A Review and Implementation of Object Detection Models and Optimizations for Real-time Medical Mask Detection during the COVID-19 Pandemic

Ioanna C. Gogou
Dept. of Computer Engineering and Informatics
University of Patras
Patras, Greece
gogou@ceid.upatras.gr

Dimitrios A. Koutsomitropoulos
Dept. of Computer Engineering and Informatics
University of Patras
Patras, Greece
koutsomi@ceid.upatras.gr

***Abstract*—Convolutional Neural Networks (CNN) are commonly used for the problem of object detection thanks to their increased accuracy. Nevertheless, the performance of CNN-based detection models is ambiguous when detection speed is considered. To the best of our knowledge, there has not been sufficient evaluation of the available methods in terms of the speed/accuracy trade-off in related literature. This work assesses the most fundamental object detection models on the Common Objects in Context (COCO) dataset with respect to this trade-off, their memory consumption, and computational and storage cost. Next, we select a highly efficient model called YOLOv5 to train on the topical and unexplored dataset of human faces with medical masks, the Properly-Wearing Masked Faces Dataset (PWMFD), and analyze the benefits of specific optimization techniques for real-time medical mask detection: transfer learning, data augmentations, and a Squeeze-and-Excitation attention mechanism. Using our findings in the context of the COVID-19 pandemic, we propose an optimized model based on YOLOv5s using transfer learning for the detection of correctly and incorrectly worn medical masks that surpassed more than two times in speed (69 frames per second) the state-of-the-art model SE-YOLOv3 on the PWMFD dataset while maintaining the same level of mean Average Precision (67%).**

***Index Terms*—real-time object detection, medical mask detection, video surveillance, YOLOv5, PWMFD**

I. Introduction

Computer vision has become an integral part of modern systems in transportation, manufacturing, and healthcare. In the last decade, the task of object detection as a deep learning problem has accumulated immense scientific interest. Convolutional Neural Networks (CNN) have shown excellent results in extracting the abstract features of image data, thanks to their similarities to the biological neural networks of the human brain [1]. Their promising capabilities have motivated scientists towards brilliant inventions of new state-of-the-art object detectors resulting in a continuous increase in accuracy. Nevertheless, their performance is ambiguous when detection speed is considered, which is usually sacrificed in favor of accuracy. Conducting accurate object detection in real-time is a realistic requirement of modern systems, especially embedded ones with hardware limitations. However, the available methods have yet to be fully evaluated as published research [2]–[5] tends to overlook the trade-off between accuracy and speed, compare models on different machine learning frameworks, or exclude newer models, resulting in indefinite results.

This work focuses on giving a solution to this problem by assessing some of the most fundamental CNN-based object detection models: Faster R-CNN [6] and Mask R-CNN [7] of the family of Region-based Convolutional Neural Networks (R-CNN) [8], RetinaNet [9], Single Shot MultiBox Detector (SSD) [10], and You Only Look Once (YOLO) [11] and its newer versions [12]–[15]. The objective is to evaluate and compare them in terms of GPU memory consumption, computational and storage cost as well as their speed/accuracy trade-off. We seek to reach fair conclusions by executing the models through a common pipeline, using the same machine learning framework, dataset, and GPU. At the same time, we ensure that our experiments can be reproduced through our accompanying open-source code.

Among those we evaluated, we choose a highly efficient model to train and optimize for real-time detection on a topical and novel dataset that has yet to be extensively tested. In view of the protective measures put in place during the COVID-19 pandemic, the need for real-time detection of correctly and incorrectly worn medical masks in data streams has become evident. According to the Worldwide Health Organization (WHO), the use of medical masks combined with other health measures is recommended for the containment of the virus [16]. In this context, we propose an optimized real-time detector of correctly and incorrectly worn medical masks. To achieve top performance, we investigate optimization techniques used before in medical mask detection. For instance, transfer learning from larger and more diverse object detection datasets is expected to improve model accuracy.

The main contributions of this work are the following:

- An analysis of the speed/accuracy trade-off of known object detectors using the same framework, dataset, and GPU. No other similar study has been published that includes YOLOv5 [15], whose performance has yet to be extensively tested.

- The accuracy and speed of YOLOv5s were evaluated for the first time on the newly-developed Properly-Wearing Masked Faces Dataset (PWMFD) [17] dataset. Furthermore, the effects of transfer learning, data augmentations, and an attention mechanism were assessed for medical mask detection.
- A real-time medical mask detection model based on YOLOv5 was proposed that surpassed more than two times in speed (69 fps) the state-of-the-art model SE-YOLOv3 [17] on the PWMFD dataset while maintaining the same level of mean Average Precision (mAP) at 67%. This increase in speed gives room for using the model on embedded devices with lower hardware capabilities, while still achieving real-time detection.
- All results are reproducible through our open-source code on GitHub[1]. The weight file of our medical mask detector is also available on GitHub[2] and on Hugging Face[3].

II. Previous Work

A. Object Detection Models

The development of AlexNet [18] in 2012 paved the way towards the CNN-based object detection models we know today. Introduced in 2014, R-CNN [8] was the first one to adopt the idea of region proposals for object detection, produced through a selective search algorithm. In 2015, its successor, Fast R-CNN [19], increased detection speed by computing a feature map for the whole image rather than for each proposal. It was improved further with Faster R-CNN [6] by replacing selective search with a more efficient fully convolutional region proposal network. In 2017, Mask R-CNN [7] was an extension of Faster R-CNN for image segmentation.

In 2015, the first one-shot model was published named YOLO [11]. Two iterations followed, YOLOv2 [12] and YOLOv3 [13], improving its performance with the addition of anchors, batch normalization, the Darknet-53 backbone, and three detection heads. In 2020, its development resumed with YOLOv4 [14]. It included the enhanced CSPDarknet-53 backbone, a Spatial Pyramid Pooling (SPP) [20] layer, and a Path Aggregation Network (PANet) [21]. Shortly after, YOLOv5 [15] was launched with only small alterations. Its role as the fifth version of YOLO was a controversial subject as no official publication has been released to this day. Nevertheless, it shows promising results through consistent updates, which are worth investigating. Finally, SSD [10] proposed in 2016 and RetinaNet [9] in 2017 are two other notable one-shot models. The latter became known for introducing focal loss to combat foreground-background imbalance.

B. Speed/Accuracy Trade-off of Object Detection Models

In recent years, several reviews of modern object detection models have been published. In 2019, a survey [2] was conducted on the improvements in object detection in the last 20 years. Nonetheless, the survey merely reported on the performance of the models in question in related bibliography. No experiment was performed to measure their accuracy and speed. In contrast, another review in the same year [3] included an experimental evaluation of more than 26 models on the Visual Object Classes (VOC) [22] and Common Objects in Context (COCO) [23] datasets. Although both accuracy and speed were measured, the trade-off between the two parameters was not analyzed. Moreover, the tested models were implemented in different machine learning frameworks and programming languages, thus obscuring detection speed.

Two studies published in 2017 [4] and 2018 [5] assessed the speed/accuracy trade-off of object detection models using the same framework. In [4], the Tensorflow implementations of Faster R-CNN [6], SSD [10], and Region-based Fully Convolutional Network (R-FCN) [24] were evaluated on COCO [23] based on their speed/accuracy trade-off while testing different backbones, image resolutions, and numbers of region proposals. In [5], a similar analysis was conducted using the same models with the addition of Mask R-CNN [7] and SSDlite [25]. The aspect of memory consumption and detection speed on different devices was investigated as well. Nevertheless, both studies do not take into account newer models, such as YOLOv5 [15] and RetinaNet [9].

C. Medical Mask Detection

Initially, interest in medical mask detection was limited. After the outbreak of COVID-19, numerous medical mask detection models were proposed to limit infection.

In 2020, the Super-Resolution and Classification Network [26] was designed and trained using transfer learning. In 2021, RetinaFaceMask [27] was introduced, a one-shot model based on RetinaNet, using transfer learning from a human face dataset to the Masked Faces for Face Mask Detection Dataset (MAFA-FMD) made by the authors. In the same year, a hybrid medical mask recognition model [28] combining ResNet-50 [29] with a Support Vector Machine (SVM) was proposed, after being trained on both real-world and synthetic data using transfer learning. Later in [30], the authors replaced the SVM with YOLOv2 [12]. A medical mask detector published in [31] was based on Inception-v3 [32] and trained on a synthetic mask dataset using transfer learning from a general object dataset and several data augmentations.

Despite achieving near-perfect accuracy, the above models were not evaluated on their detection speed. In view of this, a real-time mask detection model was designed, SE-YOLOv3 [17], and trained on the novel Properly-Wearing Masked Face Detection (PWMFD) dataset created by the authors. It introduced a Squeeze-and-Excitation (SE) attention mechanism [33] to YOLOv3, achieving 66% mAP and 28 fps. However, its performance was evaluated using a high-end GPU, rendering it unsuitable for lightweight devices. Moving to YOLOv4, the hybrid mask detection model tiny-YOLOv4-SPP was proposed [34]. It significantly reduced training time while increasing accuracy compared to the original tiny-YOLOv4, but the aspect of real-time detection was not considered.

[1]github.com/joangog/object-detection
[2]github.com/joangog/object-detection-assets
[3]huggingface.co/joangog/pwmfd-yolov5

III. Evaluation of Object Detection Models

A. Evaluation Methodology

We evaluated the object detection models shown in Table I in terms of memory consumption, computational and storage cost as well as their speed/accuracy trade-off. The models include Faster R-CNN with two different backbones and image sizes, Mask R-CNN, RetinaNet, SSD and its memory-efficient version SSDlite, YOLOv3 with its SPP and tiny variants, YOLOv4, and YOLOv5 including its large, medium, small, and nano variants. For all models, we utilized the same framework, dataset, and GPU.

1) Framework: All models are implemented in PyTorch and are offered in the Torchvision package. The only exceptions are YOLOv3, YOLOv4, and YOLOv5, which are implemented in GitHub repositories[4][5][6].

2) Dataset: The evaluation is done on the val subset of the COCO2017 [23] dataset. COCO contains 118,000 examples for training and 5,000 for validation belonging to 80 classes of everyday objects. No training phase is performed as all models are pre-trained on the train subset. The data are loaded with a batch size of 1 to imitate the stream-like insertion when detecting in real-time.

3) Environment: The code for the experiment is organized in two Jupyter notebooks (coco17_inference.pynb, analysis.pynb) and is executed through the Google Colab platform. We chose the option of a local runtime, which uses the GPU of our system (Nvidia Geforce GTX 960, 4 GB). Every model goes through the same inference pipeline.

4) Metrics: To estimate the memory usage of each model we calculate the maximum GPU memory allocated to our GPU device by CUDA for the program. To quantify computational costs, Giga Floating Point Operations (GFLOPs) are counted using the ptflops[7] Python package. The storage cost is derived from the size of the weight file of each respective model. We measure detection speed in frames (images) per second (fps). For accuracy, we use the mean Average Precision (mAP) of all classes. According to the COCO evaluation standard[8], Average Precision (AP) is calculated using 101-point interpolation on the area under the P-R (Precision-Recall) curve, as follows:

$$AP_{101} = \frac{1}{101} \sum_{R=\{0,0.01,\ldots,0.99,1\}} P_{interp}(R) \tag{1}$$

where

$$P_{interp}(R) = \max_{\tilde{R}:\tilde{R}\geq R} P(\tilde{R}) \tag{2}$$

B. Memory Consumption

In Fig. 1a, the higher the number of parameters and image size, the more GPU memory the model consumes. Interestingly, YOLOv4 has 925 MB of memory consumption, three times more than models with roughly the same parameter count and image size, such as YOLOv3 (301 MB). In contrast, YOLOv3 utilizes approximately three-fourths of the memory consumed by YOLOv5l, despite having more parameters and the same image size. SSDlite, having the fewest parameters and smallest image size, uses merely 33 MB.

[4]github.com/ultralytics/yolov3
[5]github.com/Tianxiaomo/pytorch-YOLOv4
[6]github.com/ultralytics/yolov5
[7]github.com/sovrasov/flops-counter.pytorch
[8]cocodataset.org/#detection-eval

TABLE I
Models evaluated on the COCO2017 [23] dataset.

Model	Backbone	Image Size[a]
Faster R-CNN [6]	MobileNetV3-Large [35]	800
Faster R-CNN	MobileNetV3-Large	320
Faster R-CNN	ResNet-50 [29]	800
Mask R-CNN [7]	ResNet-50	800
RetinaNet [9]	ResNet-50	800
SSD [10]	VGG16 [36]	300
SSDlite [25]	MobileNetV3-Large	320
YOLOv3 [13]	Darknet53 [13]	640
YOLOv3-spp [13]	Darknet53	640
YOLOv3-tiny [13]	Darknet53	640
YOLOv4 [14]	CSPDarknet53 [14]	608
YOLOv5l [15]	Modified CSPDarknet [15]	640
YOLOv5m [15]	Modified CSPDarknet	640
YOLOv5s [15]	Modified CSPDarknet	640
YOLOv5n [15]	Modified CSPDarknet	640

[a]Given a value of N, the image size in pixels is NxN.

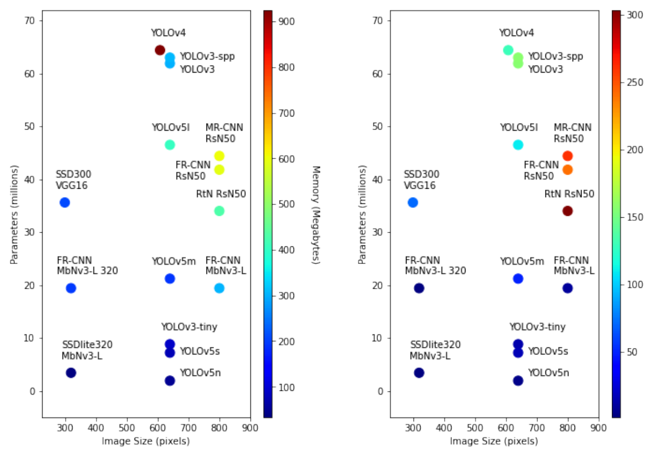
C. Computational and Storage Cost

Firstly, the number of GFLOPs of a model is closely related to its number of parameters and image size. For instance, in Fig. 1b, between the two implementations of Faster R-CNN with MobileNet-v3 and ResNet-50, the second executed 27 times more GFLOPs than the first while having just over twice the same number of parameters. In general, models that use the MobileNet-v3 backbone, i.e., SSDlite and Faster R-CNN, proved to be the least expensive in GFLOPs. Subsequently, comparing the two versions of Faster R-CNN with different image sizes, the one with size 800 cost almost six times more GFLOPs than the one with 320. The sole exception to the rule was RetinaNet and YOLOv3 that, despite having the same image size and fewer parameters than Faster R-CNN ResNet-50 and YOLOv4 respectively, executed more GFLOPs than their counterpart.

According to Fig. 2, storage cost increases linearly to the number of parameters in a model. Nevertheless, YOLOv3 and YOLOv5 cost significantly less storage space than models with an equivalent number of parameters.

D. Speed/Accuracy Trade-off

In Fig. 3 we illustrate the trade-off between the speed (fps) and accuracy (mAP) of all models. An immediate observation is a decrease in speed as the accuracy of a model increases. On a general note, a favorable model would be one that achieves both high accuracy and speed. Thus, it would be found in the top right corner of Fig. 3. All variations of YOLOv5 provided the best balance between accuracy and speed, whereas RetinaNet, SSD, SSDlite, and YOLOv3-tiny ranked last.



(a) Memory Consumption (b) Computational Cost

Fig. 1. Memory consumption in Megabytes (a) and computational cost in GFLOPs (b) in relation to parameter count and image size of object detection models evaluated on the COCO2017 [23] dataset.

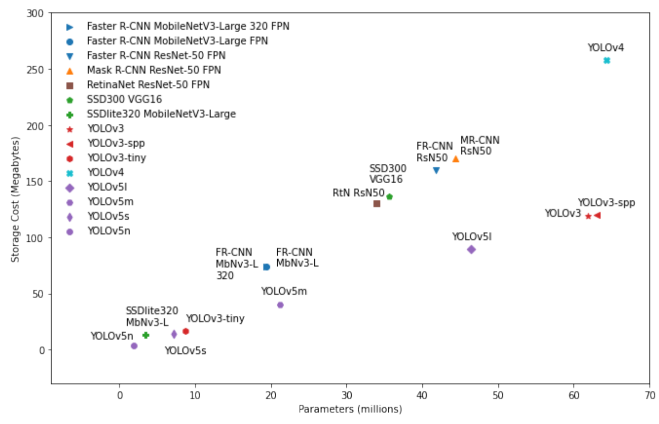


Fig. 2. Relationship between storage cost in Megabytes and parameter count of object detection models evaluated on the COCO2017 [23] dataset.

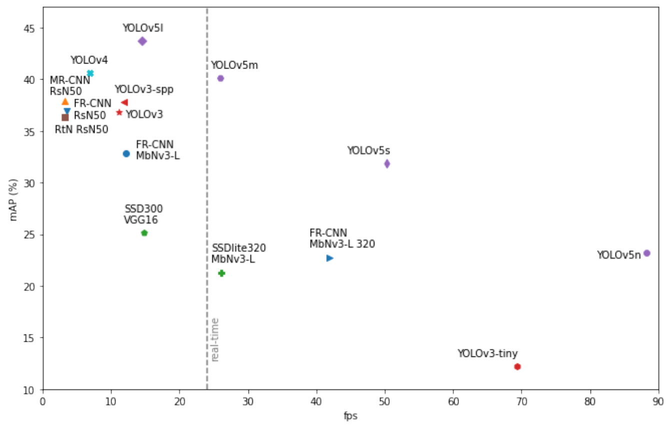


Fig. 3. Speed (fps) / Accuracy (mAP) trade-off of object detection models evaluated on the COCO2017 [23] dataset.

IV. Implementation of Medical Mask Detector

A. *Configuration*

1) Dataset: To achieve desirable results, a realistic and diverse dataset for both localization and recognition of medical masks is required; one that is large enough to ensure high accuracy but does not exceed our hardware limitations. Therefore, we selected the newly-created PWMFD [17] dataset, using its train and validation subsets to train and evaluate our model respectively. It includes 9,205 real-life images with 18,532 annotations of faces belonging to three classes, "with mask", "incorrect mask" and "without mask."

2) Environment: The code for the experiment is executed through Google Colab and is organized in three Jupyter notebooks (mask_training.pynb, mask_inference.pynb, analysis.pynb). Training is performed using the GPU provided by Colab (Nvidia Tesla K80, 12 GB) due to its memory facilitating a larger batch size, whereas evaluation is performed using our local GPU (Nvidia Geforce GTX 960, 4 GB) because of its higher detection speed.

3) Training and Evaluation: During training, a batch size of 32 is used. Training lasts for 50 epochs, as more resulted in overfitting. Learning rate is updated according to the OneCycleLR [37] policy with values in the range of [0.001, 0.01]. As an optimizer, Stochastic Gradient Descent is employed with a value of 0.937 for momentum and 0.0005 for weight decay. Cross-Entropy is used for classification loss and Complete Intersection over Union (CIoU) for localization loss. After training, the final weights are selected from the epoch with the highest mAP. During evaluation, inference is performed with a batch size of 1 to mimic the sequential input of data in real-time. COCO mAP and fps are measured as metrics.

B. *Model*

According to our study in Section III, YOLOv5 [15], and particularly its medium, small and nano variants, provided the best balance between accuracy, speed, memory consumption, and computational and storage costs for real-time object detection. Its architecture is depicted in Fig. 4. To implement our medical mask detector, we chose to train its small variant, YOLOv5s, on PWMFD [17] with an image size of 320, achieving 33% mAP and 69 fps. To elevate its performance, we experimented with various optimizations during training.

C. *Optimizations*

1) Transfer Learning: Inspired by the success of transfer learning in previous medical mask detectors [26]–[28], [30], [31], we apply weights pre-trained on COCO [23] to PWMFD [17]. This technique is known for significantly decreasing training time and the need for a large dataset [38], both crucial in our case. We tested various training schemes for YOLOv5s on PWMFD: without transfer learning using random initial weights (row 1 in Table II), and with transfer learning using the COCO weights (rows 2-4 in Table II), while experimenting with freezing the weights of different layers before training. The highest mAP (38%) was achieved by freezing the pre-trained backbone, that is, training only the head on PWMFD.

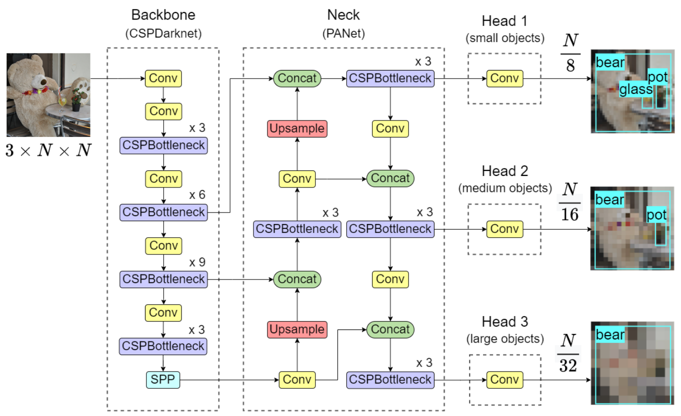


Fig. 4. Architecture of YOLOv5 [15] consisting of a CSPDarknet [14] type of backbone, an SPP [20] layer, a PANet [21] as the neck, and three heads for the detection of objects of different sizes.

2) Data Augmentations: To prevent overfitting, we utilize the following basic data augmentations: translation, scaling, flipping, and Hue-Saturation-Value (HSV) transformations. Furthermore, we assess the effect of two novel transformations, mosaic and mixup [39]. The first combines multiple images forming a mosaic, while the second stacks two images on top of one another with a degree of transparency. The potential benefits of mosaic and mixup for YOLO were explored in [14] and in [17] for mask detection. We trained YOLOv5s with three different data augmentation combinations as shown in Table III. Mosaic nearly doubled accuracy (mAP 67%), but the addition of mixup was not beneficial.

3) Attention Mechanism: In [17], by introducing two SE blocks to the backbone of YOLOv3 as an attention mechanism along with mixup and focal loss, the accuracy of YOLOv3 on PWMFD rose by 8.6%. The SE mechanism applies input-dependent weights to the channels of the feature map to create a better representation of the image. Focal loss is an improvement on Cross-Entropy loss that assists the model in focusing on hard misclassified examples during training. We apply the same strategy to YOLOv5. Our ablation study in Table IV shows that these optimizations did not affect speed but they impacted accuracy negatively. Therefore, they are not used in our final mask detector. Nevertheless, Fig. 5 illustrates higher accuracy for small and medium-sized objects when using SE with mixup.

D. Evaluation

Our final mask detector is based on YOLOv5s with transfer learning from COCO to PWMFD while freezing the backbone, and uses mosaic and other basic data augmentations. According to Table V, it is twice as accurate as the baseline YOLOv5s, while being equal in speed. At the same time, when compared using PWMFD, it is as accurate and more than two times faster on our own lower-end GTX 960 GPU than the state-of-the-art SE-YOLOv3 on a GTX 2070. This significant increase in speed gives room for its use in embedded devices with lower hardware capabilities, while still achieving real-time detection. Detection examples are illustrated in Fig. 6.

TABLE II
PERFORMANCE OF YOLOv5s [15] ON PWMFD [17] WITH VARIOUS TRANSFER LEARNING (TL) AND LAYER FREEZING SCHEMES.

	mAP	mAP@50	mAP@75
No TL	0.33	0.59	0.33
TL + No Freeze	0.35	0.60	0.39
TL + Freeze Backbone	**0.38**	**0.63**	**0.43**
TL + Freeze All[a]	0.03	0.10	0.01

[a]Except output layer.

TABLE III
PERFORMANCE OF YOLOv5s [15] ON PWMFD [17] WITH VARIOUS DATA AUGMENTATION COMBINATIONS.

	mAP	mAP@50	mAP@75
Basic Transformations[a]	0.38	0.63	0.43
Basic Trans. + Mosaic	**0.67**	**0.92**	**0.81**
Basic Trans. + Mosaic + Mixup	0.65	0.91	0.78

[a]Translation, scaling, flipping, and Hue-Saturation-Value.

TABLE IV
PERFORMANCE OF YOLOv5s [15] ON PWMFD [17] WITH SELECTED OPTIMIZATIONS FROM SE-YOLOv3 [17] (SQUEEZE-AND-EXCITATION (SE) ATTENTION MECHANISM, MIXUP, AND FOCAL LOSS).

	mAP	mAP@50	mAP@75	fps
No SE	**0.67**	0.92	**0.81**	69
SE	0.61	**0.93**	0.74	68
SE + Mixup	0.58	0.90	0.68	69
SE + Focal Loss	0.38	0.63	0.45	**71**
SE + Mixup + Focal Loss	0.37	0.62	0.43	70

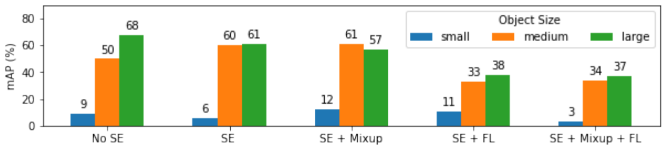


Fig. 5. Accuracy of YOLOv5s [15] on PWMFD [17] for three object sizes (small: area $< 32^2$, medium: $32^2 <$ area $< 96^2$, large: area $> 96^2$ in COCO evaluation metrics) with selected optimizations from SE-YOLOv3 [17].

TABLE V
PERFORMANCE OF OUR OPTIMIZED YOLOv5s COMPARED TO BASELINE YOLOv5 [15] AND SE-YOLOv3 [17].

	mAP	mAP@50	mAP@75	fps
SE-YOLOv3[a] [17]	0.66	**0.96**	0.79	28
YOLOv5s [15]	0.33	0.59	0.33	**69**
YOLOv5s + TL + Freeze BB + Mosaic	**0.67**	0.92	**0.81**	**69**

[a]With image size 320 and a GTX 2070 GPU.

V. Conclusion and Future Work

Evaluating object detection models fairly is a task that requires multiple parameters to be addressed besides accuracy. Our goal is to provide an informative analysis of fundamental object detectors that also includes speed, memory consumption, and computational and storage costs. Using our findings, we propose an optimized YOLOv5s-based model for real-time

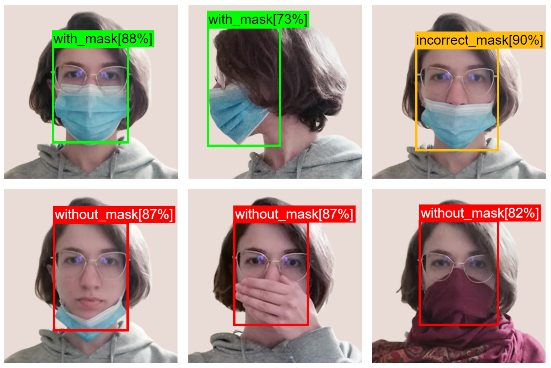


Fig. 6. Detection examples using our optimized YOLOv5s model.

mask detection to protect public health amidst the COVID-19 pandemic. Our optimizations led to increased accuracy (mAP 67%) that rivaled that of state-of-the-art SE-YOLOv3 on PWMFD, while being more than two times faster (69 fps). At the same time, applying the SE attention mechanism of SE-YOLOv3 to YOLOv5s along with mixup improved accuracy for small and medium-sized objects, but not large ones.

In the future, we would like to research optimizations to combat the side effects on large objects, thus improving total accuracy. Also, our model does not utilize an important characteristic of data streams, the relationship between consecutive frames. This could be achieved by exploring object detection models that implement this using Recurrent Neural Networks.

After the end of the pandemic, we are hopeful that with the help of our model the healthcare sector can be better prepared for a similar crisis. Our proposed optimizations may also be useful for other related problems such as face detection.

References

[1] Y. Lecun and Y. Bengio, "Convolutional networks for images, speech and time series," in *The Handbook of Brain Theory and Neural Networks*. The MIT Press, 1995.

[2] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv preprint arXiv:1905.05055*, 2019.

[3] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, 2019.

[4] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3296–3297.

[5] A. Srivastava *et al.*, "Performance and memory trade-offs of deep learning object detection in fast streaming high-definition images," in *Proc. IEEE Int. Conf. Big Data*, 2018, pp. 3915–3924.

[6] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015.

[7] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.

[8] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 580–587.

[9] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017.

[10] W. Liu *et al.*, "SSD: Single Shot Multibox Detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, p. 21–37.

[11] J. Redmon, K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 779–788.

[12] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7263–7271.

[13] ——, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[14] A. Bochkovskiy, C. Wang, and H. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[15] G. Jocher *et al.*, *ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support*, 2021.

[16] World Health Organization, "Advice on the use of masks in the context of COVID-19: interim guidance, 6 april 2020," Tech. Rep., 2020.

[17] X. Jiang, T. Gao, Z. Zhu, and Y. Zhao, "Real-time face mask detection method based on YOLOv3," *Electronics*, vol. 10, no. 7, 2021.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[19] R. B. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, 2015.

[21] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation." in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8759–8768.

[22] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) challenge." *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.

[23] T. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[24] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016.

[25] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.

[26] B. Qin and D. Li, "Identifying facemask-wearing condition using image super-resolution with classification network to prevent COVID-19," *Sensors*, vol. 20, no. 18, 2020.

[27] X. Fan and M. Jiang, "RetinaFaceMask: A single stage face mask detector for assisting control of the COVID-19 pandemic," in *Proc. IEEE Int. Conf. Syst. Man Cybern. Syst.*, 2021, pp. 832–837.

[28] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic," *Measurement*, vol. 167, 2021.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[30] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection," *Sustainable Cities and Society*, vol. 65, 2021.

[31] G. Jignesh Chowdary, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, "Face mask detection using transfer learning of InceptionV3," in *Proc. Int. Conf. Big Data Analytics*, 2020, p. 81–90.

[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016.

[33] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.

[34] A. Kumar, A. Kalia, A. Sharma, and M. Kaushal, "A hybrid tiny YOLO v4-SPP module based improved face mask detection vision system," *J. Ambient Intell. Human. Comput.*, 2021.

[35] A. Howard *et al.*, "Searching for mobilenetv3," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019.

[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[37] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Proc. Artif. Intell. and Mach. Learn. for Multi-Domain Oper. Appl.*, 2019, pp. 369–386.

[38] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Proc. Int. Conf. Artif. Neural Netw.*, 2018, pp. 270–279.

[39] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.