Comparison of YOLOv3, YOLOv5s and MobileNet-SSD V2 for Real-Time Mask Detection

Rakkshab Varadharajan Iyer[1], Priyansh Shashikant Ringe[2], Kevin Prabhulal Bhensdadiya[3]

[1-3]Dept. of Electronics & Communication Engineering, Institute of technology, Nirma University, Gujarat, India

---***---

Abstract - *Today, deep learning evinces its true potential with a multitude of use-cases and is seminal in different technological domains. One of the more trending application of deep learning is object recognition and tracking. Recent developments have showed promising results concerning the same. This paper discusses and compares various systematic approaches that analyses images and determines if the person captured is wearing a face mask correctly, incorrectly and not wearing one at all. Mask detection is carried out on images, videos and real time surveillance using three widely used machine learning algorithms: YOLOv3, YOLOv5 and MobileNet-SSD V2. Each model detects the presence of mask on a person's face, which will be judged on the basis of their accuracy and how smoothly the video is processed. Performance of the three algorithms is determined for detecting the presence of face mask on a person in real time in terms of FPS of the results.*

Key Words: *YOLOv3; YOLOv5; MobileNet-SSD; mAP; FPS*

1. INTRODUCTION

In the final months of 2019, a new deadly virus was discovered in Wuhan, China which gripped the world by its throat and ruined countless lives and the economy. Governments worldwide started working on various strategies like social distancing, application of sanitizers and PPE kits to prevent spread. The Coronavirus disease 2019 (Covid-19) is a serious public health and economic issue because of its detrimental effects like, mortality, acute respiratory infections and financial crisis. The virus spreads rapidly in crowded environments and being in close contact with the host. Wearing masks was declared the first line of personal defense against this unseen evil and has been made mandatory by all governments across the globe. However, there is also a significant percentage of people in the world that refuse to comply. To enforce wearing masks, authorities have begun to impose fines on those who do not wear masks in public. This is effective when the number of people under observation are less, but if the crowds are huge, it becomes a bigger problem.

Mask detection ensures the rate of transmission is controlled in predicaments like a highly crowded public space and other areas with high risk of explosive transmission. The primary objective of this paper is to detect and track the presence of a masked face along with other conditions like, if the mask is worn improperly or not worn at all. Given an input, be it an image or a real-time video, a bounding box of the masked face is illustrated in the output based on YOLOv3, v5s and MobileNet-SSD V2.

Object detection due to its wide variety of possible use-cases is a deeper aspect of computer vision. Most mask detection techniques focus on face recognition and construction, paradigmatic of traditional machine learning algorithms. The focus of this paper is to solely detect and track people wearing/not wearing/wearing incorrectly face masks and compare the results on the basis of their accuracy and how smoothly it is able to process video (FPS), with the latter being considered a more deciding parameter for real-time detection.

2. EXISTING ALGORITHMS

The object detection consists of two parts: localization and classification. The detection pipeline starts by extracting the selective features (Haar, HOG, Convolutional layer) and then a localizer or classifier is used to classify the object. Generally, these localizer and classifier run over an image based on region proposal approach or in-sliding window technique over the image. Methods like Deformable Parts Models (DPM) are paradigmatic of sliding window approach and methods like R-CNN take advantage of region proposal approach to generate bounding boxes and thereafter, run a classifier over the expounded bounding boxes. Then post-processing is carried out to filter out duplicate bounding boxes.

The nature of the pipeline used in these methods is hard to optimize and very complex because in such systems each component is trained separately. But systems like YOLO have reformed object detection as a one-step, single-regression problem, by unifying in single network. Therefore, in such systems like YOLO, the algorithm performs calculations on an image to predict where they are, and classify those objects. Also, MobileNetV2 has shown good accuracy with low latency and low power models.

In this paper YOLOv3, YOLOv5s and MobileNet-SSD V2 systems have been compared to identify the best suitable algorithm for mask detection system.

2.1 YOLOv3

YOLOv3 performs both localization and classification with the help of one neural network only, which makes it one of the best for real time applications, thus making YOLOv3 faster as compared to other algorithms. This feature allows YOLOv3 to train on real time inputs and perform detections near accurately. YOLOv3 can reach speeds from 45 frames per seconds (fps) up to 155 fps [4]. It uses a hybrid approach from YOLOv2 and Darknet-19, wherein the image is passed once during the detection and output is generated in the upcoming stage, making YOLOv3 quicker than the R-CNN algorithm, which performs detection on different sections of the input image and subsequently multiple predictions are made on the input for the various regions. Because YOLOv3 is trained on full images without hard negative mining and using multi-scale training method, optimizing the performance of the algorithm. [7]

In this method, the input image is initially split-up into grids and these grids are further divided into several bounded boxes. The grid predicts the bounding boxes, along with their width and height. Sometimes the grids may not contain any object which results in an objectness score of zero. Situations like these may create instability and add error to model, but it can be countered by increasing the number of coordinates of the bounding boxes and thereby reducing the void space in an image. This alternative, improves the efficiency of object identification in the image. In a specific grid cell, YOLOv3 is capable of predicting and analyzing multiple bounding boxes. In this case, the objects in an image are predicted using multiple object predictors. [7]

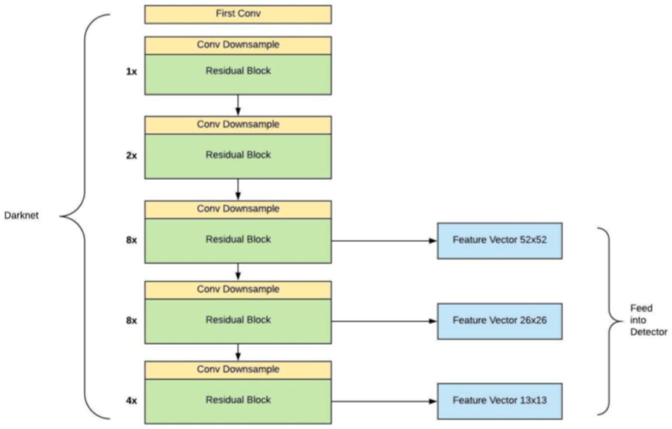


Figure -1: Multi-scale detector was appended aside network to make detection 3 times in 3 different scales

A seminal property of YOLOv3, is that it detects objects on three scales as portrayed in Fig 1. More specifically, YOLOv3 makes predictions at 82nd, 94th, and 106th layer, which are precisely provided by the stride of the model network, which are 32, 16, and 8 respectively. [4]

YOLOv2 uses a custom deep 30-convolutional layers for Darknet architecture, more than YOLOv1, which used 11 layers [4]. For deep neural networks, increased number of layers imply increased accuracy. However, the input image is downsampled when forwarded to deeper layers, entailing in loss of fine-grained features. This is why YOLOv2 often struggled with small object detections.

2.2 YOLOv5

YOLO has been dominating its field for a long time and there has been a major breakthrough in May 2020. Two updated and better versions of YOLO were introduced one after the other. One was the YOLOv4 developed by the conventional authors Joseph Redmon and Alexey Bochkovskiy [4], the other being the freshly released YOLOv5 by Glenn Jocher [3]. Not being the conventional author of the YOLO series, this new release was received with some controversy, but skipping past it, the v5 model has shown a substantial performance increase from its predecessors.
However, YOLOv5 possessed loads of advantages in engineering. The highly appreciated change being the usage of Python language instead of C as in its previous versions. That makes installation and integration with IoT devices a lot easier. In addition, the PyTorch community is also larger than the Darknet community, which means that PyTorch will receive more contributions and has a great growth potential in the future.

Along with the development of YOLO in 2016, many object detection algorithms with different approaches have achieved remarkable achievements as well. These advancements have formulated two concepts of architectural object detection: One-stage detector and Two-stage detector.

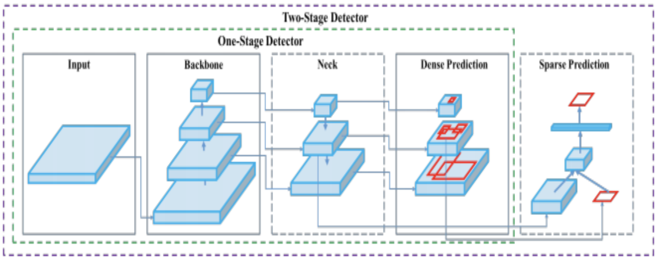


Figure 2. Two concepts of architectural object detection [6].

The YOLOv5 network consists of three main parts-

a. Backbone - A CNN layer aggregate image features at different scales.
b. Neck – Set of layers to combine image features and pass them forward to prediction.
c. Head - Takes features from the neck and performs localization and classification.

The common point of all object detection architectures is that the input image features will be compressed through a feature extractor (Backbone) and then forwarded to the object detector (including Detection Neck and Detection Head) as in Fig 2. Detection Neck (or Neck) works as a feature aggregator, tasked to combine and mix the features formed in the Backbone to prepare for the forthcoming step in the Detection Head (or Head) [3]. The difference here is that the

Head is responsible for detections along with classification and localization for each bounding box. The two-stage detector implements these 2 tasks separately and combines their results later (Sparse Detection), whereas the one-stage detector implements it at the same time (Dense Detection) also displayed in Fig 2. YOLO is a single-stage detector, therefore, You Only Look Once.

In the case of a one-stage detector, the function of the head is to perform dense predictions. The dense prediction is the final prediction composed of the prediction confidence score, the probability classes and a vector containing the predicted bounding box coordinates (center, height, width). YOLOv5 has an identical head to YOLOv3 for detection with the anchor-based detection steps, and 3-levels of detection granularity.

YOLOv5 comes with various versions, each having its own unique characteristic. These versions being:

1. yolov5-s - The small version
2. yolov5-m - The medium version
3. yolov5-l – The large version
4. yolov5-x - The extra-large version

The performance analysis of all these models as per Glenn Jocher is provided below in Fig 3.

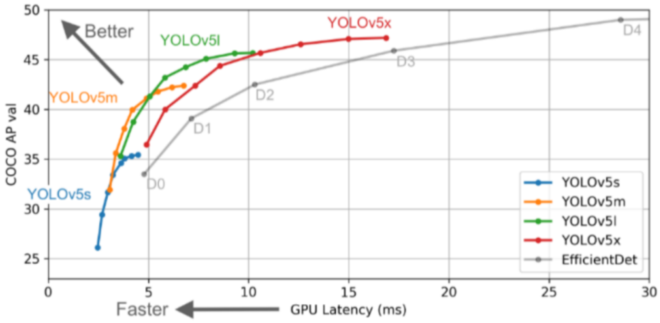


Figure -3: A comparative plot of performance of the YOLOv5 family [3]

Since this paper is focused on real-time detection, speed is a factor of utmost importance, hence the smallest version has been chosen as the representative of the YOLOv5 family for its performance analysis.

2.3 MobileNet-SSD V2

The general trend observed is that computer vision models are getting more deeper and complex in order to achieve greater accuracy. However, these advances are increasing the size and latency, and cannot be used on computationally handicapped systems.

In such cases, MobileNet comes handy. This is a model designed specifically for mobile and embedded applications requiring high speed. Its first version (MobileNetV1) had a depthwise separable convolution, which lowered the model size and complexity cost of the network to a decent level, to make it usable for low processing applications.

Thereafter in the second edition of the MobileNet family, an inverted residual structure is provided for much better modularity and this version has been named MobileNetV2. This has helped in removal of non-linearities in narrow layers resulting in cutting-edge performance for the above-mentioned applications. [5]

Around the time the first version of MobileNet was introduced, Google released Single Shot Detector (SSD) for applications heavily dependent on both speed and accuracy equally. As the name itself suggests, SSD essentially detected multiple objects in an image using a single shot.

MobileNet is a model catering to decent speed with its only drawback being its accuracy. SSD proved really helpful to the model since it got the means to improve its accuracy while maintaining the models' speed. SSD algorithm was designed in such a way that it could be integrated with various networks such as YOLO, MobileNet and VGG architecture. Thus, MobileNet was integrated with SSD for superior performances and it was termed MobileNet-SSD. This integrated architecture is shown in Fig 4.

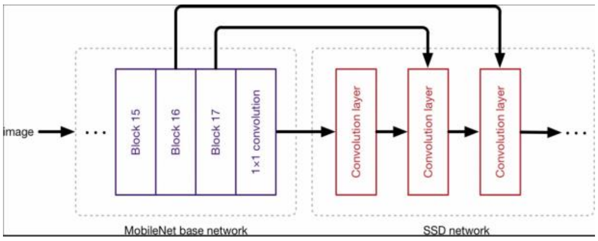


Figure -4: Block Diagram of the MobileNet-SSD Architecture [8]

For the performance analysis of the application presented in this paper, MobileNetV2 integrated with SSD has been used.

3. METHODOLOGY

3.1 Dataset

The data has been manually collected by mass downloading google images of specific classes. A repertoire of images possessing multiple angles of humans with masks were used to ensure the models are being fed all kinds of variations for frictionless operation upon deployment. After gathering all suitable images, they were cropped to provide an aspect ratio of 1:1 to avoid lossy training due to dimensional variations. In the present case, the images have been cropped to a resolution of 416x416 for a fairly quick training process. As one increases this resolution, the training time required for the models also increases.

After pre-processing, the images were labelled and these annotations (containing the locations of the bounding boxes) were saved in encoded formats depending on the requirements of the models. Such uniformity has been maintained in order to guarantee equivalent training of all the models.

Once the images were well annotated, they were split apart as shown in Fig 5. into training, validation and test sets. Applying such a split isn't necessary, but the validation set helps in keeping a check on the erroneous detections.

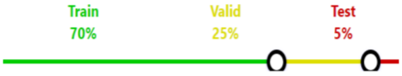


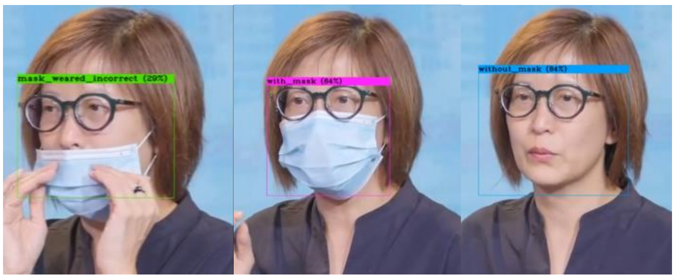
Figure - 5: Training splits of the used Dataset

3.2 Training

The models were trained using a Google Cloud platform instance on Nvidia Tesla T4 GPU having 16GB of RAM. Carrying this out on the platform helped in reducing the training period by quite a bit. Learning rates for all the models was initialized at 0.001, and it was set to depreciate every 1,000 steps. All three models were trained for 900 epochs and since the dataset consisted of almost 1,000 images and the batch size was set to 10, each epoch meant the model underwent 100 steps. Thus, the training continued for almost 90,000 steps. The loss plot for all three stabilized before the end mark, with the YOLO models dipping at about 40,000 steps and SSD stabilizing at 50,000 steps. As mentioned earlier, out of the total dataset, 1,000 images were used for testing and 450 images were set apart for validation and testing.

4. RESULTS & DISCUSSION

A comparison of all three models was carried out after the completion of the training process. The models have been deployed using three devices, namely, Jetson Nano, Nvidia GTX 1660 Ti and Nvidia Tesla T4. This has been done to determine their performance in a high-tier, mid-tier and low-tier processing unit. Now coming over to the factors that are to be taken into consideration for distinguishing the three models, there are two such influential parameters which would determine the model that would be fit for specific use cases. These being the mean Average Precision (mAP) and the processing capability of the model measured via Frames per Second (fps) of the resultant processed video. The output of the trained models is depicted in Fig 6.



(a) **(b)** **(c)**

Figure -6: Outputs of the YOLOv5s Architecture. **(a)** Incorrectly worn mask detected. **(b)** Mask worn correctly detected. **(c)** No mask detected

Often while validating a model's performance, the accuracy is prioritised over the speed and it is assumed that high-calibre GPUs are available in plenty and hence the speed factor will be tolerable. Whereas for real-time deployment, speed is an equally crucial factor and such models are commonly used in gadgets possessing relatively low processing capabilities for routine applications like the present one. For a detection to be considered real-time, the generally acceptable value of fps is 15.

Table -1: Performance analysis of the three models

Model	**mAP (%)**	**FPS**		
		TeslaT4	***1660 Ti***	***Jetson Nano***
YOLOv3	**54.3**	80	21	8
YOLOv5s	37.6	**100**	**28**	**15**
MobileNet-SSD V2	33.7	94	26	**15**

The performance of each of the trained models is provided in Table 1. Keeping the above-mentioned points in mind, it can be inferred that YOLOv5s is the best fit model for real-time situations with optimal values of both accuracy and fps. It can be debated that MobileNet-SSD V2 provides somewhat similar speed to that of YOLOv5s, but it just lacks in the accuracy department. For real-time purposes, speed is a determining factor but accuracy of the model is also essential for fluent functioning. Coming over to YOLOv3, the model caters to excellent accuracy, but it requires computation-intensive hardware. If such a device is available, then this model would suffice the speed requirement. So, it can be said that depending on the requirement of various applications, either of the models can be chosen.

5. CONCLUSIONS

Thus, three models have been compared and each of them demonstrates their own unique characteristics. Each of the models were successful in the required application of mask detection, with the YOLOv5s being the most optimal model for real-time deployment due to its speed and accuracy combination. The other two are also pretty decent models having different use case scenarios. It can be concluded that it is very much possible for computer vision applications to be used in real-time and all these models are suitable to be converted into marketable products.

REFERENCES

[1] Deepa, R., et al. "Comparison of Yolo, SSD, Faster RCNN for Real Time Tennis Ball Tracking for Action Decision Networks." 2019 International Conference on Advances in Computing and Communication Engineering (ICACCE). IEEE, 2019.

[2] Liu, Yifan, et al. "Research on the Use of YOLOv5 Object Detection Algorithm in Mask Wearing Recognition." World Scientific Research Journal 6.11 (2020): 276-284.

[3] Jocher, G., Stoken, A., Borovec, J., Changyu, L., & Hogan, A. (2020). ultralytics/yolov5: v3. 0. Zenodo.

[4] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).

[5] Chiu, Yu-Chen, et al. "Mobilenet-SSDv2: An improved object detection model for embedded systems." 2020 International Conference on System Science and Engineering (ICSSE). IEEE, 2020.

[6] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

[7] R. Deepa, E. Tamilselvan, E. S. Abrar and S. Sampath, "Comparison of Yolo, SSD, Faster RCNN for Real Time Tennis Ball Tracking for Action Decision Networks," 2019 International Conference on Advances in Computing and Communication Engineering (ICACCE), 2019, pp. 1-4, doi: 10.1109/ICACCE46606.2019.9079965.

[8] Hollemans, Matthijs. (2018, April 22). “MobileNet Version2.” https://machinethink.net/blog/mobilenet-v2/