PERFORMANCE EVALUATION OF YOLOV4 AND YOLOV4- TINY FOR REAL-TIME FACE-MASK DETECTION ON MOBILE DEVICES

Katerina Ntzelepi[1], Michael E. Filippakis[1],
Maria Eleni Poulou[2] and Athanasios Angelakis[3]

[1]Department of Digital Systems, University of Piraeus, Athens, Greece
[2]Department of Mechanical Engineering, University of West Attica, Athens, Greece
[3]JADS, Einhoven University of Technology, Den Bosch, Netherlands

ABSTRACT

The viral outbreak of COVID-19 that started in the year 2019, radically changed our everyday life, with a detrimental impact on the simple, daily habits of citizens. In many countries around the world, the usage of mask is necessary as a protection measure against covid-19. Every service, organization, various stores, schools, universities, hospitals, companies and many other places, which are attended by hundreds of people every day, make the use of a mask necessary to enter them. This fact requires the control of the persons when they enter the respective spaces to determine if they are wearing a mask when entering the area. In this research we compared performance on YOLOv4 and the Tiny-YOLOv4 algorithm on images, recorded video, and real time video. In the next step we will implement the YOLOv4 TFlite and Tiny YOLOv4 TFlite model for mobile applications using the Android Studio platform. On the proposed dataset YOLOv4 achieved 92.91% mAP and training took around 2 hours for 1000 iterations. On the other hand, YOLOv4-tiny achieved 74.75% mAP and training took less than half an hour for 1000 iterations. For further improvement we convert YOLOv4 and YOLOv4-tiny to YOLOv4 TFlite and YOLOv4-tiny TFlite respectively. After this step we compare YOLOv4 TFlite and YOLOv4-tiny TFlite model performance on mobile device. YOLOv4 TFlite achieved 96.92% accuracy on real time video at 5017ms and YOLOv4-tiny 74.72% accuracy on real time video at 491ms.

KEYWORDS

YOLOv4, Tiny-YOLOv4, Mask Detection, Object Detection, Real-time videos, Deep Learning, TensorFlow, TFlite.

1. INTRODUCTION

The epidemic COVID-19 has spread globally since the end of 2019 and still threats the humanity despite the ongoing vaccinations campaigns [1].

It is believed that COVID-19 has started from bats in Wuhan China in November 2019 and spread dramatically fast to 114 countries around the world [2]. The difference between COVID-19 and any other coronavirus from the past is that it can be spread by air and can infect people by contact within a very short time. Also, COVID-19 will be latent in the infected person's body, and symptoms may not outbreak until up to 14 days later. During this period, the patient may not show any symptoms, making the asymptomatic patients impossible to detect and therefore isolate from the rest of the population. People with underlying diseases and over 50 are at greater risk. For them the chances of a possible infection in the respiratory system by the virus causing a long time disability and/or deadly disease are higher. Due to this information, some countries announced COVID-19 restrictions such as national lockdowns, curfews, travel restrictions, closing public places, physical distancing, and closing of borders. In addition governments around the world have obliged their citizens to wear face masks in public places such as: stores, schools, universities and workplaces despite the increase of vaccination rates. These restrictions presented acute challenges in developing countries where weakened infrastructure, overstretched health systems, insufficient funding and limited public health surveillance compromised their potential efficacy [3]. As a consequence of the pandemic the world has faced an economic pressure that has led to social and political challenges. Although statistics has shown that wearing a face mask decreases significantly the transfer of the virus (see [1]) the large-scale restrictions are not easy to implement, adhere to and subsequently difficult to practice and maintain. This lead to an imperfect public compliance, especially if there is a significant impact on social and political norms, economy, and psychological wellbeing of the affected population. Hence it is understandable that the attention is now shifting towards the vaccination of populations after the successful development of several vaccines. However, emerging COVID-19 variants such as porous borders, regular movement of informal traders and sale of fake vaccination certificates continue to threaten progress made towards virus containment in some countries [3].

Due to this facts, the world Health Organization (WHO) has suggested that the use of a medical mask is crucial for the restriction of the spread of the virus. It is common knowledge that prevention is better than cure and in that note one should wear a face mask while meeting people [4]. In nowdays the use of a mask is necessary to enter various places such as schools, supermarkets, shopping malls, cinemas, companie officies, hospitals, gyms etc. If someone fails to wear a mask, they are not legally allowed to enter any indoor place or outdoor crowed places. Also, in many countries when cases of corona virus appear to increase dramatically it is asked by the citizens to wear a mask even if they walk alone in the street.

Because of the necessity of the mask in the prevention of COVID-19 many crowed places as supermarkets, public transports, airports, schools and shopping malls have to supervise the surrounding environment in order to keep the members of it in accordance to the law. This step, though necessary, enhances socials problems.

Nowadays, object detection technology allows us to use camera-enabled devices as a way to perform the face mask wearing detection in a non-contact automatic manner [5]. Especially, deep learning algorithms is the most popular on object detection field. At this moment YOLOv4 (You Only Look Once) is one of the most popular object detection models, because it combines high accuracy and speed. YOLOv4 is faster than a recent state-of-the-art object detection model, EfficientDet, with a similar accuracy when run on similar machines [6]. Also, YOLOv4-tiny [6] was designed to increase speed and decrease computational cost when possible.

In this paper, the authors present one of the most popular object detection algorithms, namely YOLOv4 and YOLOv4-tiny version which is ideal for real time mask detection. In the problem of mask detection on public the speed of the model is one of the most important issues as well as the accuracy of the prediction. Additionally, YOLOv4-tiny is a very lightweight model. With the help of the Tensorflow framework the YOLOv4-tiny can be converted to tflite and therefore become much lighter. With this technique we will develop a mask detection model that can be used on embedded devices or on surveillance cameras.

The rest of the paper is organized as follows. Next section presents the related works based on mask detection or object detection with YOLOv4 and YOLOv4 tiny. Section 3 explains how YOLOv4 and YOLOv4 tiny works. In Section 4 we present the results of object detection algorithms and how stimulates on embedded devices. Section 5 concludes the paper. Finally, section 6 is about future work on the field of mask detection.

2. RELATED WORKS

Akhil Kumar [7] tested tiny YOLO v4 with spatial pyramid pooling (SPP) on a self-created dataset of 52,635 images. In the experiment proposed tiny YOLO v4-SPP network is compared with the original tiny YOLO v4 model. The proposed model achieved mAP (mean average precision) value of 64.31% which was 6.6% higher than tiny YOLO v4. Also, the calculated AP (average precision) of tiny YOLO v4-SPP was 84.42% and 70.37% for original tiny YOLO v4, which is very important in the detection of the presence of a small object like a face mask.

Preeti Nagrath [2] proposed a very lightweight model for face mask detection, so that it is ideal for embedded devices. This technique combined Single Shot Multibox Detector as a face detector and MobilenetV2 architecture as a framework for the classifier. The results showed that the proposed SSDMNV2 methodology had the highest performance with 92.64% accuracy while AlexNet were second with 89.2% accuracy and the LeNet – 5 last with 84.6% accuracy. Also, the results of F1 Score showed that the proposed SSDMNV2 methodology had the highest performance with 0.93 F1 Score while ResNet – 50 had 0.91 F1 Score, VGG-16 had 0.92 F1 Score, AlexNet 0.88 F1 Score and LeNet – 5 last with 0.85 F1 Score. In the last comparison, the results showed that SSDMNV2 model had 15.71 FPS, while LeNet – 5 had 14.55 FPS, ResNet – 50 had 2.89 FPS, VGG -16 had 2.76 FPS and AlexNet – 5 last with 6.31 FPS.

Biparnak Roy [8] performed a comparative study on Moxa3K benchmark dataset, which contains 3000 images with classes 'mask' and 'nomask', using object detection algorithms such as YOLOv3, YOLOv3Tiny, SSD and Faster R-CNN. Biparnak Roy [17] performed a comparative study on Moxa3K benchmark dataset, which contains 3000 images with classes 'mask' and 'nomask', using object detection algorithms such as YOLOv3, YOLOv3Tiny, SSD and Faster R-CNN. The authors tested for detection with tiny YOLO v3 attaining mAP of 56.57% and FPS of 46.5, YOLO v3 detecting with mAP value of 63.99% and FPS of 21.2, SSD attaining mAP of 46.52% and FPS of 67.1, and Faster-RCNN detecting with mAP value of 60.05% and FPS of 14.8.

Sarah Hraybi [1] tested YOLOv4-Tiny, YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x on dataset, which includes images of persons in crowed places with or without wearing medical mask. The results showed that YOLOv4-Tiny achieved mAP of 55.5% and speed of 4.0 ms, whereas YOLOv5s achieved the best speed of 2.1 ms and mAP of 45.3%. It is noticed that YOLOv5x achieves the lowest mAP of 43.7% and the lowest speed of 6.1 ms as well.

Xiaoyu Wang [9] improved the most popular and powerful object detection algorithm YoloV4 by applying CenterLoss and pruning algorithm. This experiment used dataset, which was divided to Real man wears the real mask, Real man no mask, Real man wears the fake mask (By photoshop), Cartoon man wears the mask and the total count of images was 6689 images. After this appliance YoloV4 model achieved 80.75 Average Precision (mAP) and proved that the algorithm works perfectly in different situations and particular in outdoor activities, such as stadiums and college physical education classes.

Elliot Mbunge [3] researched the literature on artificial intelligence models that have been used to detect face masks. In this study the author discovered that deep learning models such as the Inceptionv3 convolutional neural network achiev 99.9% accuracy in detecting COVID-19 face masks, but the problem is that this deep learning models it would be better to apply on real word datasets. Also, in this study it was found that very powerful and popular deep learning models with increased training parameters, such as inception-v4, Mask R-CNN, Faster R-CNN, YOLOv3, Xception, and DenseNet are not yet implemented to detect face masks.

Peishu Wu [10] presented a comparison among FMD-Yolo and other eight advanced face mask detection algorithms on two databases, MD-2 dataset and MD-3 dataset, on five evaluation metrics of mAP, AP50, AP75, mAR and AR50. The experiments on MD-2 database showed that FMD-Yolo model achieved values of 66.4% mAP, 92.0% AP50, 80.0% AP75, 77.4% mAR and 95.5% AR50 and made an improvement of 4.6%, 1.8%, 5.5%, 10.0% and 2.3% respectively. In addition, the experiments on MD-2 database showed that FMD-Yolo model achieved values of 57.5% mAP, 88.4% AP50, 64.3% AP75, 69.9% mAR and 92.9% AR50 and made an improvement of 3.9%, 3.8%, 3.9%, 7.1%, and 0.2%respectively.

A. Kumar et al. [11] performed eight tiny variants of the YOLO algorithm on dataset with 52,635 images for four different class labels namely, with masks, without masks, masks incorrectly, and mask area. On the proposed dataset, original YOLO v4 achieved a mAP value of 71.69 % which was highest among all the original YOLO variants, tiny YOLO v4 achieved a mAP value of 57.71 % which was highest. To improve the performance of all tiny variants of the YOLO algorithm added more convolution layers to the original network and proposed the M-T YOLO v1, M-T YOLO v2, M-T YOLO v3 and M-T YOLO v4 by increasing the value of mAP by 4.12 % for tiny YOLO v3 and 2.54 % for tiny YOLO v4.

Jimin Yu [5] proposed a face mask recognition and standard wear detection algorithm based on the improved YOLO-v4. To increase the performance of YOLO-v4 algorithm improved CSPDarkNet53, reduced the computation and redundancy of the adaptive image scaling algorithm, improved PANet structure and a face mask detection data set is made according to the standard wearing of masks. The experiment results show that mAP of the proposed model of face mask recognition can reach 98.3% and the frame rate is high at 54.57 FPS compared with other popular algorithms of face mask recognition such as YOLO-v4, YOLO-v3, SSD and Faster R-CNN.

Zicong Jiang [12] presented a real-time object detection method, suitable for developing on the mobile and embedded devices, based on YOLOv4-tiny. To improve the original YOLOv4-tiny algorithm used two ResBlock-D modules in ResNet-D, designed an auxiliary residual network block and merged the auxiliary network and backbone network to construct the whole network structure. Compared our proposed method with YOLOv3-tiny and YOLOv4-tiny, our proposed method has the largest FPS, and YOLOv4-tiny has the largest mAP followed by our proposed method. The mAP of our proposed method is 38% and YOLOv4-tiny method is 38.1%. The relative mAP only reduces by 0.26%. The FPS of our proposed method is 294 and YOLOv4-tiny method is 270. The relative FPS increases by 8.9%.

Yuxuan Cai [13] presented YOLObile framework, a real-time object detection on mobile devices with block-punched pruning scheme. Experimental results indicate that our pruning scheme achieves 14× compression rate of YOLOv4 with 49.0 mAP. Under our YOLObile framework, we achieve 17 FPS inference speed using GPU on Samsung Galaxy S20. By incorporating our proposed GPU-CPU collaborative scheme, the inference speed is increased to 19.1 FPS, and outperforms the original YOLOv4 by 5× speedup.

Xinqi Fan [14] proposed single stage face mask detector, named RetinaFaceMask model. The model was developed on PyTorch deep learning framework. The model was trained for 250 epochs with a stochastic gradient descent (SGD) algorithm of learning rate 10^3 and momentum 0.9. An NVIDIA GeForce RTX 2080 Ti GPU was employed. The input image resolution is 840 × 840 for RetinaFaceMask and is 640 × 640 for RetinaFaceMask-Light. The results of the comparison showed that RetinaFaceMask achieved the highest performance on AIZOO dataset with 95% value of APN, 94.6% value of APM and 94.8% value of Map while RetinaFaceMask-Light was second with 95% value of APN, 94.6% value of APM and 94.8% value of Map then follows the RetinaFace with 92.8% value of APN, 93.1%, value of APM and 93% value of Map, YOLOv3 with 92.6% value of APN, 93.7%, value of APM and 93.1% value of mAP and Faster R-CNN with 83.3% value of APN, 83.7%, value of APM and 83.5% value of Map. It is noticed that SSD achieved the lowest performance with 89.6% value of APN, 91.9% value of APM and 90.8% value of Map. Additionally, the results of the comparison on MAFA-FMD showed that RetinaFaceMask achieved the highest performance again with 59.8% value of APN, 89.6% value of APM and 68.3% value of Map while RetinaFaceMask-Light was second with 55.9% value of APN, 88.6% value of APM and 59.8% value of Map then follows the RetinaFace with 58.7% value of APN, 87.4%, value of APM and 66.5% value of Map, YOLOv3 with 61.3% value of APN, 88.9%, value of APM and 66.1% value of mAP and Faster R-CNN with 55.7% value of APN, 86.3%, value of APM and 62.0% value of Map. It is noticed that SSD achieved the lowest performance with 89.6% value of APN, 80.7% value of APM and 46.5% value of Map.

Madhura Inamdar [4] provided a Facemasknet model of 8 layers and compared the proposed model with five popular deep learning networks for mask recognition. Facemasknet achieved the high accuracy with value of 98.6%, ResNet model achieved 74.6% value of accuracy, MobileNetV2 achieved 97.6 % value of accuracy, FSA-Net and HGL method achieved accuracy with value of 74.97% and 75.08% respectively and SRCNet achieved the lowest value of accuracy 98.70 %.

M. Loey et al. [15] presented a hybrid face masks detection model by integrating ResNet-50 classifer with YOLO v2 for medical face mask detection. The authors used ResNet-50 as the feature extractor network and employed the strategy of transfer learning. The feature vector produced by ResNet-50 was passed to the YOLO v2 detection network to perform the detection task. The proposed hybrid model compared with other popular methodologies and achieved the higher value of mAP 81%.

S. S. Pattanshetti et al. [16] present that YOLOv4 is one of the most efficient, flexible, robust, lightweight, and easy to use algorithm for real time object detection. Also, compared with R-CNN, Fast R-CNN, Faster R-CNN etc. Also, YOLOv4 is proved better in terms of speed (AP) and Performance (FPS) compared to previous YOLOv4 algorithms.

Adban Akib Protik [17] proposed a Personal Protective Equipment Detection Method Using YOLOv4 and TensorFlow. The authors used image augmentation techniques on a combined dataset. The weight converted to TensorFlow format and checked live object detection performance and in TensorFlow format authors implemented additional features like live object count and keeping records.

Zuopeng Zhao et al. [18] present a lightweight network SAI-YOLO based on YOLOv4-Tiny to solve the problem of detecting drivers wearing masks in the epidemic period. SAI-YOLO network achieved a mAP of 86% for facemask wearing detection and recognition of 88% for mask-wearing specifications. The average detection time after acceleration on the resource-constrained device Raspberry Pi 4B was 197 ms, which meets the practical application requirements.

Xu Li et al. [19] proposed an improved Yolov4_tiny model that uses divide and conquer ideas for multi-scale target detection, strengthens the model’s detection of small green peppers, and introduces adaptive feature fusion and feature attention mechanisms to identify blocked green peppers and dense green peppers in the detection effect. Experiments showed that the Yolov4_tiny model improves the detection performance while ensuring the detection speed of the lightweight model and reaches the performance standard of the large-scale target detection model.

Pooja Mahto [20] presented object detection algorithm YOLOv4 for vehicle detection, using some pre-existing tweaks and techniques. The YoloV4 model achieved benchmark results with an mAP of 67.7% on the DETRAC-test dataset. Also, YOLOv4 to be definitely faster than the previous iterations. In author's calculations it was able to test around 57K frames in 25 minutes on an average, translating into a frame rate of around 38 FPS, enabling real-time computation on a single GPU.

3. YOLOV4 FOR OBJECT DETECTION

In this chapter we will analyze the proposed model that will be used for the role of mask detection. Object detection algorithms consists of 4 components input, backbone, neck and head [22]. In addition, we will be analyzing a variation of the yolo v4 model, which is contains Bag of Freebies (BoF) and Bag of Specials (BoS).

3.1. YOLOv4 Model Backbone

CSPDarknet. The backbone represents the feature extraction on object detection algorithms. The final choice of backbone for YOLOv4 is CSPDarknet53 [22]. CSPDarknet53 is a convolutional neural network and backbone for object detection that increases higher accuracy compared with to ResNet. CSPDarknet53 employs CSPNet strategy methodology to partition the feature map of the base layer in two parts, one to pass through convolutional layers and the other that would not pass-through convolutional layers and in the end merges them through a cross stage hierarchy. In this way the highest gradient flow through network is achieved [22].

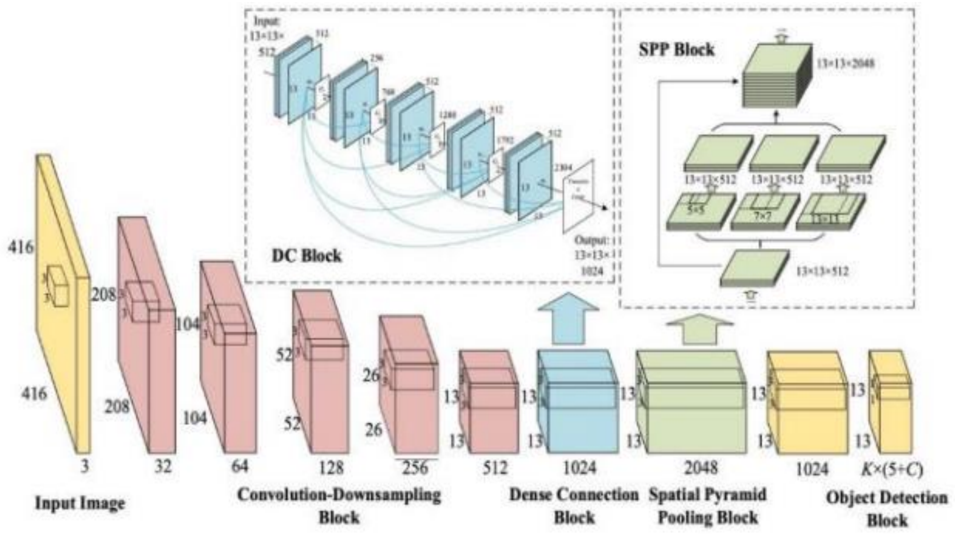


Fig. 1. CSPDarknet53 [16].

3.2. YOLOv4 Model Neck

PANet (Path Aggregation Network). YOLOv3 use FPN [22], which use a top-down path to extract and merge localized spatial information. In FPN the localized information has to cross at least 100 layers, so that the information propagates to the top. On the other hand, PANet creates a bottom-up path, which allows the dissemination of localize information goes through a shorter path. With this shorter path information reaches the top layers with a "shortcut" connection which is about 10 layers. Additionally, PANet uses features from all layers by performing ROI Align operation on each feature map to extract features for the object and is a decision of the network to choose which feature is useful. Uses Fully Convolutional Network (FCN) and Fully Connected Layers to provide more accuracy on mask prediction.

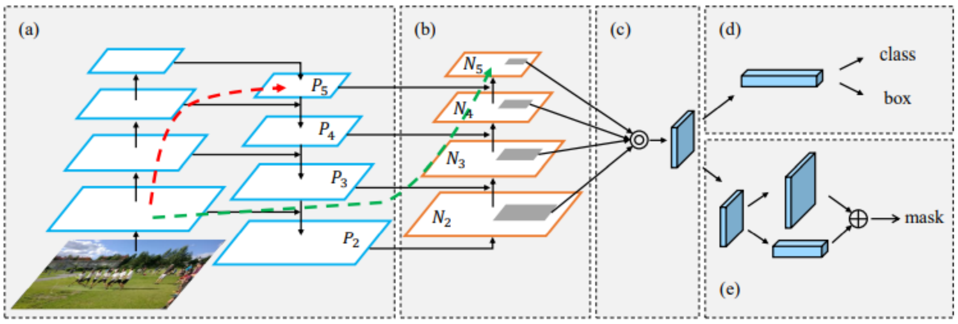


Fig. 2. PANet Architecture [23].

SPP (Spatial Pyramid Pooling Layer). SPP can accept images of different sizes and generate a fixed length output like no FC-layers networks [22] which accept images of different sizes. This design is very important and useful for image segmentation which spatial information is important. In addition, places a spatial pyramid pooling layer in place of the last pooling layer or last convolutional layer. SPP applies multi-level spatial bins . Feature maps are spatially divided into m x m bins, with m being equal to 1,2 or 4. Then to each bin for each channel a maximum pool is applied. In the case of YOLO, a maximum pool is applied to a sliding core of size say, 1 × 1, 5 × 5, 9 × 9, 13 × 13 and at the same time the spatial dimension is maintained. Feature maps from different kernel sizes are then merged as output.

3.3. YOLOv4 Model Head

YOLOv4 deploys YOLOv3 as head for detection with the anchor-based detection steps and three levels of detection granularity. YOLOv4 employs “Bag of Freebies” (BoF) [22] which is a set of techniques or methods that improve the performance and the accuracy of the network without increase the training cost. In YOLOv4 [22], the Bag of Freebies for backbone include a) CutMix and Mosaic data augmentation, b) DropBlock regularization and c) Class label smoothing. In addition, the Bag of Freebies for detector listed below [22] includes CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, Elimination of grid sensitivity, using multiple anchors for a single ground truth, Cosine annealing scheduler, Optimal hyperparameters as well as Random training shapes [22]. YOLOv4 deploys different methods called Bag of Specials (BoS). These strategies increase the inference time by a small amount but increase dramatically the performance and the accuracy of the object detector. The Bag of Specials that YOLOv4 uses for backbone include [22] Mish activation, Cross-stage partial connections (CSP) and Multi-input weighted residual connections (MiWRC). Bag of Specials of detector in YOLOv4 are Mish activation, modified SPP-block, modified SAM-block, modified PAN (path-aggregation) block, DIoU-NMS [22].

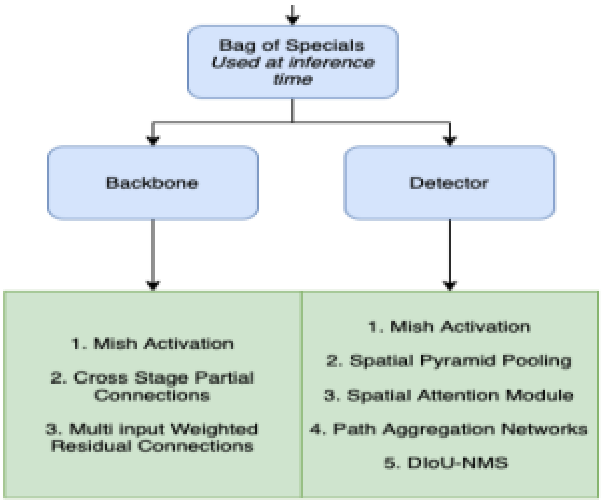

Bag of Specials
Used at inference time
Backbone
Detector
1. Mish Activation
2. Cross Stage Partial Connections
3. Multi input Weighted Residual Connections
1. Mish Activation
2. Spatial Pyramid Pooling
3. Spatial Attention Module
4. Path Aggregation Networks
5. DIoU-NMS

Fig. 3. Bag of Specials [16].

3.4. YOLOv4-tiny

YOLOv4-tiny is an object detection algorithm based on popular YOLOv4 object detector to achieve faster performance on time of object detection. Also, it is ideal for applying an object detection method to embedded systems or mobile devices, as it is a lightweight model with high performance in terms of speed. [22]

YOLOv4-tiny algorithm as a backbone method uses the CSPDarknet53-tiny network instead of the CSPDarknet53 that YOLOv4 algorithm uses. The CSPDarknet53-tiny network uses the CSPBlock module in cross stage partial network [12]. With cross-stage partial connections structure, the CSPBlock divides the input feature map into two parts and combines the two parts in the cross-stage residual edge. This makes possible the gradient flow travel on two different network paths. In the meantime, the CSPBlock can significantly reduce the computational complexity by 10–20% while ensuring the detection accuracy of the network [24]. CSPDarknet53-tiny network uses the LeakyRelu as activation function without using the extra Mish activation function that used CSPDarknet53 in YOLOv4. As a result, the computation process become much faster and simpler. To sum up, YOLOv4-tiny utilizes the LeakyRelu function as an activation function, which is defined by:

$$y_i = \begin{cases} x_i, x_i \geq 0 \\ \dfrac{x_i}{a_i}, x_i < 0 \end{cases} \qquad (1)$$

Where a_i is a constant parameter larger than 1. [12] YOLOv4 tiny uses feature pyramid network to extract feature maps with different scales to achieve a better time performance instead of YOLOv4 algorithm that uses Mish activation function and spatial pyramid pooling and path aggregation. Meanwhile, YOLOv4 tiny model uses two different scales feature maps that are 13×13 and 26×26 to predict the detection results [12].

The main differences in architecture of YOLOv4 tiny compared to architecture of YOLOv4 are the following. Firstly, YOLOv4 tiny It has only two YOLO heads as opposed to three in YOLOv4, and it has been trained from 29 pre-trained convolutional layers as opposed to YOLOv4 which has been trained from 137 pre-trained convolutional layers. Also, the initial difference between YOLOv4 tiny and YOLOv4 is that the network size is dramatically reduced. The number of convolutional layers in the CSP backbone are compressed. The number of YOLO layers are two instead of three and there are fewer anchor boxes for prediction.

4. EQUATIONS

The goal of this study was to compare YOLOv4 and YOLOv4-tiny in terms of accuracy, speed and memory usage on images, recorded video, and real-time video. After evaluating which YOLOv4 model had the best performance, the YOLOv4 and YOLOv4-tiny weights converted to tflite and in the end of the procedure YOLOv4 tflite and YOLOv4-tiny tfliteon android for mask detection. Also, YOLOv4 tflite compared with YOLOv4-tiny tflite in terms of accuracy and speed.

4.1. Experimental Dataset Analysis

In this paper, we used “Labeled Mask Dataset (YOLO_darknet)” dataset in Kaggle by tech zizou . This dataset contains 1510 images belonging to the 2 classeswith_mask and without_mask, as well as their bounding boxes in the YOLO format labeled text files. The 90% (1359 images) used for training and the 10% (images) used in testing.

4.2. Training Details

The experiments were performed using the Darknet repository written in C language. The CPU is AMD Ryzen 7 3700X 8-Core. The computation resources are 64 GB of RAM and an NVIDIA GeForce GTX 1660 GPU. In order to make full use of the GPU to accelerate the network training, the CUDA 11.0 and its matching CuDNN are installed in the system. The batch size, epoch, learning rate, momentum and decay are 64, 273, 0.001, 0.973 and 0.0005 for all methods, respectively.

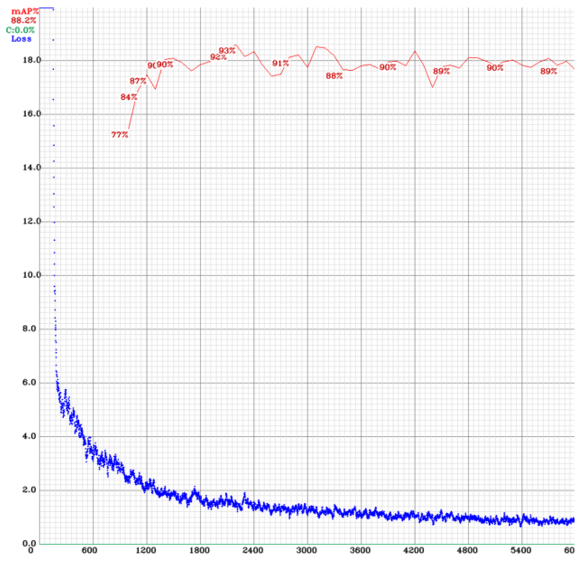


Fig. 4. The average loss and mAP graph YOLOv4.

After training YOLOv4 model a graph chart was generated as shown in Fig. 4. Average loss and mAP values which are used to perform the evaluation of the model changes a lot of times during the training of the model and prove how accurate is the model. Dataset started from 0 iteration and continued until 6000 iterations. At 1000 iterations first mAP of 77.24% was calculated. It is shown in Fig. 4.2.1, that the mAP went above 90% at around 2000 iterations. During this training the values of average loss and mAP were fluctuating from time to time. The lowest average loss of approximately 0.8 was calculated at around 2200 iterations. The best and highest mAP was calculated is 92.91%. After the end of 6000 iterations the average loss was at approximately 0.8.

After training YOLOv4-tiny model a graph chart was generated as shown in Fig. 5. Average loss and mAP values which are used to perform the evaluation of the model changes a lot of times during the training of the model and prove how accurate is the model. Dataset started from 0 iteration and continued until 6000 iterations. At 1000 iterations first mAP of 38.85% was calculated. It is shown in Fig. 4.2.2, that the mAP went above 70% at around 3000 iterations. During this training the values of average loss and mAP were fluctuating from time to time. The lowest average loss of approximately 1.3 was calculated at around 3600 iterations. The best and highest mAP was calculated is 74.75%. After the end of 6000 iterations the average loss was at approximately 1.34.

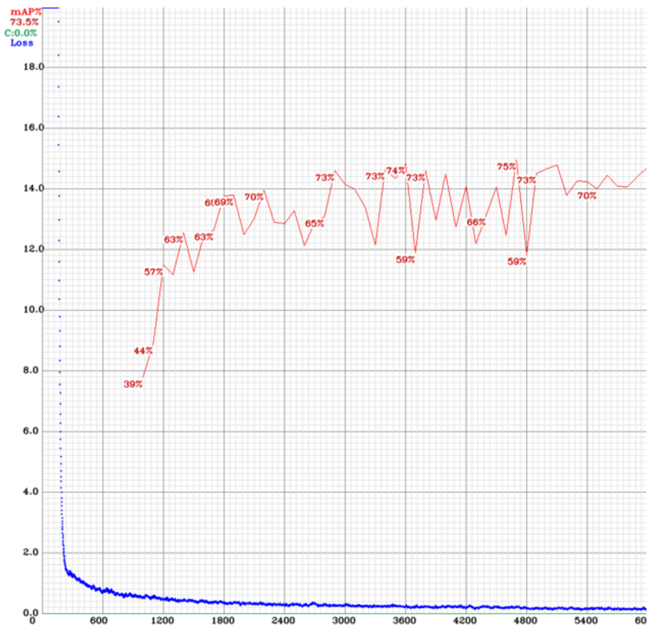


Fig. 5. The average loss and mAP graph YOLOv4-tiny.

5. Performance Evaluation

In this section we analyze the performance evaluation of the two selected in both an offline (i.e., training and predict in desktop) as well as online (i.e., running on a mobile device).

5.1. Offline Evaluation (Desktop)

Table 1 shows the training time per 1000 iterations and the weights of the training procedure of YOLOv4 models. YOLOv4-tiny took less than half an hour for 1000 iterations. YOLOv4-tiny uses Leaky Relu activation function instead of YOLOv4 which uses Mish activation function. As a result, YOLOv4-tiny takes less time in training due to less computational complexity. In total, it took around 2 h to train YOLOv4-tiny completely, which is a remarkably short time. On the other hand, YOLOv4 had a different training time of 2.21 h for 1000 iterations; it took around 13h for the model to be trained completely.

Table 1. Details regarding training YOLOv4 models.

Model Name	Average Trainning Time per 1000 iterations (h)	Weights size (KB)
YOLOv4-tiny	0.33	22.980
YOLOv4	2.12	250.037

To compare the performance of the YOLOv4 and YOLOv4-tiny network we trained and tested both models on the face masks detection dataset and evaluated for performance metrics. The comparative results for performance metrics achieved by YOLOv4 and YOLOv4- tiny network and their counterparts are shown in Table 2 and Fig. 6.

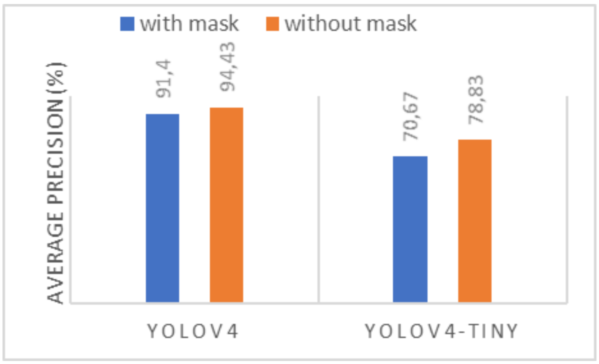


Fig. 6. Comparative values for average precision.

Table 2. Performance metrics of the YOLOv4 models.

Model	mAP	Recall	Precision	F1-score
YOLOv4	92.91%	22.980	0.86	0.84
YOLOv4-tiny	74.75%	250.037	0.96	0.71

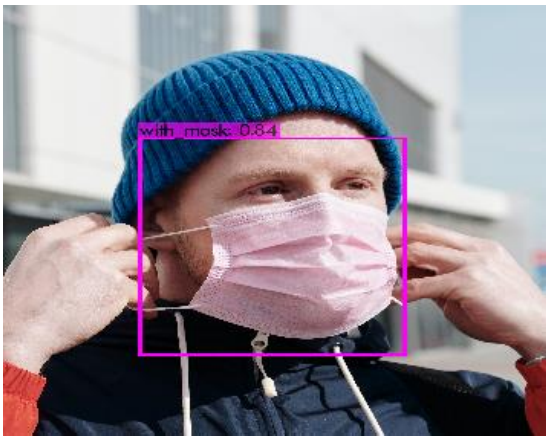


Fig. 7. YOLOv4 on image.

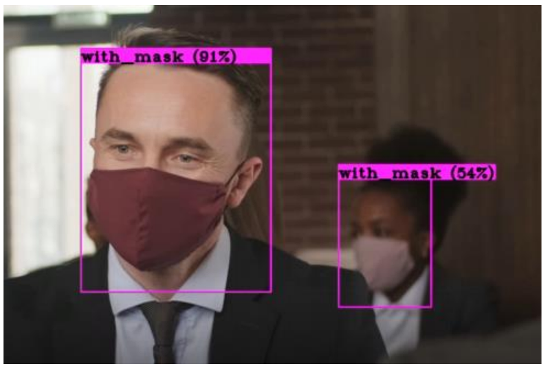


Fig. 8. YOLOv4 on video

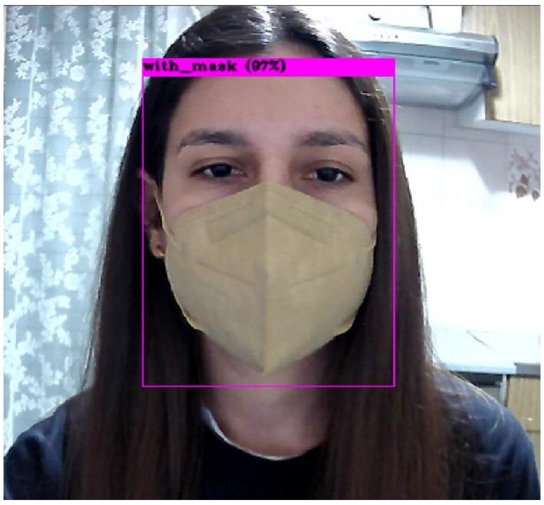


Fig. 9. YOLOv4 on real-time video.

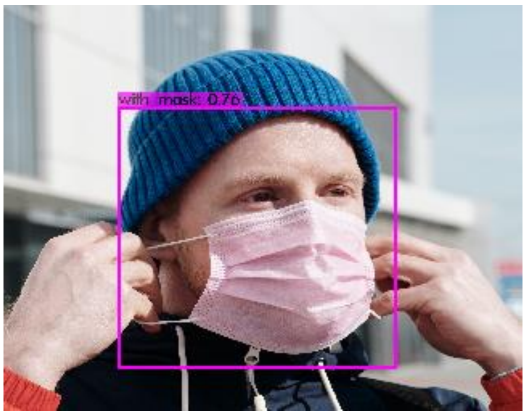


Fig. 10. YOLOv4-tiny on image.

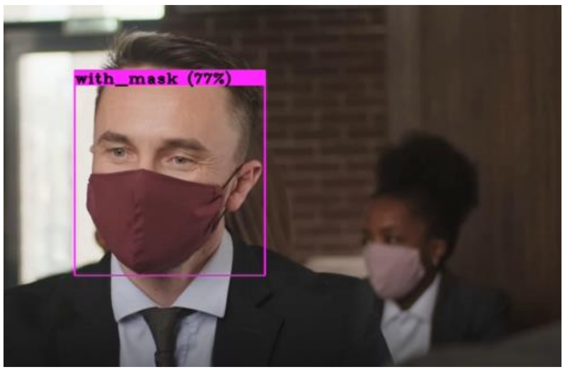


Fig. 11. YOLOv4-tiny on video.

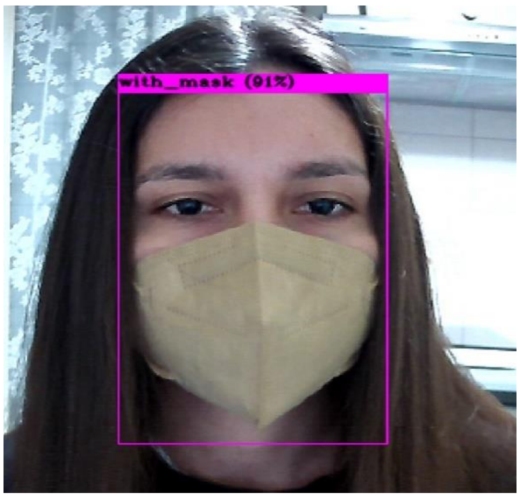


Fig. 12. YOLOv4-tiny on real-time video.

5.2. Online Evaluation (Mobile Device)

This section of the paper is dedicated to testing the model's performance on mobile devices. This can be achieved by converting the YOLOv4 and YOLOv4-tiny object detection models to TensorFlow Lite (tflite) for deploying them on mobile device. TensorFlow Lite is an open-source deep learning framework that transforms a pre-trained model in TensorFlow to a special format that can be optimized for speed and storage. This special format is ideal for deployment on embedded devices like mobile phones using Android or iOS.

TensorFlow Lite include tools used for machine learning on edge devices such as smartphones and IoT. TensorFlow Lite is the TensorFlow framework designed to visualize inference on small devices, meant to avoid a round-trip data computation to and from the server capability of real-time detection and work without internet connection.[25]

After the model is trained and converted to TensorFlow format, it will be converted to the TensorFlow Lite version. The TensorFlow Lite version achieves satisfactory accuracy while at the same time occupying less space. These properties make the TF Lite model suitable for deployment on mobile and embedded devices.

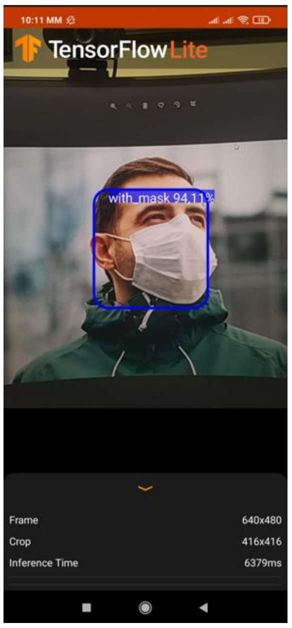
The next step is to optimize the TF Lite model. The purpose of optimization is to reduce the size of the model, but there is a trade-off between the size of the model. Edge models should be lightweight and have low latency to run inferences. This can be achieved by reducing the amount of computation required for model prediction.

TensorFlow Lite achieves optimization using quantization. The activation function, weights, and biases that are stored when the model is saved on TensorFlow format are 32-bit floating points. Quantification reduces the precision of the numbers used to represent different parameters of the TensorFlow model and thus the model becomes light weight.
Quantization can be applied to weight and activations.

- Weights with 32-bit floating points can be converted to 16-bit floating points or 8-bit floating points or integer and will reduce the size of the Model.
- Both weights and activations can be quantized by converting to an integer, and this will give low latency, smaller size, and reduced power consumption.

To test the performance of YOLOv4 and YOLOv4-tiny on difference devices, we simulate the model on smartphone which is an embedded device. The smartphone using android and is a Xiaomi Redmi note 7. We used the weights from the training of YOLOv4 and YOLOv4-tiny models for mask detections mentioned above . Firstly, we converted the weight files of YOLOv4 and YOLOv4-tiny model respectively into TensorFlow format and then save the TensorFlow models for tflite converting. In the end, we quantize both models to float16 format.

Table 3 shows the accuracy, the weights and the inference time of TFlite YOLOv4 models on the mobile device. TFlite-YOLOv4 achieved accuracy of 94.11% at inference time of 6379 ms. In contrast, TFlite-YOLOv4- tiny achieved 78.59% accuracy , which is acceptable, at much better inference time from TFlite-YOLOv4 of 554 ms. Additionally, TFlite-YOLOv4-tiny is more light-weight model with 11.548 KB weights file compared with TFlite YOLOv4, which has 125.163 KB weights file.



Frame 640x480
Crop 416x416
Inference Time 6379ms

Fig. 13. YOLOv4 on mobile device.

Table 3. Performance metrics of TFlite YOLOv4 mode.

TFlite	Accuracy (%)	Weights size (KB)	Inference Time (ms)
YOLOv4	92.91%	22.980	0.86
YOLOv4-tiny	74.75%	250.037	0.96

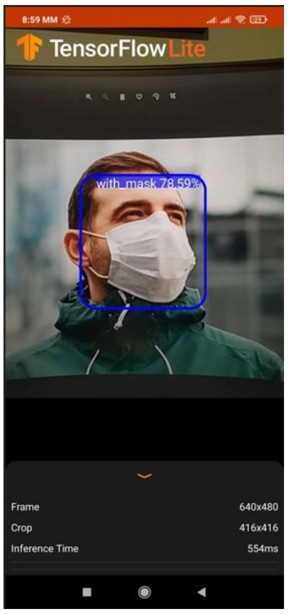
6. DISCUSSION

In this study the following limitations have appeared The training of the two proposed models, YOLOv4 and the YOLOv4-tiny requires a computer system with high capabilities and in particular with a high-performance GPU. Additionally, the training of the models is a time-consuming process, which is restrictive in case we want to test and possibly compare more models together. It is also challenging if we want to follow the same procedure for different and larger data sets. Although we use one of the fastest algorithms in real time object detection, namely YOLOv4 and Tiny-YOLOv4, the performance of time as a function of accuracy while being satisfactory to some extent has room for improvement. Finally, Tiny-YOLOv4 offers better performance on time but lacks in accuracy.

7. CONCLUSION AND FUTURE WORK

In this paper we selected two deep neural network models for the task of object detection. In particular, we wanted to know if a face mask appears on an image or a frame from a recorded or a real time video. The two selected models are a variation of the same model family: the YOLOv4 and the YOLOv4-tiny. After performing experiments on a cellphone and on a desktop personal computer, the results indicated that both models in both cases remain effective in accuracy and in time. To be more precise the model type YOLOv4-tiny is better suited for real-time video and embedded devices because of its time performance in detecting objects while model YOLOv4 is more suitable in cases where higher levels of accuracy are required.

In future applications, we will be improving the performance of real-time mask detection using the most recent version of YOLO algorithm, YOLOv5. YOLOv5 when implemented in Ultralytics PyTorch Framework may attain higher FPS. In addition, another improvement would be the detection of the correct use of face mask, this is because a lot of people may



Frame 640x480
Crop 416x416
Inference Time 554ms

Fig. 14. YOLOv4-tiny on mobile device.

wear a mask, but in the wrong way thus reducing its effectiveness. Finally, we will implement our models in surveillance cameras on airports, companies, malls etc.

REFERENCES

[1] M. R. Sarah Hraybi, "Examining YOLO for real-time face-mask detection," p. 6, 2021.
[2] R. J. a. A. M. a. R. A. a. P. K. a. J. H. b. Preeti Nagrath, "SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2," elsevier, p. 13, 2020.
[3] 3. 10. S. S. S. G. F. B. A. A. S. M. Elliot Mbunge, "Application of deep learning and machine learning models to detect COVID-19 face masks - A review," KeAi, p. 11, 2021.
[4] A. K. A. S. M. K. Akhil Kumar, "A hybrid tiny YOLO v4 SPP module based improved face mask detection vision system," Journal of Ambient Intelligence and Humanized Computing, p. 14, 2021.
[5] J. Y. a. W. Zhang, "Face Mask Wearing Detection Algorithm Based on Improved YOLO-v4," Sensors, p. 21, 2021.
[6] A. I. B. P. a. T. Ahamed, "Real Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT," Sensors, p. 32, 2021.
[7] A. K. A. S. M. K. Akhil Kumar, "A hybrid tiny YOLO v4 SPP module based improved face mask detection vision system," Journal of Ambient Intelligence and Humanized Computing, p. 14, 2021.
[8] S. N. D. G. D. D. P. B. T. D. Biparnak Roy, "MOXA: A Deep Learning Based Unmanned Approach For Real Time Monitoring of People Wearing Medical Masks," Springer, p. 10, 2020.
[9] Z. C. B. W. M. L. Xiaoyu Wang, "Application of Pruning Yolo-V4 with Center Loss in Mask Wearing Recognition for Gymnasiums and Sports Grounds of Colleges and Universities," IEEE 6th International Conference on Computer and Communications, p. 5, 2020.
[10] H. L. N. Z. F. L. Peishu Wu, "FMD-Yolo: An efficient face mask detection method for COVID-19 prevention and control in public," elsevier, p. 10, 2022.
[11] A. K. K. V. A. S. M. K. Akhil Kumar, "Scaling up face masks detection with YOLO on a novel dataset," elsevier, p. 15, 2021.
[12] L. Z. S. L. Y. J. ZICONG JIANG, "Real-time object detection method for embedded devices," p. 11.
[13] H. L. G. Y. W. N. Y. L. X. T. B. R. Y. W. Yuxuan Cai, "YOLObile: Real-Time Object Detection on Mobile Devices via Compression-Compilation Co-Design," p. 10, 2020.
[14] M. J. Xinqi Fan, "RetinaFaceMask: A Single Stage Face Mask Detector for Assisting Control of the COVID-19 Pandemic," p. 6, 2021.
[15] G. M. M. H. N. T. N. E. M. K. Mohamed Loey, "Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection," elsevier, p. 8, 2020.

[16] S. I. N. Shraddha Sanjeev Pattanshetti, "Real-Time Object Detection with Pre-eminent Speed and Precision using YOLOv4," International Journal of Research in Engineering, Science and Management, vol. 4, no. 7, p. 6, 2021.
[17] A. H. R. S. S. Adban Akib Protik, "Real-time Personal Protective Equipment (PPE) Detection Using YOLOv4 and TensorFlow," p. 7, 2021.
[18] 1.K. H. ,. M. X. L. ,. T. Z. ,. X. a. S. C. Zuopeng Zhao, "SAI-YOLO: A Lightweight Network for Real-Time Detection of Driver Mask-Wearing Specification on Resource-Constrained Devices," Hindawi, vol. 2021, p. 15, 2021.
[19] J. P. F. X. J. Z. Q. L. X. H,.L., W. Xu Li, "Fast and accurate green pepper detection in complex backgrounds via an improved Yolov4-tiny model," elsevier, p. 10, 2021.
[20] P. G. P. S. a. J. P. Pooja Mahto, "REFINING YOLOV4 FOR VEHICLE DETECTION," IJARET, vol. 11, no. 5, pp. 409-419, 2020.
[21] R.-C. C. Y.-T. L. J. K. D. H. CHRISTINE DEWI, "Yolo V4 for Advanced Traffic Sign Recognition With Synthetic Training Data Generated by Various GAN," IEEEAceess, p. 15, 2021.
[22] C.-Y. W. H.-Y. M. L. Alexey Bochkovskiy, "YOLOv4: Optimal Speed and Accuracy of Object Detection," p. 17, 2020.
[23] J. S. J. J. Shu Liu† Lu Qi† Haifang Qin, "Path Aggregation Network for Instance Segmentation," p. 11, 2018.
[24] C. D,. Z. J,.M. G. a. Z. H. Han Wu, "SORT-YM: An Algorithm of Multi-Object Tracking with YOLOv4-Tiny and Motion Prediction," p. 20, 2021.
[25] R. K. M. K. N. K. C. P. Sharjeel Anjum, "A Pull-Reporting Approach for Floor Opening Detection Using Deep-Learning on Embedded Devices," p. 9, 2021.