The Journal of Supercomputing (2023) 79:8966–8992
https://doi.org/10.1007/s11227-022-04979-2

Integration of improved YOLOv5 for face mask detector and auto-labeling to generate dataset for fighting against COVID-19

Thi-Ngot Pham[1,2] · Viet-Hoan Nguyen[3,4] · Jun-Ho Huh[2,5]

Accepted: 23 November 2022 / Published online: 3 January 2023

Abstract
One of the most effective deterrent methods is using face masks to prevent the spread of the virus during the COVID-19 pandemic. Deep learning face mask detection networks have been implemented into COVID-19 monitoring systems to provide effective supervision for public areas. However, previous works have limitations: the challenge of real-time performance (i.e., fast inference and low accuracy) and training datasets. The current study aims to propose a comprehensive solution by creating a new face mask dataset and improving the YOLOv5 baseline to balance accuracy and detection time. Particularly, we improve YOLOv5 by adding coordinate attention (CA) module into the baseline backbone following two different schemes, namely YOLOv5s-CA and YOLOV5s-C3CA. In detail, we train three models with a Kaggle dataset of 853 images consisting of three categories: without a mask "NM," with mask "M," and incorrectly worn mask "IWM" classes. The experimental results show that our modified YOLOv5 with CA module achieves the highest accuracy mAP@0.5 of 93.9% compared with 87% of baseline and detection time per image of 8.0 ms (125 FPS). In addition, we build an integrated system of improved YOLOv5-CA and auto-labeling module to create a new face mask dataset of 7110 images with more than 3500 labels for three categories from YouTube videos. Our proposed YOLOv5-CA and the state-of-the-art detection models (i.e., YOLOX, YOLOv6, and YOLOv7) are trained on our 7110 images dataset. In our dataset, the YOLOv5-CA performance enhances with mAP@0.5 of 96.8%. The results indicate the enhancement of the improved YOLOv5-CA model compared with several state-of-the-art works.

Keywords COVID-19 · Deep learning · Face mask detection · Auto-labeling · You Only Look One · YOLO · YOLOv5 · Coordinate attention

✉ Jun-Ho Huh
72networks@pukyong.ac.kr; 72networks@kmou.ac.kr

Extended author information available on the last page of the article

1 Introduction

According to the World Health Organization (WHO), the outbreak of COVID-19 accounted for over 278 million infections and just under 5.4 million deaths globally by the end of December 2021 [1]. Over two years since the beginning of the pandemic, even though vaccinations have boosted our immune system to fight against COVID-19, COVID is not fully over and still affects our lives negatively. To prevent the spread of the virus, one of the most effective deterrent methods is using face masks [2, 3]. Recently, wearing a face mask is mandatory in public places such as hospitals, and schools. However, a great number of people ignore or are misled by wearing it incorrectly, resulting in increasing infection cases and placing more caseloads on the public healthcare system.

Face mask detection based on deep learning object detection methods plays an important role in the fight against COVID-19, which achieved good results with high accuracy. From the comprehensive review of facemask detection techniques [4, 5], there are several deep learning-based algorithms, including You Only Look One (YOLO) [6], Single Shot Detector (SSD) [7], RetinaFace [8], and (Faster Recurrent Convolutional Neural Network) Faster R-CNN [9]. The previous study focused on the high accuracy of face mask detection using two-stage detection models (i.e., Faster R-CNN), while a one-stage detector (i.e., YOLO) achieved fast inference time but lower accuracy. We propose an improved (You Only Look One version 5) YOLOv5 [10] by adding coordinate attention (CA) [11] module to achieve better accuracy results with fast inference time.

On the other hand, the performance of the deep learning object detection models is heavily dependent on training datasets. Particularly, these models require a huge amount of data to accomplish their optimal performance, but most open-source datasets are small-scale with class imbalance [12–15]. For instance, the Kaggle dataset [14] includes three categories with a mask, without a mask, and incorrectly worn a mask, which is abbreviated as "M," "NM" and "IWM", respectively. The number of images of each class in this dataset is unbalanced, as the number of labels for "incorrectly masked face" (IWM) is smaller significantly than the number of the others two classes. To solve these limitations, we will create a new face mask image dataset by collecting data from different open sources and generating annotations (i.e., drawing bounding boxes and classifying categories) automatically.

In this paper, we propose a comprehensive solution encompassing creating new data by integrating our improved YOLOv5s face mask detector with an auto-labeling module. The overall technical contributions of this research are presented as follows:

- A modified YOLOv5s-baseline by adding the coordinate attention (CA) module into the YOLOv5 backbone following two different schemes called YOLOv5s-CA and YOLOv5s-C3CA.
- An integrated system of the auto-labeling module and improved YOLOv5-CA, which automatically generate the labeled images from different open sources

(i.e., YouTube videos) to solve the limitation of the dataset and reduce manual annotation work.

- We utilize our system with a demo application to create a new real face mask dataset with three class balances: “M”, “NM” and “IWM” classes. Then, our proposed YOLOv5-CA model is trained on our dataset to enhance its detection accuracy and compared with YOLOX, YOLOv6, and the latest YOLOv7.

2 Related research

2.1 Face mask dataset

In literature reviews, many face mask datasets have been proposed over the last two years since the COVID-19 outbreak. However, there are only a few datasets with annotations and open access as described in Table 1. In terms of four-listed datasets, three datasets consisting of Kaggle [14], FMLD [15], and PWMFD [16] are annotated with PASCAL VOC files, consisting of the information of bounding boxes and classes that can be trained on the deep learning object detection You Only Look Once (YOLO) training platform. Otherwise, the UFMD dataset [17] is annotated as UFMD’s images are separated into different folders for each class. UFMD dataset can be used to train with algorithms like Faster R-CNN, but impossible to train directly with the YOLO training platform without PASCAL VOC files. Besides, the class imbalance problem exists in the face mask datasets; for example, the ratios of “incorrect worn mask” (IWM) are smaller than these other two categories, implying that class distribution is extremely imbalanced. In the case of training models with an imbalanced dataset, it will increase the probability of the wrong classification. It is understandable that high ratios of classes feasibly learn better than low ratios of classes [5]. Alternatively, collecting images of the IWM class from available datasets is a complex task.

In this paper, we propose an auto-labeling method to draw bounding boxes and classify classes automatically to generate a labeled image dataset from available datasets such as UFMD and different open sources (i.e., YouTube videos) to reduce labor. Moreover, we deploy our integration system into a demo GUI application to

Table 1 Annotation of face mask datasets

Dataset Name	Main Characteristics	Image Number	Categories		
			NM	M	IWM
Kaggle [14]	Internet	853	717	3232	123
FMLD [15]	MAFA, Wider Face	41,934	32,912	29,532	1528
PWMFD [16]	WIDER Face, MAFA, RWMFD	9205	10,471	7695	366
UFMD [17]	FFHD, CelebA, LFW, YouTube videos and Internet	21,316	10,698	10,618	500

create a new real large images dataset which is a more balanced class distribution than previously available datasets.

2.2 YOLO-based face mask detection

Table 2 presents the related works of YOLO-based face mask detection. The previous studies provided various insights into our research, but there are still several problems to be solved along with room for improvement. The YOLO-based series, such as YOLO-Fastest [18], YOLOv2 [19], YOLOv3 [20], and YOLOv4 [21], and their improved variations from baseline, are quite out of date compared to YOLOv5 [10]. In Ding et al. [22], the authors build face mask detection models based on YOLOv5-baseline with the fifth update (v5.0). In this work, we improve the 6th update version (v6.0) of YOLOv5-baseline, which has recently released on Feb-2022 with more improvements and changes. Besides, adding attention mechanisms into deep learning models for image classification has proven to be helpful [23]. In [24], the authors improved the YOLOv3 baseline by adding a Squeeze-and-Excitation (SE) attention network (2019) [25]. However, the authors of CA (2021) [11] state that CA with the lightweight property performs much better than other attention methods (e.g., SENet) with MobileNet [26] model implementation. In this work, we propose the improved YOLOv5 by adding a CA module to increase accuracy while utilizing the lightweight architecture of CA to keep the fast inference time.

3 Proposed methods

3.1 Improved YOLOv5 models for face mask detection

3.1.1 YOLOv5-baseline network structure

Among different versions of deep learning object detection—You Only Look Once model (YOLO), the most controversial version is YOLO version 5 (YOLOv5) [10] which shared the same architecture and achieved similar performance as the most well-known YOLOv4 [21]. At the time of its debut, the biggest difference between the two versions is that YOLOv5 is implemented on the PyTorch platform [32], which could be used in many development environments, instead of the YOLOv4-Darknet framework [33] and the newly introduced focus layer in the CSPDarknet53 backbone. In addition, by taking advantage of the PyTorch framework, the advantages of YOLOv5 models were a significantly smaller size, faster training time, and more accessibility to deployment in real-world applications. Briefly, YOLOv5 follows the same architecture of an object detection network, which consists of four parts including input, backbone, neck, and head.

Initially, YOLOv5 utilizes CSPDarknet53 as the backbone with the focus, bottleneck, bottleneck CSP, and SPP layer. The backbone module is mainly composed of bottlenecks to reduce and expand the number of channels. The purpose of this

Table 2 Face mask detection based on one-stage detection algorithms

Ref	Dataset used	Algorithms	Contributions
Ding et al. [22]	Custom dataset with a total of 14,233 labels for 3 classes	YOLOv5SSD	Comparison of original SSD and YOLOv5 on real video data. SSD method achieved 99.97% accuracy and 6 fps while YOLO achieved mAP of 89% and 52 fps
Jiang et al. [23]	Properly-wearing masked face detection dataset (PWMFD)3 categories	SE-YOLOv3	Masked face detection introduced 'Squeeze-and-Excitation' (SE) as an attention mechanism into Darknet-53 to extract the essential feature, and adopted GIoUloss, focal loss to enhance stability and robustness, which achieved a 73.4% mAP
Wang et al. [27]	MAFA: 4065 images, WIDER FACE: 3894 images, Internet: 1138 images. Total: 9097 images with 17,532 labels	YOLO-Fastest	A server-less edge-computing face mask detection application was utilized based on JavaScript, NCNN, and WASM with an 89% accuracy mAP and 20.36 fps on a Macbook Pro 15
Loey et al. [28]	MMD: 682 images. FMD: 853 images. Total: 1415 images	YOLOv2Resnet-50	A medical face mask detection model was developed based on transfer learning with a pre-trained Resnet50 for feature extraction and YOLOv2 for face mask detection and achieved an 81% AP
Prusty et al. [29]	Kaggle 800 images: 90% training/10% testing	YOLOv3	Grayscale transformation and Gaussian blur data augmentation techniques were applied to expand the original dataset size. Masked face detection was developed with YOLOv3 on a new data augmentation dataset, which achieved a 93% mAP
Kumar et al. [30]	Self-built dataset with 52,635 augmented images from the original 11,000 images with about 50,000 labelst	Modified Tiny-YOLOv4	Eight variants of the YOLO algorithm were tested. Tiny-YOLOv4 achieved an mAP of 57.71%. A new architecture of Tiny-YOLOv4 with modifications in feature extraction networks improved mAP by 2.54% for the original tiny-YOLOv4
Yu et al. [31]	RMFD, MaskedFace-Net. Total: 10.855 image	Modified YOLOv4	YOLOv4 baseline backbone CSPDarkNet53 was modified to reduce computation costs and improve learning ability with an adaptive image scalation method to lower redundancy. And original neck structure was improved to learn more semantic information, which achieved 98,3% mAP with 54.57 fps

was to first reduce the channel by half through 1×1 convolution and then double the number of channels through 3×3 convolution to obtain features, with the number of channels of input and output kept unchanged. The YOLOv5 neck follows the PANet structure (path aggregation network). The feature extractor of the module adopted a new enhanced bottom-up path called an FPN structure (feature pyramid networks), with the propagation of low-level features being improved. Until the end of 2021, there have been six updated versions of the original YOLOv5, and the latest version is the sixth version called YOLOv5-6. It should be noted that we worked on the sixth version of YOLOv5-6 which is also called YOLOv5-baseline. YOLOv5-6 has various significant improvements over the initial version. The model architecture of the sixth update has some modifications as shown in Fig. 1, including the replacement of focus with an equivalent convolution layer for improved exportability, while a new SPPF instead of SPP layer for reduced ops and reduction in the P3 backbone layer C3 repeats from 9 to 6 for improved speed and reorders places SPPF at the end of the backbone. Following different network depths and widths, the YOLOv5 baseline can be categorized into four types, namely YOLOv5n, YOLOv5s, YOLOv5m, and YOLOv5l. Also, the volume of weight and the number of parameters of these models increase, respectively. YOLOv5n was first introduced in the sixth update version.

3.1.2 Improvement of YOLOv5s with coordinate attention (CA) module

To deploy face mask detection models into real scenarios (e.g., main-gate monitoring system), this paper focuses on improving the architecture of YOLOv5s backbone

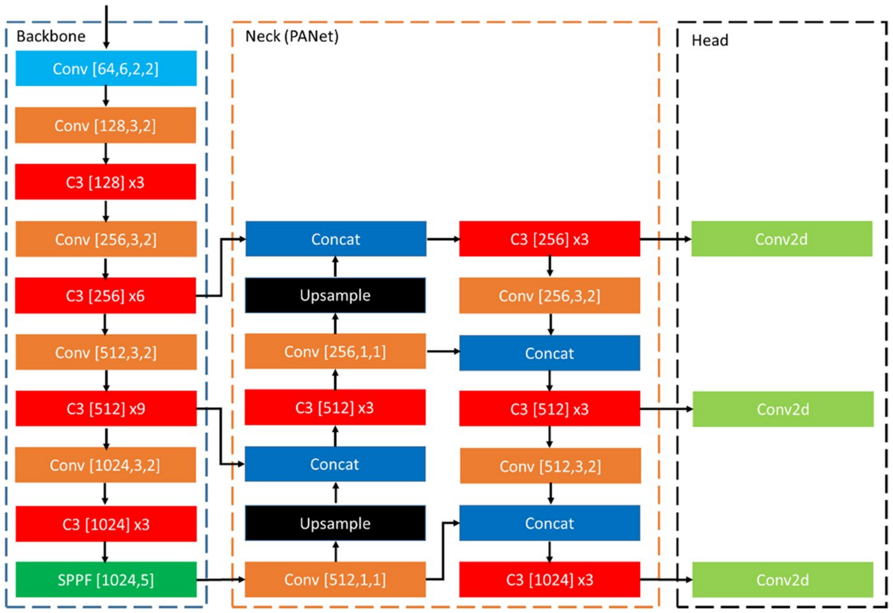

Fig. 1 The architecture of 6th Version of YOLOv5-Baseline

CSPDarknet53 which compensates the trade-off between inference speed and detection accuracy. In the details, the coordinate attention (CA) module is introduced into the backbone of the YOLOv5s model which follows two different schemes as shown in Fig. 2. The first scheme YOLOv5s-C3CA is the replacement of all of the C3 layers with C3CA along with the CA bottleneck as presented in Fig. 2b, while the second scheme YOLOv5s-CA is added to the CA module before the SPPF layer as shown in Fig. 2c. The CA module is added before SPPF and after the last scale of the feature map or replaces C3 with the C3CA bottleneck to achieve the target of improving the accuracy. Coordinate attention (CA) is a new efficient attention mechanism that was first published in 2021 [11].

The CA module reweights the importance of different channels and takes into account encoding spatial information. The input tensor is conured simultaneously from both attentions along its horizontal and vertical directions. Each element in the two attention maps indicates whether the object of interest exists in the

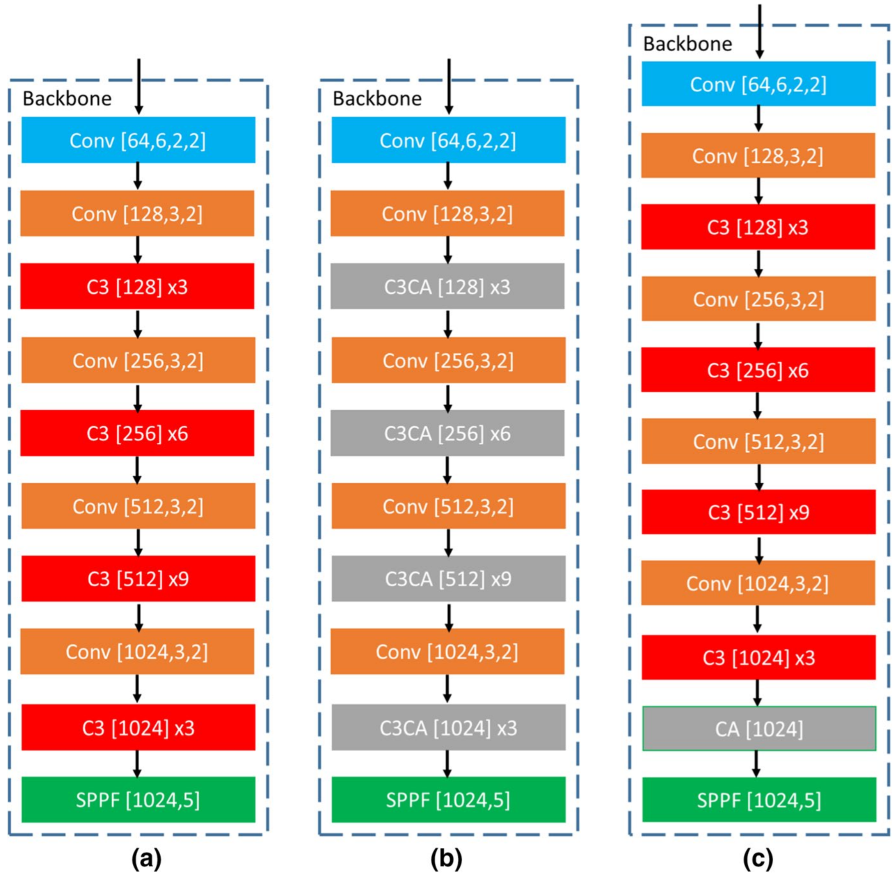

Fig. 2 Backbone Architecture of **a** YOLOv5s-Baseline, **b** Our Modified YOLOv5s-C3CA and **c** Our Modified YOLOv5s-CA

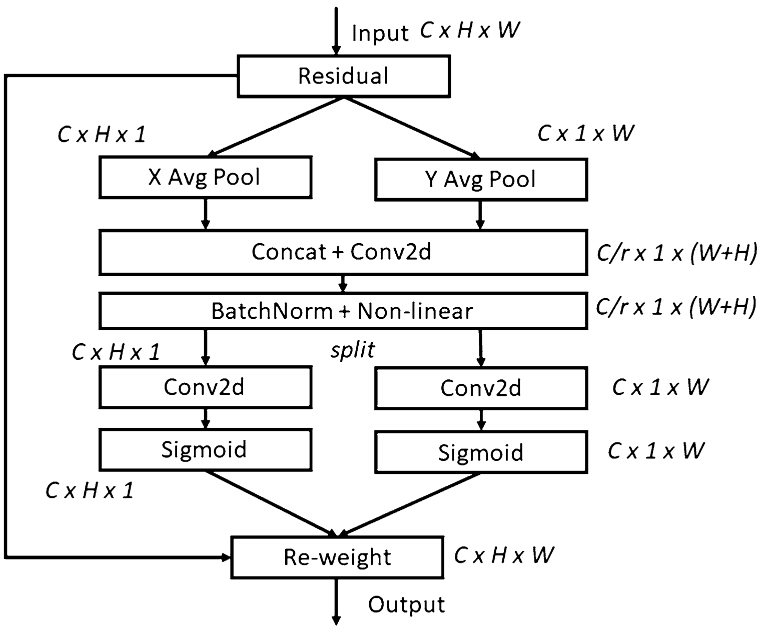

Fig. 3 Coordinate Attention Block Architecture

corresponding row and column. The encoding process helps the CA to localize the exact position of the object of interest more accurately and thus, allows the model to classify better.

Figure 3 describes the architecture of the model's coordinate attention block, whereby to alleviate the loss of position information caused by 2D global pooling, these authors decomposed the channel attention into two parallel (x and y directions) 1D feature encoding processes which effectively convert the spatial coordinate information into the generated attention map.

More specifically, these authors utilized two one-dimensional global pooling operations to aggregate the vertical and horizontal input features into two independent orientation-aware feature maps, respectively. Then, these two feature maps embedded with specific orientation information are encoded to be two attention maps, respectively, each of which captures the long-range dependencies of the input feature maps along one spatial direction. Therefore, the location information is stored in the generated attention map, while the two attention maps are then multiplied to the input feature map to enhance the representation ability of the feature map. Since this attention operation can distinguish spatial directions (i.e., coordinates) and generate coordinate-aware feature maps, this method is called coordinate attention (CA).

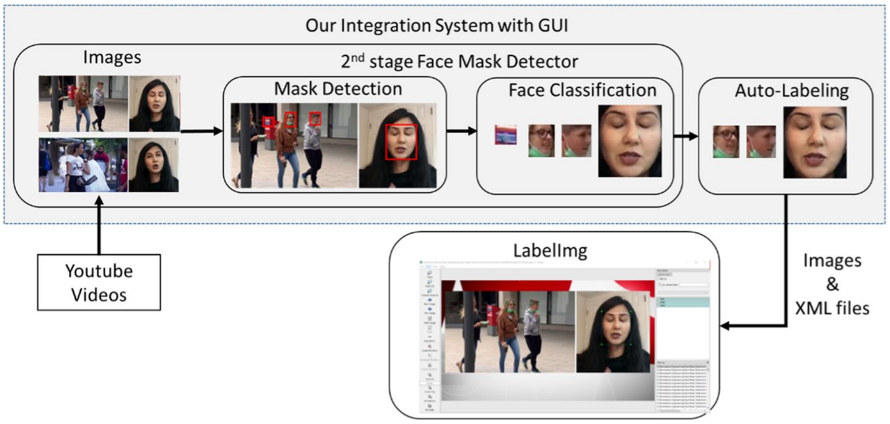

Fig. 4 Our proposed integration system of auto-labeling and 2nd stage face mask detector

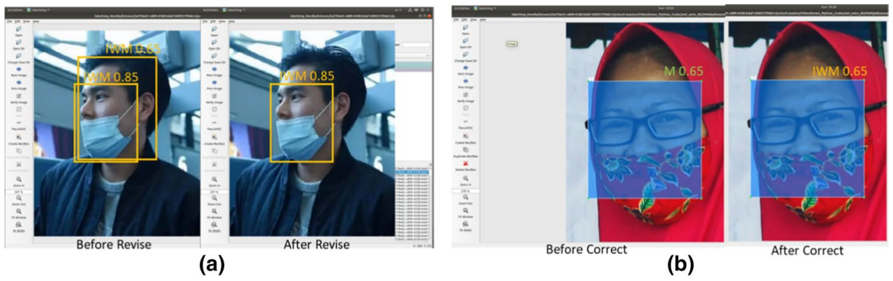

Fig. 5 Sample labeled images in case of **a** revise and **b** correct

3.2 Proposed system of auto-labeling and a two-stage face mask detector

Because the annotating process is time-consuming and labor-intensive to draw bounding boxes over multiple objects in images and name their labels, we propose an integration system of auto-labeling and a 2nd stage face mask detector to automatically perform this task. Figure 4 presents the process of creating our new labeled dataset from YouTube videos with our integration system.

To create a new image dataset, YouTube videos will be fetched into the 2nd Stage Face Mask Detector with an auto-labeling module to automatically generate annotations files with the PASCAL VOC format.

These XML files and images are saved in storage, and then, we use an open-source tool called LabelImg [34] to revise and correct these auto-labeled data as shown in Fig. 5. For example, when there are multiple bounding boxes with different confidence drawing around a single mask object, this mask object is need to revise which bounding box is the best fitting one. Then, we manually delete other bounding boxes and keep the best ones. Normally, the bounding box, which has the highest confidence, is the best choice. In another case, when a face with

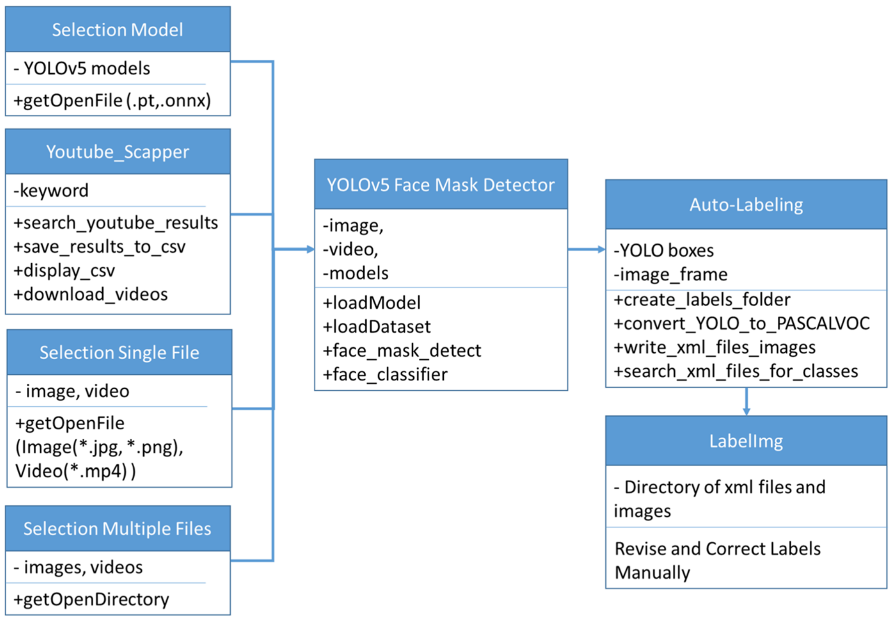

Fig. 6 UML of our system of YOLOv5-CA Faces Mask Detection integrated Auto-Labeling module

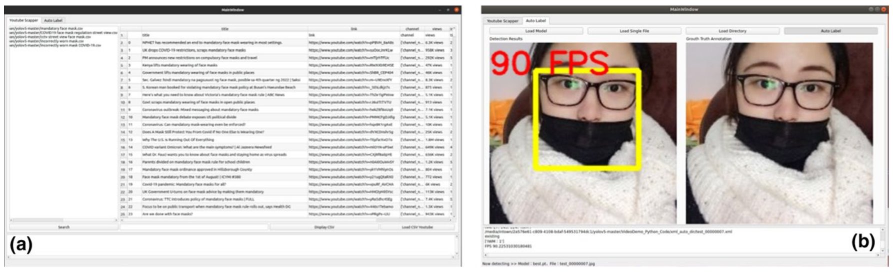

Fig. 7 GUI application of our system: **a** YouTube-Scrapper and **b** Auto-Labeling with YOLOv5 Face Mask Detection Module

a mask is classified wrongly as a mask, we need to correct the label on the top of this bounding box. The worst case is both need to revise and corrected, which takes a lot of time.

To reduce the number of labeled images that need to revise and corrected, we apply a two-stage detector with the first stage of mask detection and the second stage of face classification. Figure 4 shows an example of 2nd stage face mask detector. The core of 1st stage mask detection module is our improved YOLOv5-CA as presented in Sect. 3.1, and the 2nd stage face classification is applied to the pre-trained YOLOv5s face model in [35]. After the suspected object is detected as a region of interest (ROI) containing a mask, this object will be classified again with a pre-trained YOLOv5 face classifier. Only the detected object is also classified as the face will be the input of the auto-labeling module.

The ULM of our proposed auto-labeling system is shown in Fig. 6 which includes load input data module (i.e., select model, youtube_scapper, select single file, select multiple of image and videos), and YOLOv5 face mask detector, auto-labeling, and labeling module. We deploy a demo GUI application as shown in Fig. 7 to experiment with our system corresponding to the ULM. It should be noted that the labeling is not included in our GUI application, which is an open-source application for manually labeling images. The demo video of our face mask detection with auto-label onto the GUI application will be presented in the available data.

Figure 7a presents the YouTube_Scapper module, which is used to download videos from YouTube social media by selecting different video sources from CSV files, while Fig. 7b illustrates the GUI of the auto-labeling module to generate images and labeled annotation files from downloaded YouTube videos by the YouTube_Scapper module. As shown in Fig. 7b, the auto-label template has various buttons where users can select the model, select a single input file, or the whole directory before clicking the auto-label button to process. The input data of the face mask detector module are an image or frame-by-frame video, which will be processed through the two-stage detector. In the first stage of face detection, the suspected mask region or region of interest (ROI) will be localized together with the class label of “M”, “NM”, or “IWM”. One ROI can have multiple classified labels with a different confidence.

We apply an agnostic NMS setting of the default YOLOv5 platform, so one ROI has only one label with the highest confidence. Then, the ROI of mask prediction will be fetched to the second stage of the face classifier using the pre-trained face classification model “yolov5s-face.pt” from this link [35].

The confidence threshold of the face classifier is set high at 0.7, which means only the ROI of the mask classified as the face will be passed out as a labeled result. After processing, input data will be displayed on the “original” window screen while the labeled results will be shown on the “results” one. On the back-end side, the information of labeled results in YOLO format, including bounding boxes and classes, will be converted to Pascal VOC format, before writing to XML files in a predefined folder.

Note that, some functions of our system like load model, loadDataset, and detect are customized from this link [10], while youtube_scapper, second stage face_classfier, and auto-labeling functions are self-coded. The pseudocode of some important functions of our system will be presented as follows. Fig. 8 presents the pseudocode of the YouTube_Scapper function. By entering a search keyword (i.e., “incorrectly worn mask COVID-19”), using the selenium library and chrome web driver, the search results of YouTube videos will be collected. Then, the YouTube links and other parameters such as duration and published time will be saved into a CSV format file (i.e., “incorrectly worn mask COVID-19.csv”). After selection, part of YouTube’s links (i.e., 30 most relevant links) will be chosen to be downloaded as a “.mp4” video with the setting of highest resolution in a predefined saving folder. These videos will be selected as input for YOLOv5 face mask detection.

```
Input: keyword
Output: youtube video files ".mp4"
//search_youtube_results pseudocode
def get_video_results(keyword):
        driver = webdriver.Chrome()
        driver.get("https://www.youtube.com/results?search_query=keyword")
        for result in driver.find_elements_by_css_selector('.text-wrapper.style-scope.ytd-video-renderer'):
                link = result.find_element_by_css_selector('.title-and-badge.style-scope.ytd-video-
        renderera').get_attribute('href')
        youtube_data.append({'link': link,})
        csv_name = keyword + ".csv"
        pd.DataFrame(youtube_data).to_csv(csv_name)
//download_youtube_videos and rename  pseudocode
df = pd.read_csv("./incorrectly worn mask COVID-19.csv")
youtube_link=df["link"].tolist()]
for i in range(len(youtube_link)):
        url =str(youtube_link[i])
        get_video = YouTube(url)
        stream=get_video.streams.get_highest_resolution()
        stream.download('./youtube_video/')
        print('your video '+url+ ' is downloaded successfully')
        os.rename(os.path.basename(file),"youtube_video_"+str(i)+".mp4" )
```

Fig. 8 Pseudocode of our YouTube_Scapper function

```
Input: images, videos, model_mask weight, model_face weight
Output: YOLO bounding boxes and classes
//YOLOv5 face mask detection pseudocode
def YOLOv5 (source, mask_weight, face_weight)
    source =str(source)
    anno_folder=source.split("/")[-1][:-4]
    xml_dir_path = "./" +anno_name +"/"
    model = DetectMultiBackend(mask_weights, device=device, dnn=dnn, data=data, fp16=half)
    if source == "images":
        dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt)
    elif source == "videos":
        dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt)
    for path, im, im0s, vid_cap, s in dataset:
        pred = model(im, augment=augment, visualize=visualize)
        pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms, max_det=max_det)
        pred = face_classifier(pred, face_weight, im0s)
        for i, det in enumerate(pred):  # per image
            if len(det):
                for *xyxy, conf, cls in reversed(det):
                        c = int(cls)  # integer class
                        label = None if hide_labels else (names[c] if hide_conf else f'{names[c]} {conf:.2f}')
                        array_2_xml.append((int(xyxy[0]), int(xyxy[1]), int(xyxy[2]), int(xyxy[3]),names[c]))
                Autolabeling(xml_dir_path,anno_folder,im0s, array_2_xml)
```

Fig. 9 Pseudocode of our two-stage mask and face YOLOv5 detection module

Figure 9 presents the pseudocode of our two-stage YOLOv5 mask detector and face classifier. We customize some functions from the original YOLOv5 GitHub [10], such as “loadImages” for reading images, “loadStreams” for reading videos, “DetectMultiBackbend” for load mask model, and “non_max_suppression” with setting of agnostic_nms for classifying one object with one label with the highest confidence only. After the first stage of detection, the mask prediction results “pred” will be fetched into 2nd stage of the face classifier. After the two-stage, the YOLO bounding boxes results “pred” will be set as PASCAL VOC format in form of “xyxy[0]”, “xyxy[1]”, “xyxy[2]”, “xyxy[3]” together with class labels as “names[c]” in array “array_2_xml” to pass into “autolabeling” function to generate annotations files. Other functions like face_classifier and auto-labeling are self-code, which will be detailed and presented as follow.

The pseudocode of our face classifier function is shown in Fig. 10. The mask prediction results in the tensor format of 5 elements, including 4 bounding boxes and class, and the pre-trained face weight model and image data, will be fetched into “face_classifier”. Using the torch hub library, the pre-trained face weight model “yolov5s-face.pt” is loaded with a confidence threshold of 0.7 and an IoU threshold of 0.45. The mask prediction results of each image can include information on many suspected mask objects or none of the suspected objects. If the image consists of none of the suspected masks, face_classifier functions will return new_pred as None. If the image consists of many suspected objects, each object as variable “d” will be processed through for loop, with “a” of “d” consisting of 4 values of bounding boxes. The suspected mask area will be extracted from original image data “im0s” as “cutout” with information from “a”. These “cutout” will be classified through a face classifier. Then, the label results of

```
Input: mask prediction results, face weight, image data
Output: mask & face prediction results
//Face classifier pseudocode
def face_classifier(pred, face_weight, im0s):
    face_model=torch.hub.load (face_weight)
    new_pred=[]
    for i, d in enumerate(pred):
        if d is not None and len(d):
            for j, a in enumerate(d):
                cutout = im0s[i][int(a[1]):int(a[3]), int(a[0]):int(a[2])]
                results =  face_model ( [cutout] )
                labels = results.xyxyn [ 0 ] [ : ,-1 ].to ( 'cpu' ).numpy ( )
                if len(labels) != 0:
                    new_pred.append(d[j])
    return new_pred
```

Fig. 10 Pseudocode of our face classifier function

```
Input: xml_dir_path, folder_name, image_data, array_2_xml
Output: images, ".xml" files
//Auto labeling pseudocode
def Autolabeling(path, input_file, image, bboxes):
    os.makedirs(path)
    img_name = path + input_file +str(number)+ ".png"
    xml_name = path + input_file +str(number)+ ".xml"
    cv2.imwrite(img_name, image)
    annotation = etree.Element("annotation")
    for Object in boxes:
        class_name = Object[4]
        xmin_l = str(int(float(Object[0])))
         ymin_l = str(int(float(Object[1])))
        xmax_l = str(int(float(Object[2])))
        ymax_l = str(int(float(Object[3])))
    obj = etree.Element("object")
    annotation.append(obj)
     bndbox = etree.Element("bndbox")
    obj.append(bndbox)
    bndbox.append(xmin)
    s = etree.tostring(annotation, pretty_print=True)
```

Fig. 11 Pseudocode of our Auto-labeling module

classification are extracted and then, converted from transport to NumPy with device settings of "CPU". If labels are equal to zero, there is no face classified in "cutout" and the face_classifier function will return new_pred as None. Only if there is a face classified in "cutout", the face classifier function will return new_pred with the information of mask detected and face classified objects "d".

Figure 11 presents our auto-labeling pseudocode. The saving directory from the variable "path" will be created by "makedirs" function. Based on the user selection of the name of the input file, the corresponding annotation files and images will be created as "xml_path" + "input_file" + ".xml" or ".png". The boxes results of YOLOv5 face mask detection as an array of five elements, which consists of bounding boxes "xmin_l", "xmax_l", "ymin_l", "ymax_l" and label "class_name", will be written into ".xml" files. The original image or image frame by frame in case of video from YOLOv5 load_dataset function will be written to image format types as ".jpg" or ".png" through "imwrite" function of OpenCV with the corresponding frame by frame as variable "number". And the auto-labeling module output with two format files ".xml" annotation files and ".png" image files.

4 Experiments

4.1 Evaluation metrics

In YOLOs, the model accuracy performance is evaluated through mean average precision (mAP) [36], the higher the mAP leads the more accurate, and whereas. In order to understand mAP, there are several fundamental concepts relating to mAP as follows: Precision, Recall, Precision-Recall Curve, and F1 score.

These evaluation metrics are defined as follows [36]:

$$\text{Precision} = P = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

$$\text{Recall} = R = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

where TP, TN, FP, and FN, respectively, represent the number of true positives, true negatives, false positives, and false negatives.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

The Precision-Recall Curve is a plot of precision on the vertical axis and recalls on the horizontal axis. By definition, average precision (AP) is corresponding to the area under the Precision-Recall Curve and the AP value is determined by Eq. (5) shown as follows:

$$AP = \int_0^1 p(r)dr \tag{4}$$

where p(r) is Precision at Recall r.

In addition, the mAP is the average of the calculated average precision of all classes. In this work, there are the three classes of "M", "NM" and "IWM", thereby mAP is calculated as follows:

$$mAP = \frac{AP_M + AP_{NM} + AP_{IWM}}{3} \tag{5}$$

4.2 Environment setup

Three models including the YOLOv5-baseline, our improved YOLOv5-CA, and YOLOv5-C3CA are trained from scratch with a high option. The YOLOv5 training platform supports three hyperparameters of low, medium, and high levels for training. To evaluate fairly, other parameters are set with the default as the YOLOv5-basesline. We train these models for 300 epochs using an SGD optimizer with a batch size of 16 and an input image size of 640×640. After each epoch, the evaluation

Table 3 Preparation Dataset for Training and Testing

Datasets		Images	Labels		
			Mask (M)	Incorrectly Worn Mask (IWM)	No mask (NM)
Kaggle (853 images)	Train	682	2671	102	543
	Validation	171	561	21	174
	All	853	3232	123	717
FMLD (1450 images)	Test	1450	449	1528	178

Table 4 Training/ Testing Environment

Device	Configuration
Operating System	Ubuntu 20.04
Processor	Intel® Xeon(R) Silver 4210 CPU @ 2.20 GHz × 40
GPU	RTX 2080 10G × 2
GPU accelerator	CUDA 11.2, Cudnn 8.1
Framework	PyTorch 1.9.1
Complier IDE	Pycharm
Scripting language	Python 3.6

metrics are calculated including the AP for each class and the precision, recall, and mean AP (mAP). The Kaggle training dataset is divided into two categories, namely 80% training and 20% validation as described in Table 3. For testing our proposed YOLOv5s with a CA module, we extract 1450 images containing the class "incorrectly worn mask," of the FMLD dataset to verify our proposed method generalization ability. The training results of these models are demonstrated on available data.

These models are trained and tested by using 2 NVIDIA Graphics Processing Unit (GPU) with 10 GB of memory and a 2.20 GHz Intel® Xeon(R) Silver 4210 CPU. Our demo GUI application is developed using Python language and PyQt5 Designer. Other computer settings are described in Table 4.

5 Results and discussion

5.1 Training and testing results

Figure 12 shows the training process of the three models during 300 epochs including the bounding box loss (Box Loss), object loss (Obj Loss), and classification of the loss function (Cls Loss). Overall, both three models YOLOv5-baseline, YOLOv5s-CA, and YOLOv5s-C3CA are well-trained with no fitting phenomenon ever occurring. At the beginning of 30 epochs, the loss function value decreases rapidly. But then, the loss value remains stable roughly when the epoch reaches 50. The

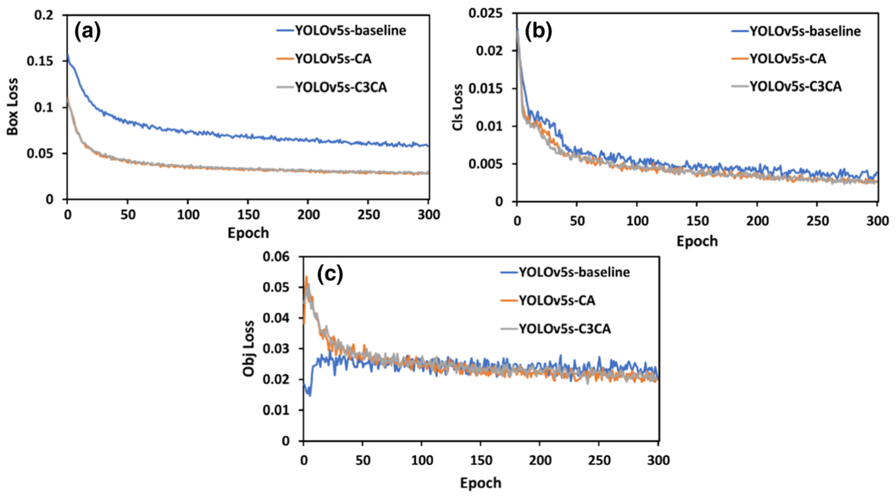

Fig. 12 Results of **a** Box Loss, **b** Classification, and **c** Object Loss during Training 300 Epochs

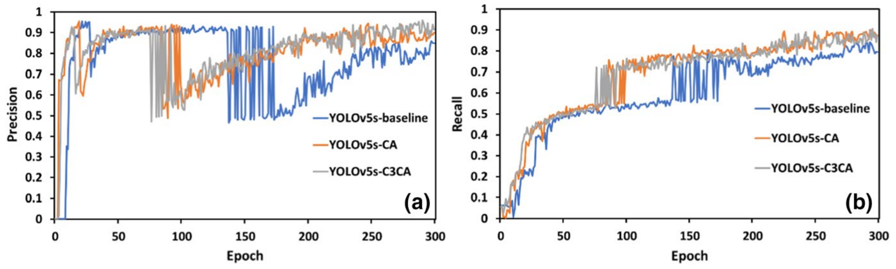

Fig. 13 Results of **a** Precision and **b** Recall during Training 300 Epochs

trained YOLOv5s model sets the loss function through the prediction and ground truth label information to update the network parameters. The convergence position of each loss function is less than 0.03 or approximately around 0.03 in the case of bounding box loss, which indicates the robustness and the effective prediction of the model. In addition, the purpose of adding the CA block is to accurately locate the position of the object of interest (or reduce box loss) and hence, allow the whole model to recognize better (or reduce the classification and object loss). In Fig. 12a, the box loss of the two CA-adding models remained lower than the YOLOv5s-baseline at approximately 0.02 after 300 epochs. Also, in Fig. 12b, c, the classification and object loss follow the same fundamentals as the box loss, but the values are lower than that of the box loss.

The Precision and Recall curves of three models during 300 epochs are shown in Fig. 13. During the first 30 epochs, these two curves rapidly increase before fluctuating significantly until 170 epochs and then, remain stable without obvious oscillation till 300 epochs. Obviously, both the Precision and Recall curves of the

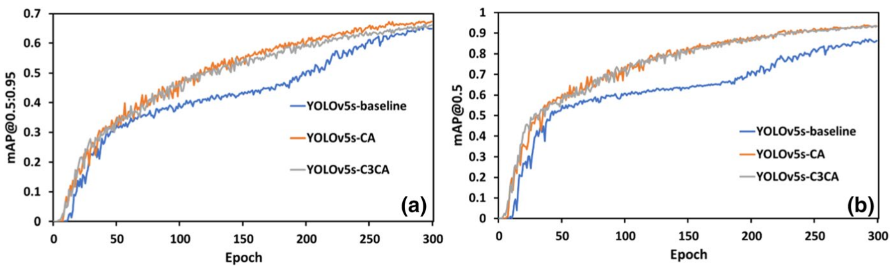

Fig. 14 Training Results of **a** mAP@0.5:0.95 and **b** mAP@0.5 during 300 Epochs

YOLOv5-baseline fall behind the YOLOv5s-C3CA and YOLOv5s-C3CA models after 170 epochs. In Fig. 13a, the precision of YOLOv5-baseline is about 0.8 compared to that of YOLOv5-CA and YOLOv5-C3CA as 0.9 and 0.87, respectively. As shown in Fig. 13b, the recall value of YOLOv5-baseline is approximately 0.73, while YOLOv5-CA and YOLOv5-C3CA are at about 0.83.

Figure 14 describes the training results of the three models with two types of accuracies, mAP@0.5:0.95 in Fig. 14a, and mAP@0.5 in Fig. 14b during 300 epochs. Overall, YOLOv5s-CA and YOLOv5s-C3CA are well-trained without obvious fluctuation and outperform the YOLOv5 baseline. These two models reach an over 80% accuracy for mAP@0.5 with over 40% for mAP@0.5:0.95 after reaching 150 epochs. Then, YOLOv5 with the CA module increases gradually in accuracy and remains steady until the end at over 90% for mAP@0.5 and 60% for mAP@0.5:0.95. While YOLOv5-baseline reaches an accuracy of mAP@0.5 at over 85% and mAP@0.5:0.95 at 60%.

A quantitative comparison between the three models is shown in Table 5. The batch size is set to 1 with a 640×640 image size when calculating the detection time per image in mini-seconds (ms). As shown in Table 5, it can be found that even though YOLOv5s-CA has one more CA layer than YOLOv5s-baseline, the weight size, parameters, and FLOPs as well as detection time per image with a batch size of 1 and 640×640 image size of the YOLOv5s-CA model is significantly similar to that of the YOLOv5s-baseline. YOLOv5s-CA has a weight of 14.4 MB, over 7.04 M parameters, and FLOPs of 15.8G and achieves 8.0 ms (125 FPS) in detection time per image. In the case of YOLOv5-C3CA, by replacing each original C3 bottleneck with a C3CA bottleneck module, YOLOv5s-C3CA is slightly lighter than the YOLOv5s-baseline in terms of volume weights, parameters, and FLOPs. Meanwhile, the detection time per image with a batch size of 1 is more than 4.3 ms compared to YOLOv5-baseline because of adding 84 more layers. Overall, the YOLOv5-CA outperforms both the YOLOv5 baseline and YOLOv5-C3CA. In detail, the YOLOv5-CA increases mAP@0.5 by 6.2%, mAP@0.5:0.95 by 1.3%, and F1-score value 6% compared to the YOLOv5s-baseline.

From the above results, adding the CA module into the YOLOv5 backbone structure proves to help boost the performance of object detection and classification with fast inference time. In the details, the training results of the best model

Table 5 Comparison Results between Three Models on Training Set

Model	Weight Size (MB)	Training Time	Detection time per image(ms)	Parameters (M)	FLOPs (G)	mAP@.5 (%)	mAP@.5:.95 (%)	F1-score
Yolov5s-baseline	14.4	0.693	8.1	7.01 (213 layers)	15.8	87	66.1	81
Yolov5s-C3CA	13.8	0.878	12.4	6.7 (297 layers)	15.3	92	63.7	86
Yolov5s-CA	14.4	0.787	8	7.04 (221 layers)	15.8	93.9	67.4	87

Table 6 Accuracy results of YOLOv5s-CA

Class	Precision (%)	Recall (%)	F1-score	AP@.5 (%)	AP@.5:.95 (%)
Incorrectly Worn Mask (IWM)	89.3	74.5	–/–	88	63.9
Mask (M)	92.8	97.8	–/–	98.5	72.9
No Mask (NM)	77.4	94.6	–/–	95.1	65.4
All	86.5	89	87	93.9	67.4

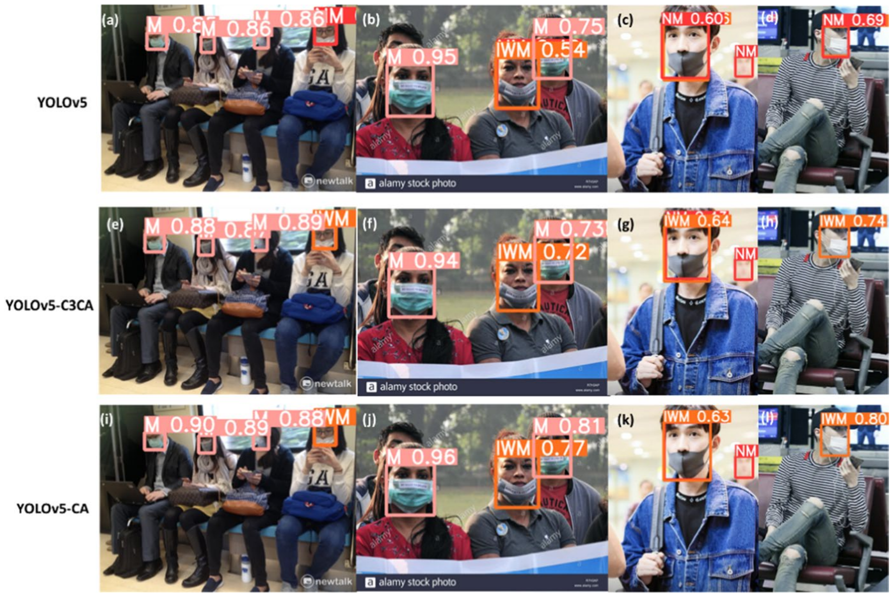

Fig. 15 Visualization of results of three models on Kaggle testing images

YOLOv5s-CA are shown in Table 6 including the precision, recall, F1-score, and accuracy of mAP@0.5 and mAP@0.5:0.95, respectively.

The visualization comparison results of the three models on Kaggle testing images are presented in Fig. 15. As in Fig. 15a, the YOLOv5s-baseline classifies wrongly the "IWM" label for "NM", while the two modified YOLOv5s with CA blocks classify correctly as shown in Fig. 15e, i. In Fig. 15b, the YOLOv5s-baseline exhibits confusion between "IWM" and "NM" as the two bounding boxes overlap each other. Meanwhile, the "IWM" confidence score is 0.54 or higher than that of "NM", and hence, "IWM" is drawn forward. As in Fig. 15c, g, both the YOLOv5s-baseline and YOLOv5s-C3CA both get confused between the two "IWM" and "NM" classes, but YOLOv5s-C3CA has a higher "IWM" confidence score than that of the YOLOv5s-baseline. As shown in Fig. 15k, only YOLOv5s-CA classifies the "IWM" label correctly. All in all, YOLOv5s-CA outperforms

Table 7 Comparison results of three models on testing FLMD set

Testing dataset	Models	Precision (%)	Recall (%)	F1-score	mAP@.5 (%)	mAP@.5:.95 (%)
FMLD 1450 images	YOLOv5s-baseline	34.2	69.5	31	43	19.9
	YOLOv5s-C3CA	41.8	62.8	37	43.5	19.8
	YOLOv5s-CA	37.7	66	35	45.3	20.6

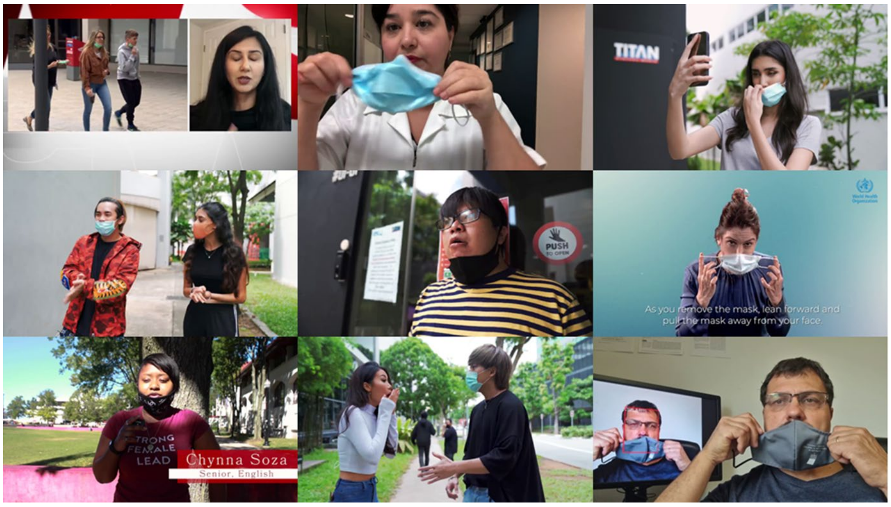
TITAN
PUSH TO OPEN
As you remove the mask, lean forward and pull the mask away from your face.
STRONG FEMALE LEAD
Chynna Soza
Senior, English

Fig. 16 Sample of our image dataset

both YOLOv5s-C3CA and the YOLOv5s-baseline in terms of correct classification of the three classes, especially the "IWM" class.

Table 7 describes the results of the testing set of 1450 images FMLD dataset. The testing is conducted with the same configuration as training. The improved YOLOv5-CA achieves the highest accuracy at 45.3% for mAP@0.5 and 20.6% for mAP@0.5:0.95, which increases by 2.3 and 0.7%, respectively, compared to the YOLOv5-baseline. YOLOv5s-C3CA obtains better results than the YOLOv5-baseline, but the difference is only 0.5% mAP@0.5. Whereas, the F-1-score value of the YOLOv5-C3CA gets a higher 2% than the YOLOv5-CA.

Table 8 The dataset of our images dataset into three categories

Datasets	Images	Labels		
		Mask (M)	Incorrectly Worn Mask (IWM)	No Mask (NM)
Our dataset (Kaggle, YouTube)	7110	4155	3559	4362

5.2 Auto-label results

We utilize our demo application of integration system with auto-label module and 2nd stage of YOLOv5 mask detection and face classification to process 30 YouTube videos and labels (70% auto and 30% manual) total of 6257 images. Figure 16 shows some examples of our labeled images from different angles and resolutions of face, gender, age, mask types, and mask color. As shown in Table 8, we combine these 6257 images and the original 853 Kaggle images to make our dataset of 7110 images, which includes “M”, “IWM”, and “NM” classes of 4155, 3559, and 4362 labels, respectively.

We further compare our proposed YOLOv5-CA with stage-of-the-art deep learning object detection algorithms, including YOLOX (July 2021) [37], YOLOv6 (June 2022) [38], and the latest single-stage object detector YOLOv7 (July-2022) [39] in terms of Precision, Recall, F1-score and mAP metric values. Their released source code and improved YOLOv5-CA are run on our 7110 images dataset with the same default settings for 30 epochs with a batch size of 16 and input image size of 640×640. It is noted that several evaluation metrics are not calculated by their released source code except for mAP value because of the primary evaluation metric of object detection [36].

These evaluation metrics are stated as “–/–”. It is clearly shown in Table 9 that our proposed YOLOv5s-CA method outperforms other methods with the accuracy mAP@0.5 of 96.8%. Following the time of release, each version of the YOLO-based model shows some improvement in terms of accuracy compared to the previous version. The accuracy gap reduces significantly to only 0.2% between our work and the latest YOLOv7-Tiny.

5.3 Discussion

Our improved YOLOv5s follows two different schemes of adding the CA module into the backbone of the YOLOv5s-baseline network called YOLOv5s-CA and YOLOv5s-C3CA. From the experiment results, both of our improved YOLOv5s-CA and YOLOv5-C3CA models achieve better accuracy than the YOLOv5 baseline while keeping fast inference time. Because of the lightweight structure of CA, the parameter of YOLOv5-CA is only 0.4 M parameters higher than that of YOLOv5 with approximately the same inference time of 0.8 ms. With the advanced GPU computing resources, adding the CA module into the YOLOv5 structure still have many schemes for improvements, such as other modifications in the neck and head structure of YOLOv5.

Dataset plays as an important role in the performance of objection detection based on deep learning methods, especially a one-stage detector, likely the YOLO family. Normally, when starting a new type of objection detection, the dataset is initially small-scale because it is challenging (time and labor) to do annotation manually. Particularly, the YOLO family requires images with annotations files of bounding boxes and classes, implying the requirement of manually drawing bounding boxes around an object of interest and labeling classes. Auto-labeling is first

Table 9 Comparison of our work and stage-of-the-arts algorithms

Models	Trained Model Weight Parameters	AP@0.5			mAP@0.5	Precision	Recall	F1-score
		NM	M	IWM				
YOLOX-n	–/– 0.91 M paras, 1.08 GFLOPs 7.6 MB	90.83	90.04	90.91	90.59	–/–	–/–	–/–
YOLOX-s	–/– 9 M paras, 26.8 GFLOPs 71.8 MB	90.86	90.57	90.91	90.78	–/–	–/–	–/–
YOLOv6-n	–/– 4.3 M paras, 11.1 GFLOPs 9.7 MB	–/–	–/–	–/–	95.72	–/—	–/–	–/–
YOLOv6-s	–/– 17.2 M paras, 44.2 GLOPs 38 MB	–/–	–/–	–/–	95.75	–/–	–/–	–/–
YOLOv7-Tiny	263 layers, 6 M paras, 13.2 GLOPs 12.3 MB	98	92.6	99.3	96.6	94	93.9	94
Our work (YOLOv5s-CA)	221 layers 7.042 M paras, 15.8 GFLOPs 14.4 MB	98.1	93	99.2	96.8	95.9	92.3	94

introduced in 2019, with much commercial software or open-source codes, but these software tools are not well-customized for our work. Instead, we develop a do-it-yourself (DIY) system that integrates an auto-labeling module and YOLO that would fully support our research and allow its application to other projects with new types of objects. With the development of the YOLO family, many variations such as YOLOX (2021), and YOLOv7 (2022) share the same PyTorch development platform as YOLOv5. Hence, our DIY module (auto-labeling and CA) can easily be customized to the new releases of the YOLO version. Apart from face mask detection, our system feasibly extends to helmets, safety vests, and glove detection in terms of personal protective equipment (PPE) automated inspection systems. This system helps to monitor adherence to safety protocols on construction sites, hospitals, or in a smart factory.

In contrast to these good points, there are several drawbacks of our system that need to be overcome. In the preliminary experiment, we apply single-state mask detection without the second stage of face classifier and without agnostic non-max-suppression setting on mask detection, the number of labeled images that need to revise and corrected is more than 20 percent or the auto-labeling only boosts the labeling processing by 50 percentage. To boost the auto-labeling to 70 percent by applying the 2nd stage face classifier and agnostic_nms setting, the computation requirements are much higher and the time of processing for single images is much slower. We perform this supercomputing task by utilizing Intel® Xeon(R) Silver 4210 CPU @ 2.20 GHz with 40 cores and two GPU RTX2080Ti 10 GB simultaneously and 1 TB Memory Disk to generate over 6000 images and XML files from over 130 GB of 30 YouTube videos. It is noted that these videos are downloaded from YouTube with the highest resolution possible (such as 720×720) to get better image quality and the duration time from 5 to 10 min to reduce the processing time. Also, the pre-processing of extracting images from videos at 30 frames per second (fps) also takes many times as we only extract 1 image per 3 frames to reduce the similar images. The similarity of images is also the drawback of extracting images from videos. We consider extending collecting images from different sources (e.g., google search images).

6 Conclusion

In response to the pandemic, we propose an improved YOLOv5s-CA face mask detection that is capable of identifying people's faces with or without masks, as well as incorrectly worn masks on real faces and real masks. Our improved YOLOv5s follow two different schemes of adding the CA module into the YOLOv5s-baseline's backbone, namely YOLOv5s-CA and YOLOv5s-C3CA. From the experiment results, compared to the YOLOv5-baseline, both of our improved YOLOv5s have better results for mAP@0.5 and mAP@0.5:0.95 accuracies, but YOLOv5s-CA achieves the best mAP@0.5 at 93.9% and 67.4% mAP@0.5:0.95 on the Kaggle dataset. In addition, we propose a system that integrates improved YOLOv5s-CA face mask detection with auto-labeling models that can reduce manual annotation work. Furthermore, we deploy an integration system into the GUI demo application, which can be easily customized for another object detection project. In this work, we utilize our GUI demo application to create a new large real dataset of 7110 images including three categories "M", "NM" and "IWM" with more than 3500 labels for each class. In addition, the proposed method YOLOv5s-CA outperforms other SOTA YOLO-based models on this 7110 images dataset. In future work, we consider taking into account GPU-based embedded devices (e.g., Nvidia Jetson) for real-time analysis with our improved YOLOv5s-CA.

Funding No Funding.

Availability of data and materials The languages used were Python and XML, and they were developed using the Pycharm IDE and utilized open-source software labeling. Training, testing results and video demonstration of the auto-labeling module are presented in this link: **Thi-Ngot Pham, Viet-Hoan Nguyen, Jun-Ho Huh** [40].

Declarations

Competing interests The authors declare that they have no competing interests. We confirm we have included a data availability statement in my main manuscript file.

References

1. WHO. WHO Corona-Viruses (COVID-19). Accessed on 5 Jan 2022. [Online] Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019
2. Palanisamy V, Thirunavukarasu R (2019) Implications of big data analytics in developing healthcare frameworks– a review. J King Saud Univ Comput Inform Sci 31(4):415–425
3. Chu DK et al (2020) Physical distancing, face masks, and eye protection to prevent person-to-person transmission of SARS-CoV-2 and COVID-19: a systematic review and meta-analysis. The Lancet 395(10242):1973–1987
4. Nowrin A, Afroz S, Rahman MS, Mahmud I, Cho Y-Z (2021) Comprehensive review on facemask detection techniques in the context of COVID-19. IEEE Access 9:106839–106864. https://doi.org/10.1109/ACCESS.2021.3100070
5. Wang B, Zheng J, Chen CLP (2022) A survey on masked facial detection methods and datasets for fighting against COVID-19. IEEE Trans Artif Intell 3(3):323–343. https://doi.org/10.1109/TAI.2021.3139058
6. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 779–788 https://doi.org/10.1109/CVPR.2016.91
7. Liu W et al (2016) SSD: single shot multibox detector, In Computer Vision – ECCV Springer International Publishing, pp 21–37
8. Fan X, Jiang M. RetinaFaceMask: a single stage face mask detector for assisting control of the COVID-19 Pandemic. arXiv:2005.03950. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2020arXiv200503950F
9. Ren S, He K, Girshick R, Sun J (2017) Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031
10. Glenn J et al. v6.0 - YOLOv5n ’Nano’ models, Roboflow integration, TensorFlow export, OpenCV DNN support. Accessed on 17 Feb 2022. [Online]. Available: https://github.com/ultralytics/yolov5/releases/tag/v6.0
11. Hou Q, Zhou D, Feng J. Coordinate attention for efficient mobile network design. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021 (pp. 13713–13722)
12. Wang B, Zhao Y, Chen CP (2021) Hybrid transfer learning and broad learning system for wearing mask detection in the COVID-19 era. IEEE Trans. Instrum. Meas. 70:5009612
13. [Online]. Available: https://github.com/AIZOOTech/FaceMaskDetection
14. [Online]. Available: https://www.kaggle.com/andrewmvd/face-maskdetection
15. Batagelj B, Peer P, Struc V, Dobri Sek S (2021) How to correctly detect face-masks for covid-19 from visual information? Appl Sci 11(5):1–24
16. Jiang X, Gao T, Zhu Z, Zhao Y (2021) Real-time face mask detection method based on yolov3. Electronics 10(7):1–17
17. Eyiokur FI, Ekenel HK, Waibel A (2021) A computer vision system to help prevent the transmission of covid-19. arXiv preprint arXiv:2103.08773
18. [Online] “Yolo-fastest: Yolo universal target detection model combined with efficientnet-lite,” https://github.com/dog-qiuqiu/ Yolo-Fastest, 2020
19. Redmon J, Farhadi A (2017) YOLO9000: better, faster, stronger. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 6517–6525, https://doi.org/10.1109/CVPR.2017.690
20. Redmon J, Farhadi A. YOLOv3: an incremental improvement. arXiv:1804.02767. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2018arXiv180402767R

21. Bochkovskiy A, Wang C-Y, Liao H-YM. YOLOv4: optimal speed and accuracy of object detection. arXiv:2004.10934. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2020arXiv200410934B
22. Ding Y, Li Z, Yastremsky D (2021) Real-time face mask detection in video data. arXiv preprint arXiv:2105.01816
23. Guo MH, Xu TX, Liu JJ et al (2022) Attention mechanisms in computer vision: a survey. Comp Visual Media 8:331–368
24. Jiang X, Gao T, Zhu Z, Zhao Y (2021) Real-time face mask detection method based on YOLOv3. Electronics. https://doi.org/10.3390/electronics10070837
25. Hu J, Shen L, Albanie S, Sun G, Wu E (2019) Squeeze-and excitation networks
26. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C (2018) MobileNetV2: Inverted residuals and linear bottlenecks. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 4510–4520, https://doi.org/10.1109/CVPR.2018.00474
27. Wang Z, Wang P, Louis PC, Wheless LE, Huo Y. WearMask: fast in-browser face mask detection with serverless edge computing for COVID-19. arXiv:2101.00784. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2021arXiv210100784W
28. Loey M, Manogaran G, Taha MHN, Khalifa NEM (2021) A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. Measurement 167:108288
29. Prusty MR, Tripathi V, Dubey A (2021) A novel data augmentation approach for mask detection using deep transfer learning. Intell-Based Med 5:100037
30. Kumar A, Kalia A, Verma K, Sharma A, Kaushal M (2021) Scaling up face masks detection with yolo on a novel dataset. Optik 239:166744
31. Yu J, Zhang W (2021) Face mask wearing detection algorithm based on improved yolo-v4. Sensors 21(9):1–21
32. Paszke A et al (2019) PyTorch: an imperative style, high-performance deep learning library. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., p 721
33. Josephn R. Darknet: open source neural networks in c. Accessed on 17 Feb 2022. [Online] Available: http://pjreddie.com/darknet
34. [Online] https://github.com/tzutalin/labelImg/
35. [Online] https://github.com/deepcam-cn/yolov5-face
36. Lin T-Y et al (2014) Microsoft COCO: common objects in context. In Computer Vision – ECCV 2014, Springer International Publishing, Cham, pp 740–755
37. [Online] https://github.com/Megvii-BaseDetection/YOLOX
38. [Online] https://github.com/meituan/YOLOv6
39. [Online] https://github.com/WongKinYiu/yolov7
40. Pham T-N, Nguyen V-H, Huh J-H. Github Code. https://github.com/MiuMiao93/Integration-system-of-improved-YOLOv5-face-mask-detection-and-auto-label-module

Authors and Affiliations

Thi-Ngot Pham[1,2] · Viet-Hoan Nguyen[3,4] · Jun-Ho Huh[2,5]

Thi-Ngot Pham
ngotpham@g.kmou.ac.kr

Viet-Hoan Nguyen
viethoan_nvh@pukyong.ac.kr

[1] Department of Data Informatics, (National) Korea Maritime and Ocean University, Busan, Republic of Korea

[2] Interdisciplinary Major of Ocean Renewable Energy Engineering, (National) Korea Maritime and Ocean University, Busan 49112, Republic of Korea

[3] Department of Artificial Intelligence Convergence, Pukyong National University, Busan, Republic of Korea

[4] Research Institute, Intown Company, Busan, Republic of Korea

[5] Department of Data Sciences, (National) Korea Maritime and Ocean University, Busan, Republic of Korea