor sceneries with people in it. This Crowdhuman dataset only consists of one class, which is human, with annotations of a full box, visible box, and head. This dataset is designed to identify humans in the crowd. Hence, the name is Crowdhuman.

The experiment done in the research is performed on the Windows platform using the Darknet framework with Open-Source Computer Vision Library (OpenCV) support. The hardware used is Intel Core i5-10300H Processor and NVIDIA GeForce RTX 2060 with Max-Q Design for the GPU to accelerate the training and test process. Then, the training results are compared using the validation set of Crowdhuman to calculate accuracy and speed using a sample video containing people from the Internet to evaluate the performance of the detection rate of the model.

Regarding accuracy, Mean Average Precision (MAP) and AvgIoU are used as parameters, and they will be evaluated in this experiment with confidence thresholds of 50% and 75%. The mAP50 is used as the primary metric for accuracy, and FPS for speed.

The research experiments with the models on various hyperparameters, such as input size, steps, and total iterations. Other variables, like the training method, are also tested. Table I demonstrates the results of the explorations with the input size. The dataset is trained with YOLOv4 using CSPDarknet53 backbone, and an input size of 512 pixels gains the best accuracy of 71.59% mAP50 (74.04% AP50), but a similar model using an input size of 320 pixels gains the highest FPS of 31.6–31.8.

TABLE I
Results of Input Size Exploration.

Method	Size	FPS	MAP50
YOLOv4 (CSPDarknet53)	320	31.6–31.8	55.81–57.36%
YOLOv4 (CSPDarknet53)	416	26.2–28.6	62.44–64.87%
YOLOv4 (CSPDarknet53)	512	15.7–18.3	67.03–71.59%
YOLOv4 (VGG16)	512	12.7	55.62%
YOLOv4 Tiny	320	63.1	33.30%
YOLOv4 Tiny	416	63.4	40.54–42.12%
YOLOv4 Tiny	512	63.1	48.30%

A similar condition happens on YOLOv4 Tiny, where the largest input size gives the best accuracy and vice versa. From this result, it can be concluded that a smaller input size gives a faster yet more inaccurate result as the larger input size gives the opposite result. It is because the number of the features that need to be extracted and calculated from the feature maps is smaller on the smaller input size and bigger on the larger input size, which means more features will be learned and processed.

However, in this experiment, the speed of method YOLOv4 Tiny does not differ much. It may be due to the limitations of the testing environment, such as the hardware and programs used. The result of the training method experiment is shown in Table II.

In the research, the transfer learning method performs better than the other two tested methods: trained from scratch and fine-tuning. In training using YOLOv4 (CSPDarknet53), at the beginning phase of the experiment, the research follows the recommendation given by the authors of YOLOv4 of total iteration, which is $2000 * n$ for n classes if the number of classes is greater than or equal to three classes, or 6,000 iterations for less than three classes. So, 6,000 are selected as the number of iterations.

While evaluating the experiment results, it is found that the accuracy can be further improved by increasing the number of iterations since the model is trained more than before. It is also known that if the model is trained with too much iteration, it may overfit, and the accuracy will worsen instead of improve. So, it is also experimented on 12,000 iterations. It proves

TABLE II
RESULTS OF TRAINING METHODS EXPLORATION.

YOLOv4 (CSPDarknet53), Total Iteration 6,000, Steps 4,800; 5,400										
Training Method	Input Size	FPS	Confidence Threshold 50%				Confidence Threshold 75%			
			APperson	APhead	MAP50	Avg IoU	APperson	APhead	MAP75	Avg IoU
	320	31.8	62.15%	52.57%	57.36%	43.57%	22.27%	27.23%	24.75%	30.60%
Transfer learning	416	28.6	68.50%	61.24%	64.87%	47.86%	27.47%	34.23%	30.85%	35.91%
	512	15.9	70.13%	64.90%	67.51%	50.74%	29.10%	36.24%	32.75%	39.75%
	320	31.6	59.28%	52.34%	55.81%	43.22%	18.08%	26.27%	22.17%	30.29%
Training from Scratch	416	27.2	65.29%	60.35%	62.82%	47.53%	22.89%	32.42%	27.66%	35.80%
	512	15.7	68.81%	65.25%	67.03%	50.91%	26.34%	36.74%	31.54%	40.23%
YOLOv4 (CSPDarknet53), Total Iteration 12,000, Steps 9,600; 10,800										
Training Method	Input Size	FPS	Confidence Threshold 50%				Confidence Threshold 75%			
			APperson	APhead	MAP50	Avg IoU	APperson	APhead	MAP75	Avg IoU
Transfer Learning	512	16.2	74.04%	69.13%	71.59%	57.88%	35.11%	41.19%	38.15%	40.99%
Training from Scratch	512	18.3	71.53%	66.71%	69.12%	51.19%	31.24%	38.70%	34.97%	40.35%

TABLE III
RESULTS OF ITERATION EXPLORATION.

YOLOv4 (CSPDarknet53), Steps 4,800, 5,400, Training from Scratch Method										
Total Iteration	Input Size	FPS	Confidence Threshold 50%				Confidence Threshold 75%			
			APperson	APhead	MAP50	Avg IoU	APperson	APhead	MAP75	Avg IoU
6,000	416	27.2	64.97%	59.92%	62.44%	46.43%	22.50%	32.24%	27.37%	34.88%
12,000	416	27.2	65.29%	60.35%	62.82%	47.53%	22.89%	32.42%	27.66%	35.80%
YOLOv4 Tiny, Training from Scratch Method										
Total Iteration	Input Size	FPS	Confidence Threshold 50%				Confidence Threshold 75%			
			APperson	APhead	MAP50	Avg IoU	APperson	APhead	MAP75	Avg IoU
17,391	512	63.4	51.24%	29.83%	40.54%	51.89%	13.59%	18.21%	15.90%	36.29%
24,000	512	63.1	58.52%	38.20%	48.36%	60.54%	18.24%	25.74%	21.99%	40.00%

TABLE IV
RESULTS OF STEPS EXPLORATION.

YOLOv4 (CSPDarknet53), Steps 4,800, 5,400, Training from Scratch Method										
Steps	Input Size	FPS	Confidence Threshold 50%				Confidence Threshold 75%			
			APperson	APhead	MAP50	Avg IoU	APperson	APhead	MAP75	Avg IoU
	320	31.8	62.15%	52.57%	57.36%	43.57%	22.27%	27.23%	24.75%	30.60%
4,800; 5,400	416	28.6	68.50%	61.24%	64.87%	47.86%	27.47%	34.23%	30.85%	35.91%
	512	15.9	70.13%	64.90%	67.51%	50.74%	29.10%	36.24%	32.75%	39.75%
	320	31.7	59.93%	53.01%	56.47%	51.81%	19.12%	27.13%	23.12%	31.93%
9,600; 10,800	416	27.3	65.87%	60.87%	63.37%	55.73%	23.94%	33.45%	28.69%	36.40%
	512	16.2	74.04%	69.13%	71.59%	57.88%	35.11%	41.19%	38.15%	40.99%

correct since the model’s accuracy improves on higher iterations (see Table III). The parameter determines the value of the learning rate at such an iteration. Steps set at 4,800 and 5,400 with a scale value of 0.1 will shrink the learning rate to 10% of its previous value at 4,800 and 5,400 iterations, e.g., from 0.01 to 0.001 at 4,800 iterations and 0.0001 at 5,400 iterations. This value is determined following the recommendation of the authors of YOLOv4 while using 6,000 iterations. The research still uses this step value on 12,000 iterations to analyze the impact of steps. While running an experiment on selected models, it is found that the average loss and mAP graph seem to be slowing down on such iterations. It is figured out that it should be beneficial for the model to learn faster rather than slower in that iteration range. Hence, the research also experiments with steps 9,600 and 10,800, with the comparison shown in Table IV.

Nevertheless, the experiment shows puzzling results in the same configuration. Some give better results while others do not, as Table IV shows. Next, the

TABLE V
RESULTS OF YOLOV4 (VGG16) ARCHITECTURE.

YOLOv4 (VGG16), Total Iteration 12,000, Steps 9,600, 10,800, Training from Scratch Method									
Input Size	FPS	Confidence Threshold 50%				Confidence Threshold 75%			
		APperson	APhead	MAP50	Avg IoU	APperson	APhead	MAP75	Avg IoU
512	12.7	61.00%	50.25%	55.62%	43.04%	16.69%	24.66%	20.67%	32.51%

TABLE VI
RESULTS OF TOP FIVE ACCURACY (ALL OF WHICH ARE YOLOV4 (CSPDARKNET53)).

Training Method	Total Iteration	Steps	Input Size	FPS	Confidence Threshold 50%				Confidence Threshold 50%			
					APperson	APhead	MAP50	Avg IoU	APperson	APhead	MAP75	Avg IoU
Transfer Learning	12,000	9,600;10,800	512	16.2	74.04%	69.13%	71.59%	57.88%	35.11%	41.19%	38.15%	40.99%
Fine Tuning	12,000	9,600;10,800	512	18.3	71.53%	66.71%	69.12%	51.19%	31.24%	38.70%	34.97%	40.35%
Transfer Learning	12,000	4,800;5,400	512	15.9	70.13%	64.90%	67.51%	50.74%	29.10%	36.24%	32.75%	39.75%
Training from Scratch	12,000	4,800;5,400	512	15.7	68.81%	65.25%	67.03%	50.91%	26.34%	36.74%	31.54%	40.23%
Transfer Learning	12,000	4,800;5,400	416	28.6	68.50%	61.24%	64.87%	47.86%	27.47%	34.23%	30.85%	35.91%

TABLE VII
SUMMARY RESULTS.

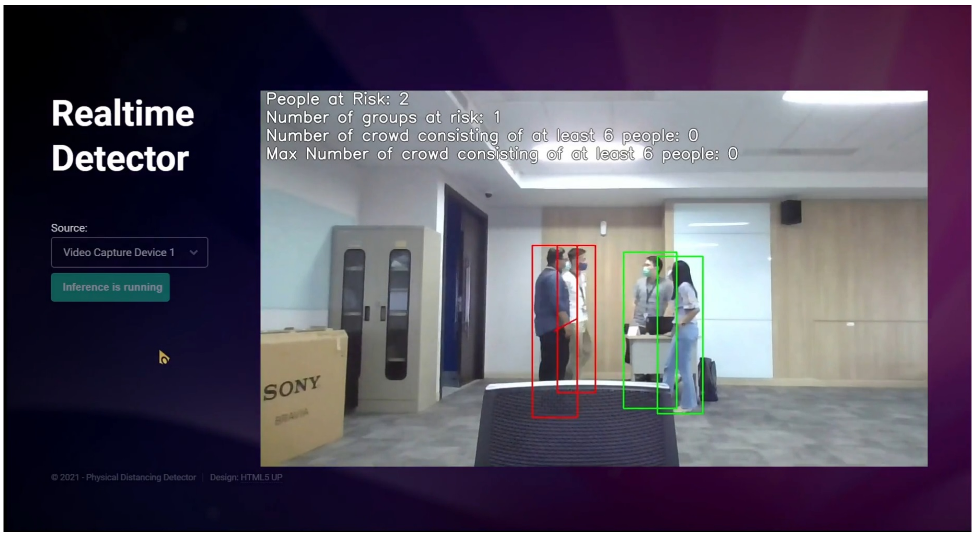
Factor	Speed	Accuracy
Method	YOLOv4 Tiny	YOLOv4 (CSPDarknet53)
Input Size	320	512
Steps	NA	˜
Training Method	NA	Transfer Learning

research also runs an experiment on the YOLOv4 (VGG16) method. However, as it is compared to another method with a similar hyperparameter configuration, the result of this experiment is worse in speed and accuracy. Thus, this method’s experiment is discontinued (see Table V). The highest result of the experiment is on YOLOv4 using the CSPDarknet53 backbone with 74.04% AP50 (71.59% mAP50) and 16 FPS. This experiment uses 12,000 iterations, steps of 9,600 and 10,800, and an input size of 512 pixels. The top five results from the experiment are nominated by method YOLOv4 (CSP-Darknet53), which can be seen in Table VI.

Table VII summarizes the explained experiments, each representing the best options for the related factor. YOLOv4 Tiny, with an input size of 320 pixels, is the fastest method in terms of speed. On the other hand, steps and training methods do not affect the speed. In terms of accuracy, method YOLOv4 with CSPDarknet53 backbone gives the best accuracy. The biggest input size experimented with 512 pixels also gives the best accuracy.

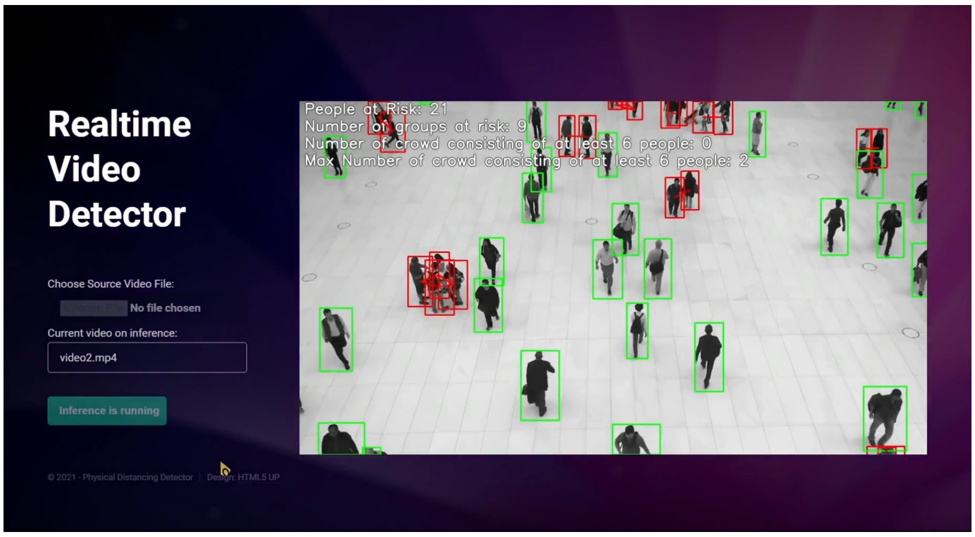
Also, in this experiment, transfer learning provides the highest accuracy for the training method. Steps affect the accuracy, but it has yet to be concluded which gives the optimum result since the experiment shows inconsistent results. The MAP is the average or mean of all AP in all currently trained classes, which are person and head. AP contains Precision, Recall, which involve both True Positive (TP) and False Negative (FN). For example, TP for class person indicates the number of correctly detected people, whilst FN indicates the number of incorrectly undetected people. The results provide a lower AP value than the baseline (77.19% AP50 using RetinaNet) [21]. However, the YOLOv4 framework provides faster training and inference time than the RetinaNet, which is consistent with another similar study [22].

The program proposed in the research uses Python on the Windows platform. It also uses OpenCV 4.5.1 and Darknet framework to load the trained model, threading, Socketio, Argparse, and other libraries to support the process. The program is implemented on the web using the Flask framework. Figures 4 and 5 show some screenshots from the implemented website. This implementation does not include the calibration process to measure the actual distance in pixels of the picture. For simplicity, let the y value represent the distance of 1 meter in pixels. This y value is obtained from a scenario where a video is recorded in a normal handheld camera position and standing, with objects 8.75–8.86 meters away from the camera. From this scenario, it obtains that the distance of 1 meter in such a condition is 160 pixels. This distance only applies to



People at Risk: 2
Number of groups at risk: 1
Number of crowd consisting of at least 6 people: 0
Max Number of crowd consisting of at least 6 people: 0

Fig. 4. Real-time detection from camera.


Realtime Video Detector
Choose Source Video File:
No file chosen
Current video on inference:
video2.mp4
Inference is running
© 2021 - Physical Distancing Detector | Design: HTML5 UP

People at Risk: 21
Number of groups at risk: 9
Number of crowd consisting of at least 6 people: 0
Max Number of crowd consisting of at least 6 people: 2

Fig. 5. Real-time detection from file.

this specific scenario and should not be used for videos with a different condition. This result can be added to future development.

IV. Conclusion

The research aims to explore and implement deep learning using object detection, which is used in physical distancing detectors in Indonesia. Several models have been trained and implemented using a web application to observe the situation and tighten the implementation of physical distancing in their areas (see Fig. 4 and Fig. 5) to answer the problem stated in the introduction. The models are trained and tested on the Crowdhuman dataset. From the results, the YOLOv4 model with CSPDarknet53 has the best and most balanced result in terms of accuracy and speed compared to the other two models for object detection tasks, which produces 71.59% mAP50 (74.04% AP50) with 16–18 FPS on the test machines. These results also answer the problem in the introduction to building real-time object recognition with 16–18 FPS. YOLOv4 provides faster training and inference, which is consistent with the previous research.

This model will be used to calculate the distance and cluster detected people based on the location coordinates provided by the model. However, there is currently no mechanism used in this research to calibrate the actual distance of the video to the distance in the video itself. This issue can be added to future development.

Author Contribution

Writing—original draft, A. K. R. S., and R. G. P.; Methodology, A. K. R. S., A. C., and R. G. P.; Formal analysis, A. K. R. S.; Analysis result review, A. K. R. S., and A. C. All authors have read and agreed to the published version of the manuscript.

References

[1] Z. Allam, "The first 50 days of COVID-19: A detailed chronological timeline and extensive review of literature documenting the pandemic," *Surveying the Covid-19 Pandemic and Its Implications*, vol. 2020, pp. 1–7, 2020.

[2] World Health Organization, "Naming the coronavirus disease (COVID-19) and the virus that causes it," 2020. [Online]. Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/technical-guidance/naming-the-coronavirus-disease-(covid-2019)-and-the-virus-that-causes-it

[3] S. Galea, R. M. Merchant, and N. Lurie, "The mental health consequences of COVID-19 and physical distancing: The need for prevention and early intervention," *JAMA Internal Medicine*, vol. 180, no. 6, pp. 817–818, 2020.

[4] S. Matta, S. Rajpal, K. K. Chopra, and V. K. Arora, "COVID-19 vaccines and new mutant strains impacting the pandemic," *The Indian Journal of Tuberculosis*, vol. 68, no. 2, pp. 171–173, 2021.

[5] R. Fadhli, "Mengenal protokol kesehatan 5M untuk cegah COVID-19," 2023. [Online]. Available: https://tinyurl.com/4cm7v9ek

[6] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," pp. 1–17, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[7] K. B. Chethan, R. Punitha, and Mohana, "YOLOv3 and YOLOv4: Multiple object detection for surveillance applications," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, Aug. 20–22, 2020, pp. 1316–1321.

[8] Z. Zhang, S. Qiao, C. Xie, W. Shen, B. Wang, and A. L. Yuille, "Single-shot object detection with enriched semantic," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, Utah, June 18–22, 2018, pp. 5813–5821.

[9] K. H. Shih, C. T. Chiu, J. A. Lin, and Y. Y. Bu, "Real-time object detection with reduced region proposal network via multi-feature concatenation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 6, pp. 2164–2173, 2020.

[10] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research direction," *SN Computer Science*, vol. 2, pp. 1–21, 2021.

[11] Y. Liu, P. Sun, N. Wergeles, and Y. Shang, "A survey and performance evaluation of deep learning methods for small object detection," *Expert Systems with Applications*, vol. 172, 2021.

[12] J. Redmon, "Darknet: Open source neural networks in C," 2013–2016. [Online]. Available: http://pjreddie.com/darknet/

[13] J. Li and Z. Wu, "The application of YOLOv4 and a new pedestrian clustering algorithm to implement social distance monitoring during the COVID-19 pandemic," *Journal of Physics: Conference Series*, vol. 1865, pp. 1–15, 2021.

[14] D. Yang, E. Yurtsever, V. Renganathan, K. A. Redmill, and Ü. Özgüner, "A vision-based social distancing and critical density detection system for COVID-19," *Sensors*, vol. 21, no. 13, pp. 1–15, 2021.

[15] J. Walsh, O. Kesa, A. Wang, M. Ilas, P. O'Hara, O. Giles, N. Dhir, M. Girolami, and T. Damoulas, "Near real-time social distance estimation in London," *The Computer Journal*, vol. 67, no. 1, pp. 95–109, 2024.

[16] M. Razavi, H. Alikhani, V. Janfaza, B. Sadeghi, and E. Alikhani, "An automatic system to monitor the physical distance and face mask wearing of construction workers in COVID-19 pandemic," *SN Computer Science*, vol. 3, pp. 1–8, 2022.

[17] M. Kamari and Y. Ham, "AI-based risk assessment for construction site disaster preparedness through deep learning-based digital twinning," *Automation in Construction*, vol. 134, 2022.

[18] S. Park, I. C. Michelow, and Y. J. Choe, "Shifting patterns of respiratory virus activity following social distancing measures for coronavirus disease 2019 in South Korea," *The Journal of Infectious Diseases*, vol. 224, no. 11, pp. 1900–1906, 2021.

[19] O. Karaman, A. Alhudhaif, and K. Polat, "Development of smart camera systems based on artificial intelligence network for social distance detection to fight against COVID-19," *Applied Soft Computing*, vol. 110, pp. 1–11, 2021.

[20] M. A. Ansari and D. K. Singh, "Monitoring social distancing through human detection for preventing/reducing COVID spread," *International Jour-*

nal of Information Technology, vol. 13, no. 3, pp. 1255–1264, 2021.

[21] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, and J. Sun, “CrowdHuman: A benchmark for detecting human in a crowd,” 2018. [Online]. Available: https://arxiv.org/abs/1805.00123

[22] L. Tan, T. Huangfu, L. Wu, and W. Chen, “Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification,” *BMC Medical Informatics and Decision Making*, vol. 21, pp. 1–11, 2021.