



Article

Object Detection Models and Optimizations: A Bird's-Eye View on Real-Time Medical Mask Detection

Dimitrios A. Koutsomitropoulos *  and Ioanna C. Gogou 

Computer Engineering and Informatics Department, University of Patras, 26504 Patras, Greece; gogou@ceid.upatras.gr

* Correspondence: kotsomit@ceid.upatras.gr

Abstract: Convolutional Neural Networks (CNNs) are well-studied and commonly used for the problem of object detection thanks to their increased accuracy. However, high accuracy on its own says little about the effective performance of CNN-based models, especially when real-time detection tasks are involved. To the best of our knowledge, there has not been sufficient evaluation of the available methods in terms of their speed/accuracy trade-off. This work performs a review and hands-on evaluation of the most fundamental object detection models on the Common Objects in Context (COCO) dataset with respect to this trade-off, their memory footprint, and computational and storage costs. In addition, we review available datasets for medical mask detection and train YOLOv5 on the Properly Wearing Masked Faces Dataset (PWMFD). Next, we test and evaluate a set of specific optimization techniques, transfer learning, data augmentations, and attention mechanisms, and we report on their effect for real-time mask detection. Based on our findings, 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 while maintaining the same level of mean Average Precision (67%).



Citation: Koutsomitropoulos, D.A.; Gogou, I.C. Object Detection Models and Optimizations: A Bird's-Eye View on Real-Time Medical Mask Detection. *Digital* **2023**, *3*, 172–188. <https://doi.org/10.3390/digital3030012>

Academic Editors: Mirjana Ivanović, Richard Chbeir and Yannis Manolopoulos

Received: 26 May 2023

Revised: 17 June 2023

Accepted: 27 June 2023

Published: 1 July 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: real-time object detection; medical mask detection; video surveillance; YOLOv5; PWMFD; COVID-19

1. 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 toward 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, compares models on different machine learning frameworks, or excludes newer models, resulting in indefinite results.

This work focuses on giving a solution to this problem by reviewing and evaluating 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 footprint, computational and storage costs, 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.

In addition, we review and compare available datasets for medical mask detection; among the models we evaluate, we choose YOLOv5 as a highly efficient one to train for real-time medical mask detection on a topical and novel dataset that has yet to be extensively tested, the Properly Wearing Masked Faces Dataset (PWMFD) [16]. 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 World Health Organization (WHO), the use of medical masks combined with other health measures is recommended for the containment of the virus [17]. In this context, we propose an optimized real-time detector of correctly and incorrectly worn medical masks. Next, we review, test, and investigate various optimization techniques used before in medical mask detection and report on their effect, including transfer learning, data augmentations, and attention mechanisms. 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:

- A review and evaluation of state-of-the-art object detectors and an analysis of their speed/accuracy trade-off, 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 are evaluated for the first time on the newly developed Properly Wearing Masked Faces Dataset (PWMFD) [16]. Furthermore, the effects of transfer learning, data augmentations, and attention mechanisms are assessed for medical mask detection.
- A real-time medical mask detection model based on YOLOv5 is proposed that surpassed more than 2 times in speed (69 fps) the state-of-the-art model SEYOLOv3 [16] on the PWMFD while maintaining the same level of the 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.

A preliminary version of this paper has appeared in [18]. In comparison, this article shows the following:

1. It presents new evaluation results of object detection models in terms of their accuracy and speed vs. computational performance (GFLOPS).
2. It includes new results demonstrating the (detrimental) effect of augmenting the model with Transformer Encoder (TE) Attention blocks.
3. It involves a more thorough description of the detection models reviewed and evaluated.
4. It includes a new section on the characteristics, availability, and usage of datasets suitable for medical mask detection.

All results are reproducible through our open-source code on GitHub (<https://github.com/joangog/object-detection> accessed on 1 June 2023) as well as the measurements' data. The weight file of our medical mask detector is also available on GitHub (<https://github.com/joangog/object-detection-assets> accessed on 1 June 2023) and on Hugging Face (<https://huggingface.co/joangog/pwmfd-yolov5> accessed on 1 June 2023).

The rest of this paper is organized as follows: Section 2 reviews the current state of the art for object and mask detection and describes models from an architectural point of view. Section 3 is devoted to the comparative, hands-on evaluation of several object detection models, employing a uniform evaluation testbed and metrics and discussing observed tradeoffs. Section 4 presents a qualitative review of datasets that can be leveraged for implementing effective mask detection pipelines. The results of testing various optimizations for this task on well-performing models are shown in Section 5 along with our final proposed model. Finally, Section 6 summarizes the lessons learned and the outlook of our work.

2. Related Work

2.1. Object Detection Models

The development of AlexNet [19] in 2012 paved the way toward 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 [20], 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 CSPDarknet53 backbone, a Spatial Pyramid Pooling (SPP) [21] layer, and a Path Aggregation Network (PANet) [22]. 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. As of January 2023, YOLOv8 was released. Early results appearing in Github (<https://github.com/ultralytics/ultralytics> accessed on 1 June 2023) show a consistent improvement to previous versions, both in terms of accuracy and speed; however, no scientific paper has been published reporting and/or comparing the results. 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.

2.2. Models Description

The architecture of the models to be evaluated is described below, in terms of their backbone, number of parameters, and input size. The PyTorch Torchvision package provides SSD in two variants. The first is a lightweight version of SSD, the SSDlite320, with the backbone of the MobileNetV3-Large [23], which is optimized for mobile devices with a very small number of parameters, and an input image size of 320×320 . The second variant, the SSD300, has as its backbone the larger VGG16 network with 16 layers of neurons (13 conv and 3 fc leaving out the 5 pooling layers) and an input size of 300×300 . The SSD300 has 35.6 million parameters, while the SSDlite320 has about a tenth of them, i.e., 3.4 million.

Implementing RetinaNet in Torchvision uses the backbone ResNet-50 consisting of 50 layers and 34 million parameters. For Faster R-CNN, two variants with different backbones are provided, one with MobileNetV3-Large and the second with ResNet-50. For the MobileNet variant, there are two versions with different input sizes, one with 800×800 and the other with 320×320 . Also included is Faster R-CNN's version for the segmentation problem, the R-CNN Mask with Resnet-50 backbone, which produces simultaneously segmentation masks as well as bounding boxes. Mask R-CNN has a few more parameters than Faster R-CNN due to its additional functionality.

In the PyTorch implementation of G. Jocher's YOLOv3, the backbone of Darknet53 is used, as in the original implementation in the Darknet framework. Two further variants are provided, YOLOv3-tiny and YOLOv3-spp. YOLOv3-tiny is a smaller version of YOLOv3, which replaces some conv layers of the backbone with pooling layers and removes one of the three heads of YOLOv3, the one for small objects, to reduce computational costs. It consists of only 8.8 million parameters compared to YOLOv3 which has 61.9 million. YOLOv3-spp is virtually the same as YOLOv3, but one of the conv layers has been replaced by an SPP layer which helps the network to perceive the characteristics of the image at various levels of resolution, which only slightly increases the number of parameters to 63 million.

For YOLOv4, the PyTorch implementation by T. Xiaomo is used, which employs the improved backbone of CSPDarknet53, as in the original YOLOv4 implementation of the

Darknet framework, and in total consists of 64.4 million parameters. Finally, G. Jocher's YOLOv5 is distributed in five variants, n, s, m, l, and x, depending on the size of the network, of which we will consider n, s, m, and l. The number of its parameters ranges from 1.9 million up to 47 million depending on the variant. YOLOv5 uses a backbone based on CSPDarknet53 with some modifications in the implementation of the CSP Bottleneck. All the above implementations of YOLO can accept any input size. However, in order to ensure the validity of the results, we define the input size as the size of the data with which the pretrained models we use were trained. This size is 640×640 for YOLOv3 and v5, and 608×608 for YOLOv4.

2.3. 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 the 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) [24] and Common Objects in Context (COCO) [25] 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 the 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) [26] were evaluated on COCO [25] based on their speed/accuracy trade-off while testing different backbones, image resolutions, and number of region proposals. In [5], a similar analysis was conducted using the same models with the addition of Mask R-CNN [7] and SSDlite [27]. The aspect of memory consumption and detection speed on different devices was investigated as well. Nevertheless, both studies did not take into account newer models, such as YOLOv5 [15] and RetinaNet [9].

2.4. 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 [28] was designed and trained using transfer learning. In 2021, RetinaFaceMask [29] 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 [30] combining ResNet-50 [31] with a Support Vector Machine (SVM) was proposed, after being trained on both real-world and synthetic data using transfer learning. Later, in [32], the authors replaced the SVM with YOLOv2 [12]. A medical mask detector published in [33] was based on Inception-v3 [34] 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 [16] 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 [35] to YOLOv3, achieving a 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 [36]. It significantly reduced the training time while increasing the accuracy compared to the original tiny-YOLOv4, but the aspect of real-time detection was not considered.

3. Materials and Methods

3.1. Evaluation Methodology of Object Detection Models

We evaluate the object detection models shown in Table 1 in terms of memory consumption, computational and storage costs, 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 utilize the same framework, dataset, and GPU.

Table 1. Models evaluated on the COCO2017 dataset.

Model	Backbone	Image Size ^a
Faster R-CNN [6]	MobileNetV3-Large [23]	800
Faster R-CNN	MobileNetV3-Large	320
Faster R-CNN	ResNet-50 [31]	800
Mask R-CNN [7]	ResNet-50	800
RetinaNet [9]	ResNet-50	800
SSD [10]	VGG16 [37]	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 N × N.

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 (<https://github.com/ultralytics/yolov3> accessed on 1 June 2023; <https://github.com/Tianxiaomo/pytorch-YOLOv4> accessed on 1 June 2023; <https://github.com/ultralytics/yolov5> accessed on 1 June 2023).
2. Dataset: The evaluation is performed on the val subset of the COCO2017 [25] dataset. COCO appears to be among the de facto standards for measuring accuracy and speed in modern object detection models as it strikes a balance between manageable size and adequate image scene density. It contains 118,000 examples for training and 5000 for validation belonging to 80 classes of everyday objects. In addition, COCO training has been extensively used in the past as a basis for transfer learning on the specific problem of medical mask detection. No training phase is performed as all models are already pretrained 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 (<https://github.com/sovrasov/flops-counter.pytorch> accessed on 1 June 2023) 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 (<https://cocodataset.org/#detection-eval> accessed on 1 June 2023), Average Precision (AP) is calculated using 101-point interpolation on the area under the P-R (Precision–Recall) curve, as follows:

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

where

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

3.2. Memory Footprint

In Figure 1a, the higher the number of parameters and image size, the more GPU memory the model consumes. Interestingly, YOLOv4 has a 925 MB memory footprint that is 3 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.

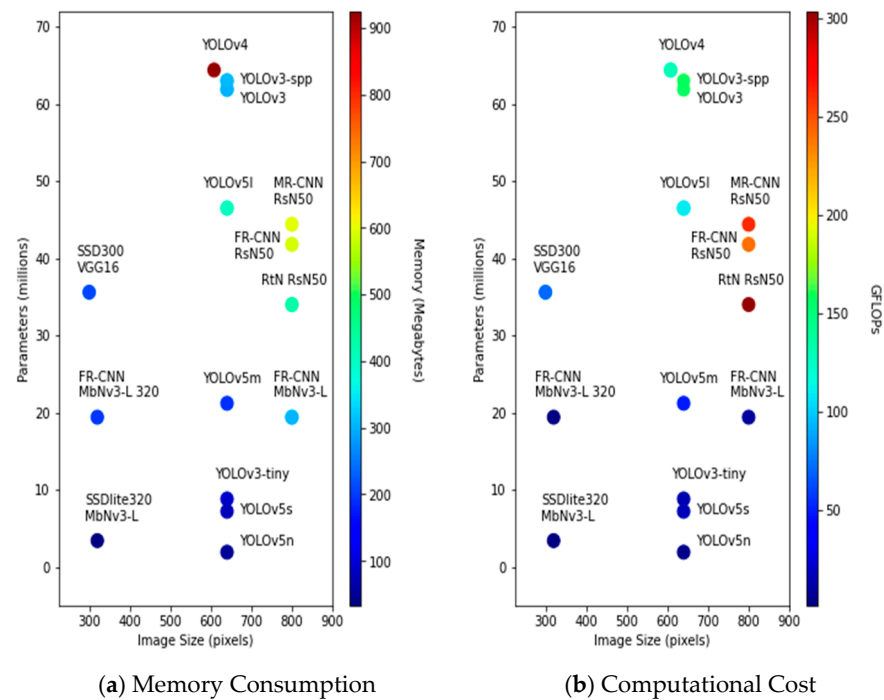


Figure 1. Memory footprint in Megabytes (a) and computational cost in GFLOPs (b) in relation to parameter count and image size of object detection models evaluated on the COCO2017 [25] dataset.

3.3. Computational Performance and Storage Costs

Firstly, the number of GFLOPs of a model is closely related to its number of parameters and image size. For instance, in Figure 1b, between the two implementations of Faster R-CNN with MobileNet-v3 and ResNet-50, the second executes 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, prove 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 costs almost 6 times more GFLOPs

than the one with 320. The sole exception to the rule is RetinaNet and YOLOv3 which, despite having the same image size and fewer parameters than Faster R-CNN ResNet50 and YOLOv4, respectively, execute more GFLOPs than their counterpart.

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

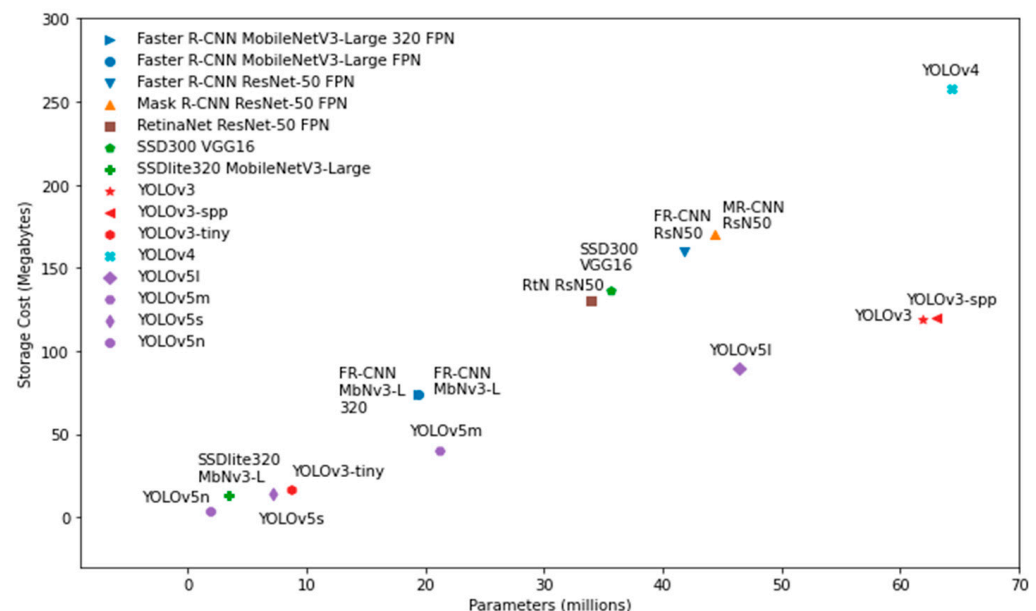


Figure 2. Relationship between storage cost in Megabytes and parameter count of object detection models evaluated on the COCO2017 [25] dataset.

3.4. Accuracy and Speed

For a fair comparison of the models' accuracy, it is necessary to evaluate this metric in terms of speed. There is no point in ranking the models based on their accuracy, if we do not understand the larger context in which a model manages to provide its degree of accuracy. This means that if a model achieves high accuracy but at the same time sacrifices a significant part of its speed for this purpose, then it becomes unsuitable for the problem of real-time detection. Accordingly, the same is true for the reverse.

Therefore, in Figure 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 Figure 3. All variations of YOLOv5 provide the best balance between accuracy and speed, whereas RetinaNet, SSD, SSDlite, and YOLOv3-tiny rank last.

3.5. Accuracy and Speed vs. Computational Performance

In Figure 4, we demonstrate the relationship between mAP and GFLOPs for each model. We notice that the higher the count of GFLOPs, the higher the accuracy, forming a hyperbolic curve of " $y = -\alpha/x + \beta$ " between the two measures. Ideally, we strive to find a model that achieves the highest possible mAP for the lowest possible GFLOPs. According to this, the best performance is attained by the models of the YOLOv5 family, especially the medium and large variants, with Faster R-CNN MobileNetV3-Large rivaling the smaller variants. The worst mAP/GFLOPs balance is observed for YOLOv3-tiny, which shows the lowest mAP in comparison to other models of a similar level of GFLOPs, and RetinaNet, which executes far more GLOPs than other models of the same order of mAP.

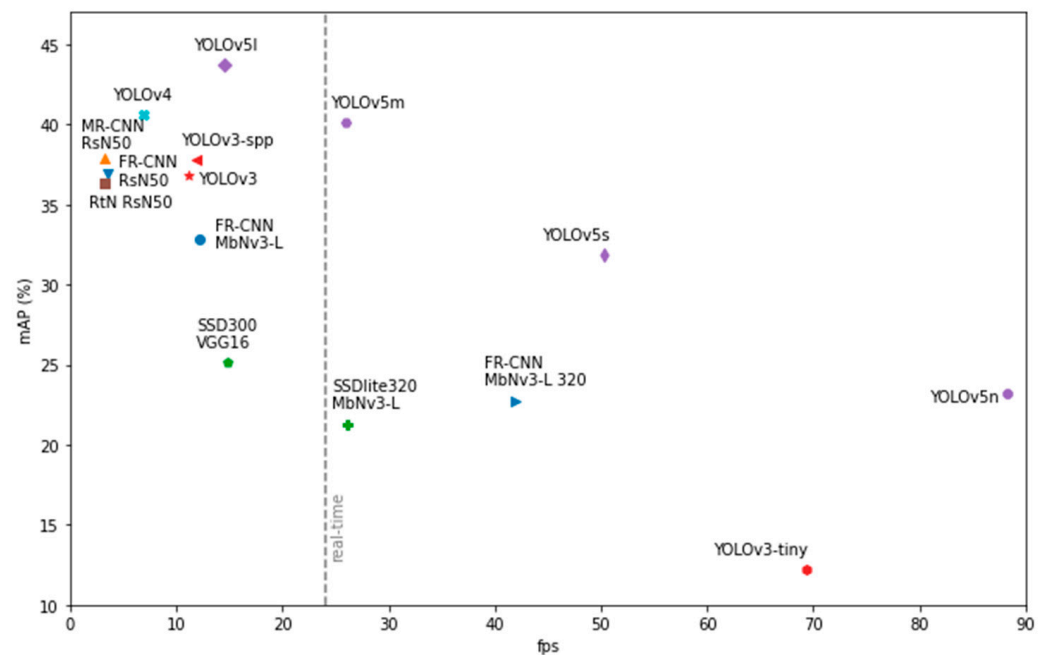


Figure 3. Speed (fps)/accuracy (mAP) trade-off of object detection models evaluated on the COCO2017 [25] dataset.

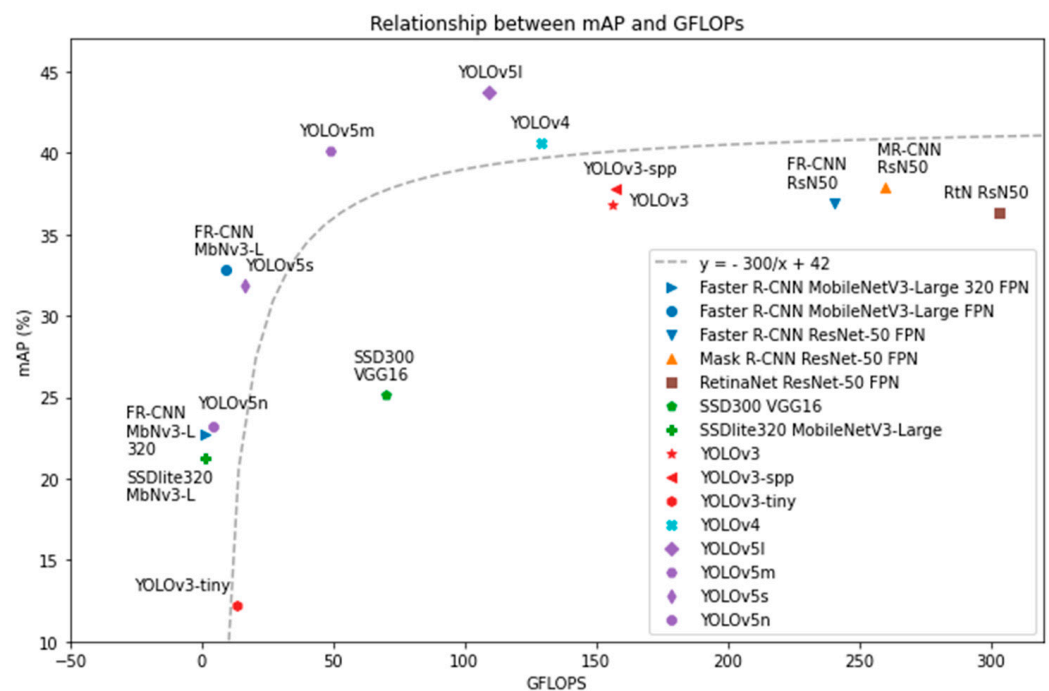


Figure 4. Relationship between mAP and GFLOPs for various object detection models.

Figure 5 illustrates the relationship between inference speed in fps and GFLOPs. We observe that it is approximated by a hyperbolic curve of the form “ $y = \alpha/x$ ”. In other words, as the GFLOPs of a model increase, the fps decrease. Nevertheless, achieving the highest fps possible with the least amount of GFLOPs is favorable. Thus, we conclude that the most efficient model in terms of speed is YOLOv5n, with YOLOv3-tiny ranking second. The implementation of RetinaNet is the least efficient with the largest number of GFLOPs and the least fps.

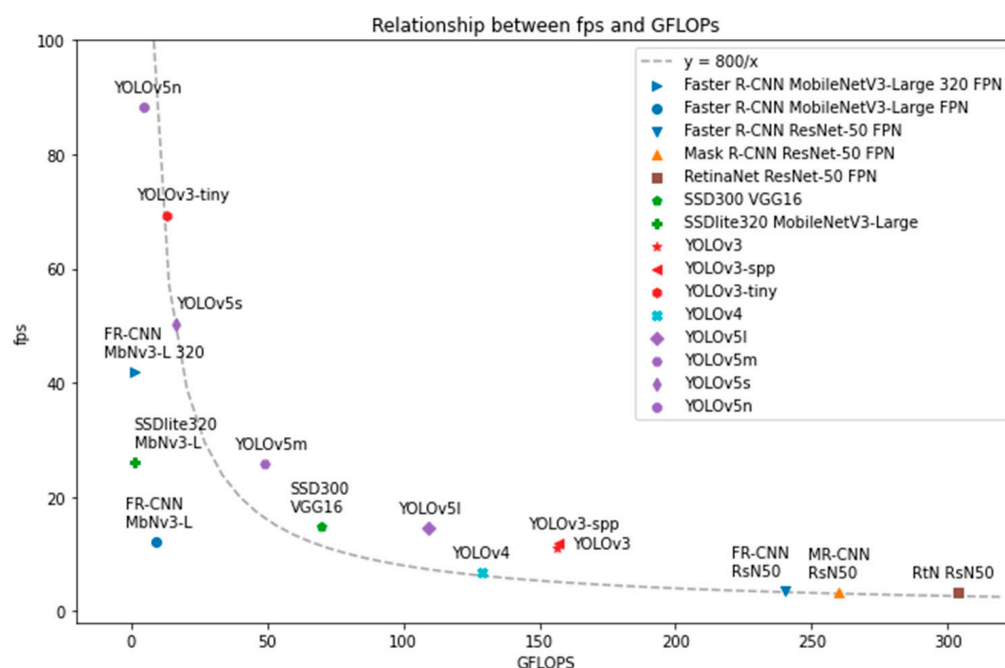


Figure 5. Relationship between fps and GFLOPs for various object detection models.

4. Datasets for Medical Mask Detection

For the problem of medical mask detection, we performed an extensive survey of the available datasets. The choice of a complex and diverse set is crucial for the development of a robust model. Although a model may exhibit maximum accuracy in a simplistic set, this does not necessarily hold in real conditions. The first datasets of medical masks have been appearing since 2017. However, since the outbreak of the COVID-19 pandemic, they have multiplied rapidly in number and some of them have already been used in medical mask detection systems. These datasets are analyzed below.

4.1. MAFA and MAFA-FMD

The Masked Faces (MAFA) dataset [38] was developed in 2017 by S. Ge et al. as the first dataset of faces with masks. It includes 30,811 non-synthetic images and a total of 35,806 masked faces divided across the number of images. On average, the faces of the set have a size of 143×143 pixels. A distinct characteristic of the dataset lies in the fact that the annotations of faces, in addition to the bounding box that encloses the face, include information about the orientation of the face, the degree of mask overlap, and the type of mask. In terms of orientation, faces are classified as left, left-front, front, right-front, and right, with the majority (71%) belonging to the front. In terms of degree of overlap, they are divided into weak, medium, and strong, depending on the number of areas they cover on the face, with the majority belonging to medium (81%). Finally, regarding the type of mask, the faces are grouped into simple, complex, body, and hybrid. In essence, "mask" is considered not only a medical mask but also a garment or part of the body, such as a hand.

In 2019, following the outbreak of COVID-19, the Masked Faces for Face Mask Detection Dataset (MAFA-FMD) [29] was formed as an improvement on MAFA. In particular, the labels of the original set were modified with the aim of specializing it in medical conditions, and the labels of persons who had not already been tagged were added. MAFA-FMD contains 56,024 identifications of persons who, unlike MAFA, also include faces without a mask at all. Faces are organized into three groups depending on whether a strictly medical mask has been placed correctly, wrongly, or not at all. Body parts or clothing used to cover the face are not considered valid masks and faces are classified as maskless. At the same time, the dataset was augmented with labels for low-resolution face samples i.e., smaller than 32×32 pixels. One disadvantage of MAFA-FMD is the imbalance of class samples,

most notably the misplaced mask class with only 1388 samples compared to the other two classes with 28,233 and 26,463 samples, respectively.

4.2. RMFD and SMFRD

In 2020, a large dataset of medical masks was issued by Z. Wang et al., the Real-World Masked Face Recognition Dataset (RMFRD) [39]. RMFRD contains 95,000 images of 525 different public faces in frontal view, of which 5000 contain a face with a medical mask and 90,000 without. Alongside this set, the Simulated Masked Face Recognition Dataset (SMFRD) [39] was published. It was used to exploit already existing large-scale face datasets. For its construction, the creators placed medical masks on each face of such sets, in an automatic way, after developing the appropriate software. In total, it contains 500,000 images of 10,000 different people with one face in each image. The advantage of synthetic sets is that they enable the usage of large-scale datasets of common problems in our own, especially in cases where the problem is new and sufficient data do not yet exist. However, the introduction of synthetic information requires the researcher to take some initiatives based on their own perception about what data the problem needs. Thus, there is a danger that the model will form biases during training. For example, in the SMFRD set, synthetic masks added to faces are drawn from only one image of a medical mask. Therefore, if the model is faced with the detection of masks of various colors, shapes, and textures, it is very likely that it will not be able to generalize sufficiently. In conclusion, the use of synthetic sets must be accompanied by the use of image sets from real life to maximize the ability of the model to generalize. Both RMFRD and SMFRD categorize faces into only two classes, those who wear a mask and those who do not. However, the labels of the two sets do not include bounding boxes, so the sets are not suitable for detecting masks, only for recognition.

4.3. MaskedFace-Net

In 2020, MaskedFace-Net [40] was developed by A. Cabani et al., and it is the largest synthetic dataset of masks differentiating correctly and incorrectly placed medical masks. The facial images were drawn from the Flickr-Faces-HQ (FFHQ) set comprising faces of high diversity in terms of age, ethnicity, and environmental conditions. Then, a medical mask image was automatically added to them with varying degrees of overlap using a machine learning model to identify the parts of the face to which the mask attaches. MaskedFace-Net contains a total of 137,016 images and is a synthesis of two subsets developed by the creators, the Correctly Masked Face Dataset (CMFD) and the Incorrectly Masked Face Dataset (IMFD), with 67,193 and 69,823 images, respectively. In the IMFD set, faces are grouped into three subsets which are differentiated according to whether the mask does not cover the nose (80%), whether it does not cover the chin (10%), and whether it does not cover the nose and mouth (10%). Unlike the large-scale SMFRD dataset, MaskedFace-Net can be used for both recognition and detection and provides detailed analysis of faces with misplaced masks. Of course, like SMFRD, as a synthetic dataset, it is proposed to be used in conjunction with a non-synthetic set.

4.4. PWMFD

The Properly Wearing Masked Face Detection Dataset (PWMFD) [16] is a dataset of faces with medical masks developed by X. Jiang et al. in 2021. It includes 9205 non-synthetic images of faces from multiple sources. Specifically, 3615 were collected from the internet, 2951 from the WIDER FACE dataset, 2581 from MAFA, and 58 from RMFRD. There are three classes used to classify faces: faces with correct mask placement, with wrong mask placement, and without a mask. In the maskless class, not only persons who did not wear a mask were placed but also persons with body parts or objects covering a part of them. Thus, the set is an important tool to ensure the proper use of medical masks, even in cases of attempts to deceive the system. Of the 9205 images, a total of 18,532 faces are highlighted: 7695 in the correct placement class, 366 in the wrong placement class, and 10,471 in the

maskless class. We observe a balance between correct and maskless classes, but the number of samples in the wrong mask class is relatively incomplete.

For implementing a medical mask detection and recognition system, we select PWMFD among the datasets reviewed. Initially, due to the nature of the problem, it is necessary for the dataset chosen to include markings for both the class and the coordinates of each face. Therefore, sets exclusively specialized in either detection or recognition, such as RMFRD and SMFRD, are not readily suitable. In addition, it is important that the dataset contains realistic and diverse images so that the model can generalize with a high degree of accuracy after training. Synthetic sets have limitations in this respect, such as SMFRD and MaskedFace-Net sets, which use a single mask format from a particular image to compose the set. To avoid prolonged training times, we need a balance between the dataset size and the percentage of realistic vs. synthetic images it contains rather than an arbitrary expansion with synthetic images. At the same time, PWMFD is also an average solution in terms of sample count per image, averaging two faces per image. A large count gives the model the opportunity to train in scenes with dense crowds of people, something that responds to realistic situations. After all, the PWMFD set is a synthesis of subsets of MAFA and RMFRD, enriched with additional data. Finally, as already mentioned, PWMFD allows the detection of misplaced masks as well, a significant advantage in real medical applications.

5. Results of Optimizations for Medical Mask Detection

5.1. Configuration

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 [16] dataset, using its train and validation subsets to train and evaluate our model, respectively. It includes 9205 real-life images with 18,532 annotations of faces belonging to three classes: “with mask”, “incorrect mask”, and “without mask.”

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

Training and Evaluation: During training, a batch size of 32 was used. Training lasted for 50 epochs, as more resulted in overfitting. The learning rate was updated according to the OneCycleLR [41] policy, with values in the range of [0.001, 0.01]. As an optimizer, Stochastic Gradient Descent was employed with a value of 0.937 for momentum and 0.0005 for weight decay. Cross-Entropy was used for classification loss and Complete Intersection over Union (CIoU) for localization loss. After training, the final weights were selected from the epoch with the highest mAP. During evaluation, inference was performed with a batch size of one to mimic the sequential input of data in real time. The COCO mAP and fps were measured as metrics.

5.2. Implementing Optimizations for Medical Mask Detection

According to our study in Section 3, 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 Figure 6. To implement our medical mask detector, we chose to train its small variant, YOLOv5s, on PWMFD [16] with an image size of 320, achieving 33% mAP and 69 fps. To elevate its performance, we experimented with various optimizations during training.

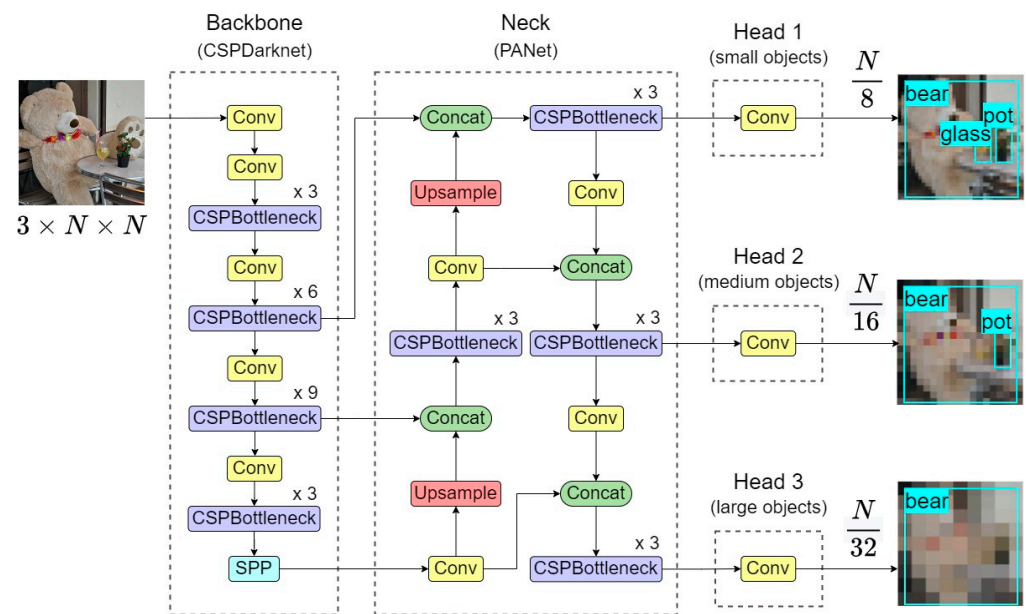


Figure 6. Architecture of YOLOv5 [15] consisting of a CSPDarknet [14] type of backbone, an SPP [21] layer, a PANet [22] as the neck, and three heads for the detection of objects of different sizes.

5.3. Transfer Learning

Inspired by the success of transfer learning in previous medical mask detectors [28–30,32,33], we applied weights pretrained on COCO [25] to PWMFD [16]. This technique is known for significantly decreasing training time and the need for a large dataset [42], 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 2) and with transfer learning using the COCO weights (rows 2–4 in Table 2), while experimenting with freezing the weights of different layers before training. The highest mAP (38%) was achieved by freezing the pretrained backbone, that is, training only the head on PWMFD.

Table 2. Performance of YOLOv5s [15] on PWMFD [16] 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.

5.4. Data Augmentations

To prevent overfitting, we utilized the following basic data augmentations: translation, scaling, flipping, and Hue–Saturation–Value (HSV) transformations. Furthermore, we assessed the effect of two novel transformations, mosaic and mixup [43]. The first combined multiple images forming a mosaic, while the second stacked two images on top of one another with a degree of transparency. The potential benefits of mosaic and mixup for YOLO were explored in [14,16] for mask detection. We trained YOLOv5s with three different data augmentation combinations as shown in Table 3. Mosaic nearly doubled the accuracy (mAP 67%), but the addition of mixup was not beneficial.

Table 3. Performance of YOLOV5S [15] on PWMFD [16] 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

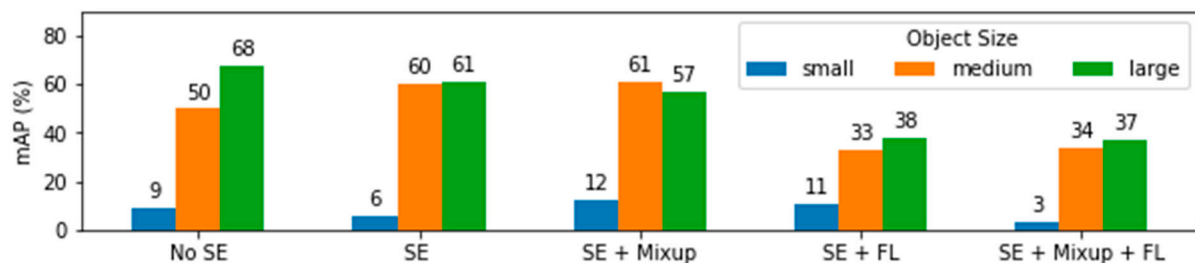
^a Translation, scaling, flipping, and Hue-Saturation-Value.

5.5. Attention Mechanism with Squeeze-and-Excitation (SE) Block

In [16], 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 raised 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 applied the same strategy to YOLOv5. Our ablation study in Table 4 shows that these optimizations did not affect speed, but they impacted accuracy negatively. Therefore, they were not used in our final mask detector. Nevertheless, Figure 7 illustrates higher accuracy for small and medium-sized objects when using SE with mixup.

Table 4. Performance of YOLOV5S [15] on PWMFD [16] with selected optimizations from SE-YOLOV3 (Squeeze-and-Excitation 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

**Figure 7.** Accuracy of YOLOv5s [15] on PWMFD [16] for 3 object sizes (small: area < 32², medium: 32² < area < 96², large: area > 96² in COCO evaluation metrics) with selected optimizations from SE-YOLOv3 [16].

5.6. Attention Mechanism with Transformer Encoder (TE) Block

In the TPH-YOLOv5 model [44], the basic architecture of YOLOv5 was enhanced by adding a Transformer Encoder block as an attention mechanism at the end of the backbone and the beginning of each head. The purpose of the backbone addition is to enhance feature maps with information about the general context in which they appear, i.e., their relationship to neighboring objects. The purpose of the head addition is to improve feature maps on each output size scale. In addition, by using TE blocks, TPH-YOLOv5 improved the accuracy of locating small, densely placed objects. Based on the architecture of TPH-YOLOv5, we evaluated the insertion of TE blocks into YOLOv5s at the end of the backbone and at the beginning of each head. The structure of the TE block was the one implemented within the C3TR block in the official code repository of YOLOv5. The new C3TR blocks replaced the C3 blocks of YOLOv5 located at the end of the backbone and at the beginning of each head. A C3TR block, like the C3, consists of a CSPNet, except that in the former a TE block is nested and in the latter a bottleneck block.

Table 5 shows the influence of using TE blocks on the backbone, heads, and the architecture as a whole. We note that using TE blocks in each case did not improve accuracy. In fact, due to the addition of the new units, the speed dropped. Adding three TE blocks to the heads reduced the speed much more (−13 fps) than adding to the backbone (−1 fps). The drop in accuracy was not as dramatic (−5%) but still greater than the drop after adding to the backbone (−1%). The use of TE blocks across the architecture had the greatest decrease in accuracy (−7%) and speed (−14 fps). For a loose IoU boundary, we observed that using TE blocks did not affect accuracy as negatively as with a tighter limit, meaning that the generated bounding boxes with TE blocks are less “tight” around each object. We also noticed that using TE blocks solely on the backbone or heads improved the detection of medium objects but negatively affected accuracy on large and especially small objects. We conclude that for our problem, the use of the attention mechanisms we examined helped the model for medium objects in terms of accuracy, but not overall, and therefore we did not use them in the final model.

Table 5. Performance of YOLOV5S [15] on PWMFD [16] with and w/o TE blocks.

	mAP	mAP@50	mAP@75	fps
No TE Block	0.67	0.92	0.81	69
TE Block (Backbone)	0.66	0.90	0.78	68
TE Block (Heads)	0.62	0.92	0.76	56
TE Block (Backbone + Heads)	0.60	0.88	0.75	55

5.7. Final Optimized Model

Our final mask detector was 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 6, it is twice as accurate as the baseline YOLOv5s while being equal in speed. At the same time, when compared using PWMFD, it was 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 Figure 8.

Table 6. Performance of our optimized YOLOv5s compared to baseline YOLOv5 [15] and SE-YOLOv3 [16].

	mAP	mAP@50	mAP@75	fps
SE-YOLOv3 ^a [16]	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.

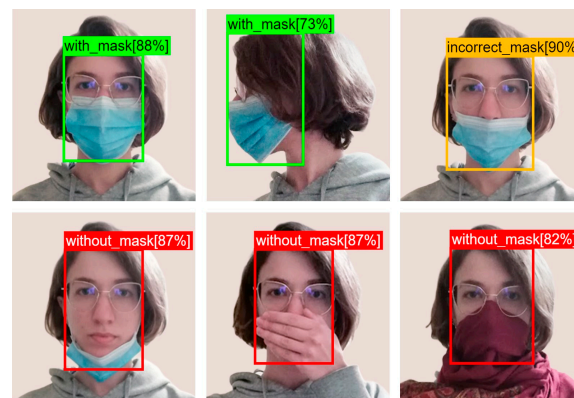


Figure 8. Detection examples using our optimized YOLOv5s model.

6. Discussion

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, accuracy, memory consumption, and computational and storage costs. YOLOv5 currently appears to provide the best balance between these parameters for real-time object detection. For the specific task of medical mask detection, our review and survey of currently available models and datasets can be useful for further designing end-to-end systems for public health administration and protection.

We also perform an evaluation of an inspiring set of potential optimizations for the task at hand. Using our findings, we propose an optimized YOLOv5s-based model for real-time mask detection to protect public health amidst the COVID-19 pandemic. Our optimizations led to increased accuracy (mAP 67%) that rivaled that of the 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 SEYOLOv3 to YOLOv5s along with mixup improved the 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. Moreover, 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. In addition, further optimized models such as YOLOv8 as well as Transformer-based models (ViTs) have yet to be fully evaluated specifically for the problem per se. Some of the optimizations we have tested and reported on have now been incorporated into v8, including mixup, for which we already concluded that it does not offer overall gains.

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.

Author Contributions: Conceptualization, D.A.K.; methodology, D.A.K. and I.C.G.; validation, I.C.G.; investigation, D.A.K. and I.C.G.; writing—original draft preparation, D.A.K. and I.C.G.; writing—review and editing, D.A.K. and I.C.G.; visualization, I.C.G.; supervision, D.A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available at <https://github.com/joangog/object-detection-assets>.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lecun, Y.; Bengio, Y. Convolutional networks for images, speech and time series. In *The Handbook of Brain Theory and Neural Networks*; The MIT Press: Cambridge, MA, USA, 1995.
2. Zou, Z.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *arXiv* **2019**, arXiv:1905.05055. [[CrossRef](#)]
3. Zhao, Z.; Zheng, P.; Xu, S.; Wu, X. Object detection with deep learning: A review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [[CrossRef](#)] [[PubMed](#)]
4. Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Korattikara, A.; Fathi, A.; Fischer, I.; Wojna, Z.; Song, Y.; Guadarrama, S.; et al. Speed/accuracy trade-offs for modern convolutional object detectors. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, Hawaii, 21–26 July 2017; pp. 3296–3297.
5. Srivastava, A.; Nguyen, D.; Aggarwal, S.; Luckow, A.; Duffy, E.; Kennedy, K.; Ziolkowski, M.; Apon, A. Performance and memory trade-offs of deep learning object detection in fast streaming high-definition images. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 3915–3924.
6. Ren, S.; He, K.; Girshick, R.B.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*; Neural Information Processing Systems Foundation, Inc.: Montreal, QC, Canada, 2015; Volume 28.
7. He, K.; Gkioxari, G.; Dollar, P.; Girshick, R.B. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2961–2969.

8. Girshick, R.B.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2014, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
9. Lin, T.; Goyal, P.; Girshick, R.B.; He, K.; Dollar, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision 2017, Venice, Italy, 22–29 October 2017.
10. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot Multibox Detector. In *ECCV 2016: Computer Vision—ECCV 2016*; Springer: Cham, Switzerland, 2016; pp. 21–37.
11. Redmon, J.; Divvala, K.; Girshick, R.B.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
12. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017, Honolulu, Hawaii, 21–26 July 2017; pp. 7263–7271.
13. Redmon, J.; Farhadi, A. YOLOv3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
14. Bochkovskiy, A.; Wang, C.; Liao, H.M. YOLOv4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
15. Jocher, G.; Stoken, A.; Chaurasia, A.; Borovec, J.; Kwon, Y.; Michael, K.; Changyu, L.; Fang, J.; Skalski, P.; Hogan, A.; et al. ultralytics/yolov5: v6.0-YOLOv5n ‘Nano’ Models, Roboflow Integration, TensorFlow Export, OpenCV DNN Support. 2021. Available online: <https://github.com/ultralytics/yolov5> (accessed on 1 June 2023).
16. Jiang, X.; Gao, T.; Zhu, Z.; Zhao, Y. Real-time face mask detection method based on YOLOv3. *Electronics* **2021**, *10*, 837. [[CrossRef](#)]
17. World Health Organization. *Advice on the Use of Masks in the Context of COVID-19: Interim Guidance*; World Health Organization: Geneva, Switzerland, 2020.
18. Gogou, I.C.; Koutsomitropoulos, D.A. A Review and Implementation of Object Detection Models and Optimizations for Real-time Medical Mask Detection during the COVID-19 Pandemic. In Proceedings of the 2022 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), Biarritz, France, 8–12 August 2022; pp. 1–6.
19. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)*; Curran Associates, Inc.: New York, USA, 2013; pp. 1097–1105.
20. Girshick, R.B. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
21. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [[CrossRef](#)] [[PubMed](#)]
22. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8759–8768.
23. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision 2019, Long Beach, CA, USA, 16–20 June 2019.
24. Everingham, M.; Gool, L.V.; Williams, C.K.I.; Winn, J.M.; Zisserman, A. The Pascal Visual Object Classes (VOC) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [[CrossRef](#)]
25. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P. Microsoft COCO: Common Objects in Context. In *ECCV 2014: Computer Vision—ECCV 2014*; Springer: Cham, Switzerland, 2014; pp. 740–755.
26. Dai, J.; Li, Y.; He, K.; Sun, J. R-FCN: Object detection via regionbased fully convolutional networks. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*; Curran Associates, Inc.: New York, NY, USA, 2016.
27. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018, Salt Lake City, UT, USA, 18–22 June 2018.
28. Qin, B.; Li, D. Identifying facemask-wearing condition using image super-resolution with classification network to prevent COVID-19. *Sensors* **2020**, *20*, 5236. [[CrossRef](#)] [[PubMed](#)]
29. Fan, X.; Jiang, M. RetinaFaceMask: A single stage face mask detector for assisting control of the COVID-19 pandemic. In Proceedings of the 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 17–20 October 2021; pp. 832–837.
30. Loey, M.; Manogaran, G.; Taha, M.H.N.; Khalifa, N.E.M. A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. *Measurement* **2021**, *167*, 108288. [[CrossRef](#)] [[PubMed](#)]
31. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
32. Loey, M.; Manogaran, G.; Taha, M.H.N.; Khalifa, N.E.M. Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection. *Sustain. Cities Soc.* **2021**, *65*, 102600. [[CrossRef](#)] [[PubMed](#)]
33. Chowdary, G.J.; Punn, N.S.; Sonbhadra, S.K.; Agarwal, S. Face mask detection using transfer learning of InceptionV3. In *BDA 2020: Big Data Analytics*; Springer: Cham, Switzerland, 2020; pp. 81–90.
34. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016, Las Vegas, NV, USA, 27–30 June 2016.
35. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018, Salt Lake City, UT, USA, 18–22 June 2018.

36. Kumar, A.; Kalia, A.; Sharma, A.; Kaushal, M. A hybrid tiny YOLO v4-SPP module based improved face mask detection vision system. *J. Ambient Intell. Human. Comput.* **2023**, *14*, 6783–6796. [[CrossRef](#)] [[PubMed](#)]
37. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
38. Ge, S.; Li, J.; Ye, Q.; Luo, Z. Detecting masked faces in the wild with LLE-CNNs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017, Honolulu, HI, USA, 21–26 July 2017; pp. 2682–2690.
39. Wang, Z.; Huang, B.; Wang, G.; Yi, P.; Jiang, K. Masked face recognition dataset and application. *IEEE Trans. Biom. Behav. Identity Sci.* **2023**, *5*, 298–304. [[CrossRef](#)]
40. Cabani, A.; Hammoudi, K.; Benhabiles, H.; Melkemi, M. MaskedFace-Net—A dataset of correctly/incorrectly masked face images in the context of COVID-19. *Smart Health* **2021**, *19*, 100144. [[CrossRef](#)] [[PubMed](#)]
41. Smith, L.N.; Topin, N. Super-convergence: Very fast training of neural networks using large learning rates. In Proceedings of the Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, Baltimore, MD, USA, 14–18 April 2019; pp. 369–386.
42. Tan, C.; Sun, F.; Kong, T.; Zhang, W.; Yang, C.; Liu, C. A survey on deep transfer learning. In *ICANN 2018: Artificial Neural Networks and Machine Learning—ICANN 2018*; Springer: Cham, Switzerland, 2018; pp. 270–279.
43. Zhang, H.; Cisse, M.; Dauphin, Y.N.; Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv* **2017**, arXiv:1710.09412.
44. Zhu, X.; Lyu, S.; Wang, X.; Zhao, Q. TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-Captured Scenarios. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, Montreal, QC, Canada, 10–17 October 2021; pp. 2778–2788.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.