Face Mask Detection

Haoran Jia, Shukai Yin, Xinghao Dong

Department of Mathematics, University of California, Los Angeles

Abstract

Although the COVID-19 pandemic is almost over, some people are still wearing masks to reduce the risk of viral transmission. Detecting whether a person is wearing a mask properly becomes a topic of interest. In this project, we developed three different mask wearing detection algorithms and analyzed their comparative performances. One customized algorithm is built upon OpenCV face detection with VGG19 and ResNet50v2 as backbone feature extractors. Two other algorithms are built on two versions of YOLO – YOLOv5 and YOLOv8. The pros and cons of these models are discussed in-depth, including their use cases, limitations, and real-world dataset displays. As a result, in the classification task, ResNet50v2 shows a best prediction accuracy of 0.94, followed by 0.92 for YOLOv5, YOLOv8 and lastly 0.88 for VGG19, and among the two YOLO versions, YOLOv8 shows better overall performance. However, since OpenCV is trained on normal faces, it has poor performance detecting face with mask compared with YOLO.

Keywords: You only look once (YOLO), VGG19, ResNet50, Image Classification, Face Detection.

Contents

1. Introduction

The COVID-19 pandemic, caused by the SARS-CoV-2 virus, has affected millions of people worldwide, resulting in millions of deaths. The virus is mainly spread through respiratory droplets and contact with contaminated surfaces. To control the spread, the World Health Organization recommends hand washing, social distancing, surface disinfection, and wearing masks. Masks are effective in preventing respiratory droplets from reaching others. N95 masks are 91% effective[6], while surgical masks are 68% effective in blocking virus transmission[2].

The pandemic has led to worldwide scientific cooperation to develop new technologies, such as artificial intelligence (AI), to fight respiratory-transmitted viruses and prevent next outbreak. AI can track the virus's spread, identify high-risk patients, and predict infections early by analyzing patient data, potentially reducing mortality rates[3]. AI can also be used to ensure people are wearing masks in public places. Machine learning and deep learning algorithms can detect face masks in real-time with surveillance camera networks.

While many studies have explored and applied models to detect face masks with high accuracy, there has been little comparison between different algorithms. This study aims to demonstrate the pros and cons of three different object detection algorithms for face mask detection: a customized model built on OpenCV face detection with two distinct image classification backbones, YOLOv5, and YOLOv8. These algorithms were trained and evaluated using a publicly available dataset on Kaggle. The methods and architectures of the algorithms are discussed, and their results are compared based on accuracy.

The paper is organized as follows. In Section 2, related works on object detection are introduced. Sections 3 and 4 present the methods and model architectures used in this study. In Section 5, the Kaggle dataset is briefly described. Section 6 provides an detailed fine tuning process during individual tuning. The model results are showed in Section 7, and their respective pros and cons are evaluated in Section 8. In section 9, potential future works are proposed. Sections 10 and 11 cite the data source and state each members' contributions to the project.

2. Related Work

Object detection is a computer vision task that involves identifying and localizing objects within an image or video. Unlike image classification, which assigns a single label to an entire image, object detection algorithms identify and locate multiple objects within an image and provide a bounding box around each object. Known for high accuracy in image classification tasks, the two convolutional neural network structures, VGG19 and ResNet50[5, 7], have been used as the backbone model in many object detection algorithms including R-CNN, Faster R-CNN, R-FCN and etc[1, 10, 11].

One of the earliest applications of VGG in object detection was in the Region-based Convolutional Neural Network (R-CNN) model proposed by Girshick et al. in 2014[10]. The model used selective search to propose regions of interest (ROIs) in an image, and then applied VGG to each ROI to extract a feature vector. Since then, several variants of the R-CNN model have been proposed, such as Fast R-CNN , which uses VGG as a backbone network with modifications to improve speed and accuracy. Similarly, in the study by Ren et al. (2015)[11], the original ResNet model was also used as the backbone network in the Faster R-CNN algorithm. The authors found that using ResNet as the backbone network led to improved detection accuracy on the PASCAL VOC 2012 dataset compared to previous state-of-the-art methods. In the He et al. (2017)[12], ResNet was used as the backbone network in the Mask R-CNN algorithm. The authors found that using ResNet improved both detection accuracy and instance segmentation performance on the COCO dataset.ResNet50 has also been used in other object detection algorithms, such as the RetinaNet algorithm developed by Lin et al. (2017)[13]. In this algorithm, ResNet50 was used to extract features from input images, which were then used to predict object locations and classes.

YOLO, which stands for "You Only Look Once," is an object detection model that was first introduced in 2016 by Joseph Redmon, Ali Farhadi, and others[4]. The model uses a single convolutional neural network to detect objects in an image and predict their bounding boxes and class probabilities. Since its release, YOLO has become one of the most widely used object detection models in the computer vision community, with several variants and improvements being proposed over the years. For example, YOLOv4, proposed in 2020, used several techniques such as mosaic data augmentation, self-attention, and a new network architecture to achieve state-of-the-art performance on several object detection benchmarks. YOLOv5, proposed in 2020, introduced a new model architecture that was simpler and faster than previous versions of YOLO, while still maintaining high accuracy. Proposed in Janurary, 2023, YOLOv8 reports a new state of the art for object detection. It has numerous architectural and developer experience changes and improvements over YOLOv5, including removing mosaic augmentation, introducing anchor free detection and creating new convolutions.

3. Methods

This section is divided into two parts. The first part outlines the general flow of the study, while the second part focuses on the specific methods that each individual learned and utilized during the study.

3.1. Part I: General Flow

3.1.1. Data Preparation

The dataset used for face detection consists of both image and annotation files, which contain information about bounding box coordinates and their corresponding labels. To prepare the data, the first step is to read the paths for the PNG and XML files separately. We then extract the relevant information from the annotations, such as object label, bounding box coordinates, and image file names, and store them in a pandas dataframe for later use. Additionally, a grammatical error in one of the labels is corrected. Finally, we use this dataframe to draw bounding boxes of different colors and labels for each detected class ("with_mask", "mask_incorrectly_worn", and "no_mask") in the corresponding image.

3.1.2. Customized algorithm

We built a customized algorithm using OpenCV Haar Cascade as face detection model and VGG19, ResNet50v2 as feature extractor to classify whether a face is wearing a mask properly. But unlike YOLO, VGG19 and ResNet50v2 to be fed with cropped image of one human frontal faces beforehand. Once we have our cropped images, we split them into training and validation datasets. We then load the pre-trained VGG19 and ResNet50v2 models and use them as the basis for our own model. Next, we train these two models separately using our prepared dataset. Lastly we use the OpenCV Haar Cascade to detect human frontal faces in real-world images, and predict image class with convolutional neural networks.

3.1.3. YOLOv5 and YOLOv8

Data preprocessing for YOLO is different. The bounding box coordinates need to be transformed from the format of four corner points to the format of center point and box width/height. Moreover, we utilized one-hot encoding to encode class labels numerically because YOLO requires all labels to be numeric. After data preprocessing, the data is split into training and testing sets as usual. Finally, we trained YOLOv5 and YOLOv8 separately using same YAML configuration files.

3.2. Part II: Some Additional Note For Individual Work

After deciding on the face mask detection topic, I discovered plenty of similar projects online, but most of them relies on SSD, faster-RCNN or earlier versions of YOLO Algorithm like YOLO v3 and YOLO v4. With the recent release of YOLOv8 that reported state-of-the-art performance in object detection task, I

decided to utilize the algorithm in this project. Beside the hope to test the ability of the most advanced YOLO algorithm on small-class object detection, one another important reason is the ease of model deployment. Ultralytics API allows for pip installation and simple training and evaluation. However, probably due to its very recent release, the insufficient documentation and tutorial involves difficulty in the actual testing. For instance, the model fails to output configuration using suggested API. Due to this particular reason, I did not try to vary the default setting for the project except writing helper functions for YOLO format data input and prediction. Though it took me a long time to learn about the implementation details and how to customize a model specifically for the task, I believe its extensibility will be beneficial for future work after perfection on the framework as YOLOv8 is designed to support all previous versions of YOLO, with similar API as the existing frameworks, making it easy to switch between different YOLO versions for less experienced users.

As mentioned above, our group proposed two other models using the predecessor of YOLOv8, namely YOLOv5, and a different feature extractor VGG19 for comparison, where ResNet50v2 was not in the original plan. To save computation cost, we deployed models with pretrained weights via keras. But due to the reproductivity of code, I decided to introduce a new structure, ResNet50v2, into the project with ease, as a parallel comparison to VGG19. The reason for the specific choice of ResNet50v2 rather than other neural network structure is that the residual structure is one of the most effective improvements on the convolutional neural networks after the publication of the traditional VGG structure. It may potentially improve on the performance of VGG19 as a backbone feature extractor.

4. Model Architecture

The whole project is built upon three algorithm structures: 1. OpenCV face detection with an image classification backbone convolutional neural network, including VGG19 and ResNet50v2; 2. YOLOv5; 3. YOLOv8.

The following section will provide an overview of the structure of the feature extraction neural networks, including VGG19, ResNet50v2, and Darknet-53, the fundamental building block for both YOLOv5 and YOLOv8.

4.1. VGG19

Developed by the Visual Geometry Group (VGG) at the University of Oxford, the VGG network is a deep convolutional neural network (CNN) originally designed for ImageNet classification but has been widely used for object detection tasks[5]. It is a deep network that consists of 19 layers, including 16 convolutional layers and 3 fully connected layers. The architecture is organized into blocks, with each block containing multiple convolutional layers followed by a max pooling layer, and 3x3 convolution filters are used for all convolutional layers.

4.2. ResNet50v2

ResNet50v2 is a convolutional neural network architecture that was first introduced in 2016 as a successor to the original ResNet50 model[8]. It is a deep learning model that has been trained on massive amounts of image data and has achieved state-of-the-art performance on a variety of computer vision tasks, including object detection, image classification, and semantic segmentation. The ResNet50v2 architecture consists of a deep convolutional neural network with 50 layers, which takes an input image and processes it through a series of convolutional layers, activation functions, pooling layers, and other operations to extract features and classify the image. It is organized into four stages, each containing a varying number of residual blocks. Each residual block is composed of multiple convolutional layers and shortcut connections that allow information to bypass one or more layers, making it easier for the model to learn and reducing the likelihood of vanishing gradients. The number of filters in each block increases as the network goes deeper, allowing it to capture increasingly complex features. In addition to the residual blocks, ResNet50v2 incorporates other

architectural modifications such as bottleneck blocks, which use 1x1 convolutions to reduce the number of parameters in the network.

4.3. Darknet-53

Darknet-53 is a convolutional neural network architecture that is widely used for object detection and image classification tasks[9]. It was introduced in 2018 as part of the YOLOv3 object detection system, which achieved state-of-the-art results on several benchmarks. It consists of 53 layers, including 52 convolutional layers and one global average pooling layer. The network is organized into several residual blocks, with each block containing multiple convolutional layers and shortcut connections. Each convolutional layer in Darknet-53 uses a 3x3 filter and is followed by a batch normalization layer and a leaky ReLU activation function. The batch normalization helps to reduce internal covariate shift and speed up training, while the leaky ReLU introduces some degree of non-linearity into the network. The shortcut connections in Darknet-53 are similar to those in the ResNet architecture and help to mitigate the problem of vanishing gradients. The residual blocks in Darknet-53 also use a technique called "route" concatenation, which allows feature maps from earlier layers to be concatenated with those from later layers.

5. Dataset

The dataset used is a Face Mask Detection dataset currently available for public access on Kaggle, intended for use in developing computer vision models to recognize whether or not individuals in an image are wearing a face mask. It is a collection of 853 images belonging to the 3 classes below with corresponding PASCAL VOC format annotation files:

- With mask
- Mask worn incorrectly
- Without mask

Some well-prepared features about this dataset for the tasks of detection and classifications include:

- Multiple targets with different classes might exist in every single image.
- An XML file for each image in the dataset, which contains information on the objects present in the image, including image filename, image size, object class, and bounding box.

Additionally, the bounding box records the coordinates of a rectangle that encloses the object in the image. For VGG19 and ResNet50, we implemented a function that uses the coordinates information to crop images to smaller images that only contain our targets. Meanwhile for YOLO training, we implemented a method that can convert to a different coordinate system based on center rather than the corner coordinates.

6. Fine Tuning

This section introduces some details for finding the best parameters for the model I worked with.

The first model I worked with is ResNet50v2. ResNet50 has been used widely as the backbone of object detection algorithms, and we chose to use the more advanced version, namely ResNet50v2, as a comparison to other models. In order to lower training expense, we applied transfer learning using pretrained weights on ImageNet, and designed different output layers for classification. The original design is simply a three-neuron output layer with SoftMax activation. The learning rate is set to be 0.00001. The model displays a constant increase in accuracy and drop in loss, with a final accuracy of 0.9597 and 0.9244 on training set and

validation set respectively. This is a clear sign of underfitting, either due to a simple model or low learning rate, so I tried to first overfit the model in the next a few attempts. A learning rate of 0.001 achieves 1.0 accuracy on training set in 10 epochs, but the test accuracy sticks around 0.935. The model is overfitted after the large increase in the learning rate. I firstly tried to apply dropout layers with different dropout rates. They all show slight drop in training accuracy from 1 and increased accuracy on the validation set, but the difference is hard to determine. To test the effectiveness of classifier, I added an additional fully connected layer with 64 neurons before output layer. With a learning rate of 0.001, the classifier generally performs better than before with a slight increase from 0.942 to 0.944. Despite the increased accuracy, one common problem seen in these attempts is that the validation loss keeps increasing. This is a still a sign of overtraining, so I start to consider using a learning rate of 0.0001 and 0.00005. It turns out that while the validation accuracy stays the same, we indeed witness a constant drop in loss, with decreasing speed, but as a trade-off, the accuracy on the training set drops, meaning we the small learning rate influences the model's ability to overfit the training set. The model can be trained for more epochs to get even better performance. From here we can conclude that a learning rate of order at most 0.0001 should be applied, probably because the model is too complicated designed for this three-class classification task that we need to tune it carefully. Also, an ideal approach is to make a use of learning rate decay as the rate of loss drop is decreasing. However, the choice is difficult to make, from my experiment, a fast decay results in local minimum and a slow decay will lead to similar problem as a large learning rate. Therefore, I chose the structure of four-layer classifier with two drop out layers in between. The learning rate and dropout rate are 0.00005 and 0.5. It achieves aounrd 0.96, 0.944 accuracy on training and validation set, with the potential to perform better after training for more epoch. However, since there is a level of randomness in the training, not only the final accuracy may vary, the model performance under different condition may also vary. The best model is hard to decide.

In addition to all the attempts with transfer learning, I also tried model with trainable pretrained layers. Though it achieves a by-far best accuracy of 0.9578 on the validation set, it constantly encounters gradient problems where the loss bouncing between 0.18 to 10000, and a sudden drop in accuracy for every 10 epochs. Due to such instability, I do not consider it as the final model.

The second model I worked with was YOLOV8. But probably due to its very recent release, the current API does not allow direct access to hyperparameters, which makes tuning in notebook extremely difficult, so I did not change it except overwriting the input and target class.

7. Results

We report here our results of the three algorithms used for this face mask detection task:

7.1. OpenCV face detection with VGG19 and ResNet50v2

With 50 epochs, in the classification task, VGG19 and ResNetV2 achieved an accuracy of 0.87 and 0.96 on the training set, 0.88 and 0.94 on the validation set, respectively. The training and testing loss and accuracy of the two models are also shown below in Figure 1 and Figure 2.

From Figure 1, we notice that the curve of testing accuracy stays above the curve of training accuracy, which is an undesirable behavior. We conjectured that this phenomena results from the imbalances of the dataset. As for the accuracy plot of ResNet50v2, we observe similar behaviors that testing accuracy larger than training accuracy. However, after around 30 epochs, the model becomes normal which suggests that ResNet50v2 has better generalization abilities.

7.2. YOLOv5 and YOLOv8

For the two different versions of the YOLO model, we first take a look at their confusion matrices (Figure 3 and 4) after training for 50 epochs, and notice that while the two models did a good job detecting

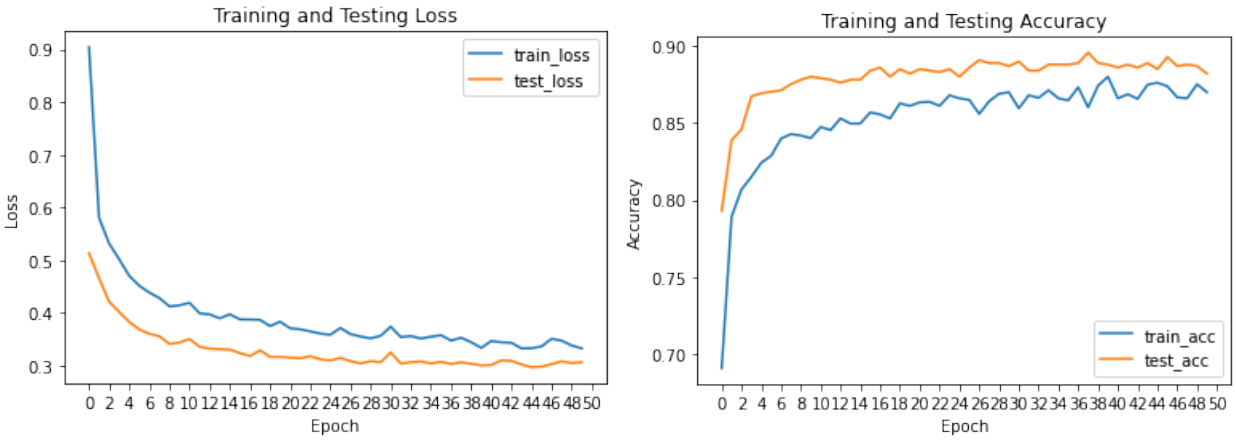


Figure 1: Loss and Accuracy for VGG19

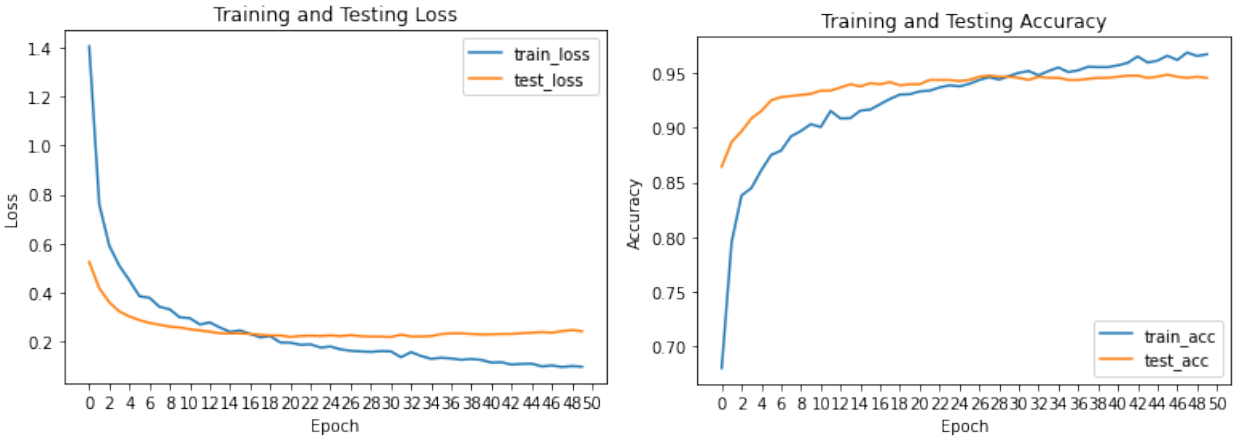


Figure 2: Loss and Accuracy for ResNet50v2

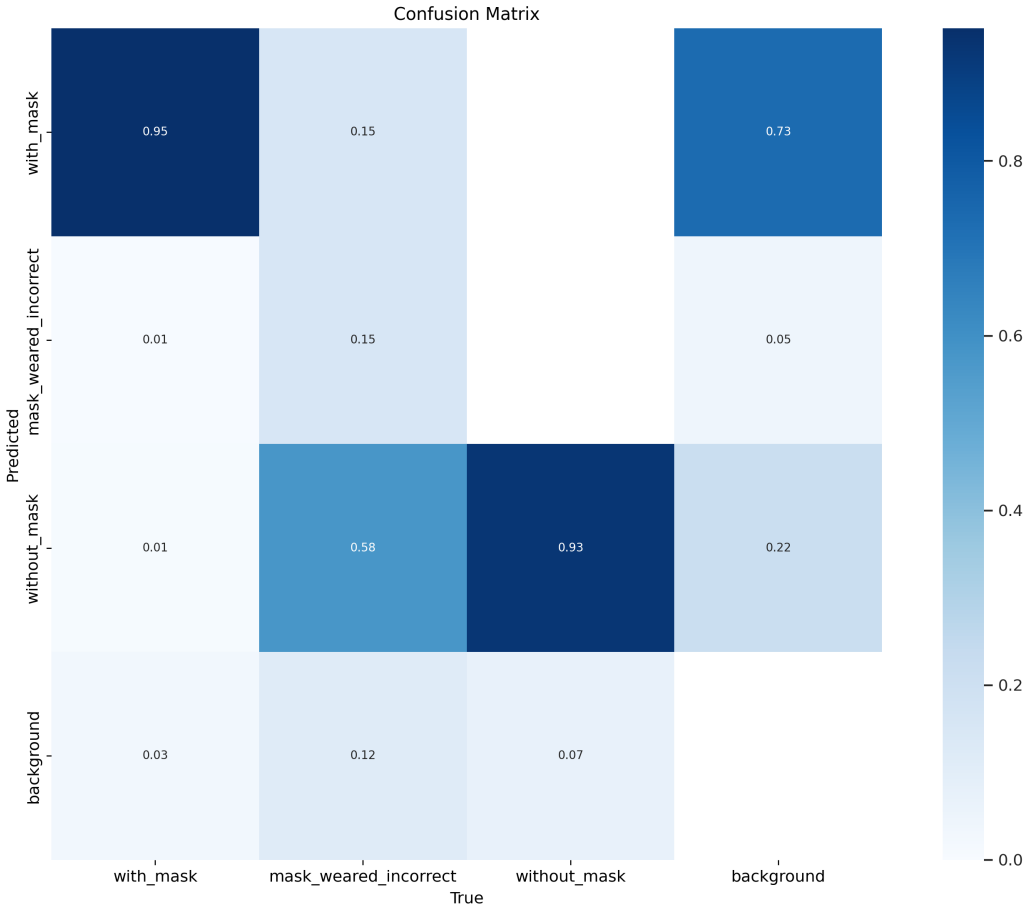


Figure 3: Confusion Matrix for YOLOv5

whether people are wearing a mask or not, but their abilities to evaluate if the mask is worn incorrectly are not very satisfying.

Since YOLO does not automatically report the accuracy of the models, we need to manually calculate the accuracy based on the confusion matrix and the summary of each model. As for YOLOv5, the summary suggests that out of 844 faces in the 171 images of the testing set, 709 faces with masks, 26 faces with masks incorrectly worn, and 109 faces with no masks are used. From the confusion matrix in Figure 3, we can calculate the accuracy of YOLOv5 as follows:

$$\frac{709 \times 0.95 + 26 \times 0.15 + 109 \times 0.93}{844} \approx 0.9227 \tag{1}$$

YOLOv8 uses the exact same training set as YOLOv5, so we can calculate the accuracy of YOLOv8 similarly as follows:

$$\frac{709 \times 0.95 + 26 \times 0.46 + 109 \times 0.86}{844} \approx 0.9233 \tag{2}$$

The accuracy suggests that the two models perform almost equally well on this detection task. However, if we look closer at the confusion matrices, we notice that, although not satisfying, YOLOv8 still does a much better job than YOLOv5 when it comes to detecting faces with masks incorrectly worn. This will be discussed further in later sections.

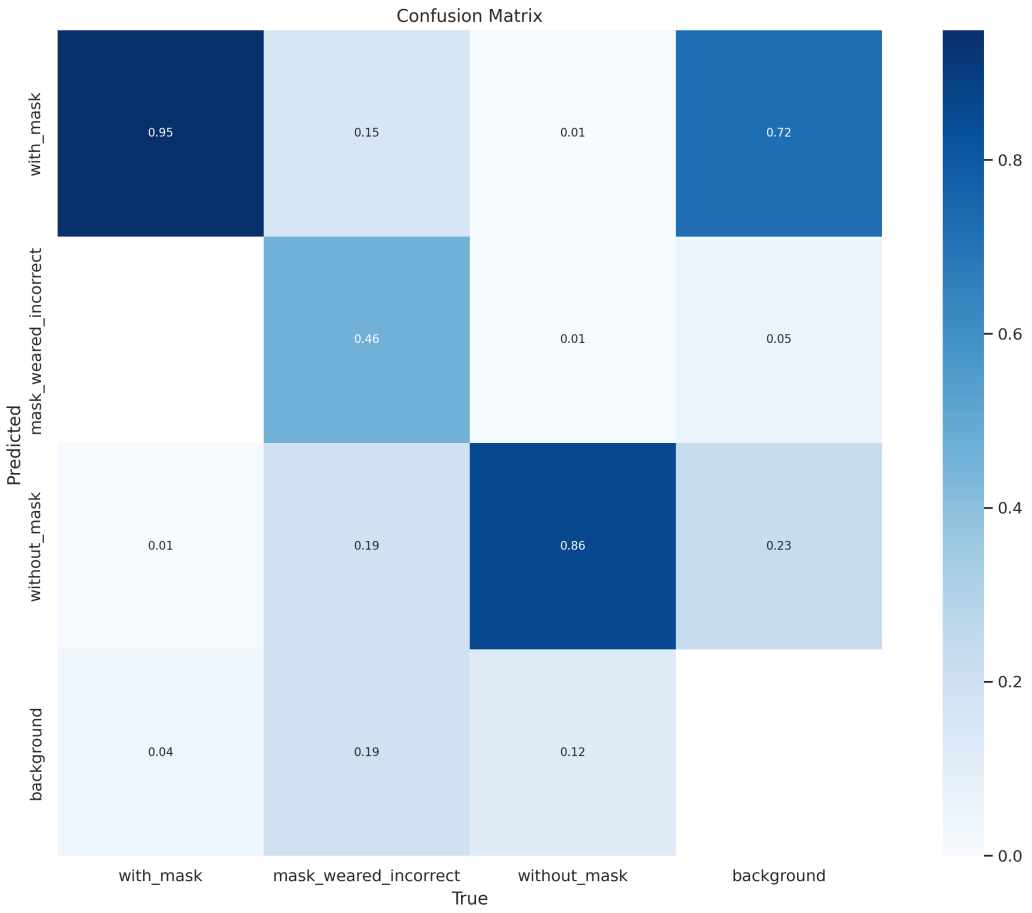


Figure 4: Confusion Matrix for YOLOv8

7.3. Further Analysis with YOLO

In this section I will provide a short comparison between YOLOv5 and YOLOv8 to explain the superior performance of the latter. All the detailed graphs can be found in Appendix B. A closer analysis on confusion matrix(figure 3 and 4) suggests that though the overall accuracy looks almost the same for YOLOv8 and YOLOv5, the prediction accuracy for class with smaller sample size differs significantly. YOLOv8 has 0.46 accuracy on incorrectly weared mask while YOLOv5 has only 0.15.

The following table displays the averaged F1 score and precision over three classes against confidence threshold, which is the minimum level of confidence that is required to classify a predicted object. A more comprehensive graph can be found in figure 14 and 15.

	F1 Score	Confidence
YOLOv5	0.73	0.402
YOLOv8	0.81	0.607
	Precision	
YOLOv5	1	0.942
YOLOv8	1	0.907

The first half shows that YOLOv8 is able to achieve higher F1 score, which means it is performing better in terms of both precision and recall, and is therefore more accurate overall. The second half suggests that

in order to achieve a precision of 1, YOLOv5 and YOLOv8 need a confidence threshold of 0.942 and 0.907, respectively, where the less strict condition suggests slightly better generalization for YOLOv8. Also, the precision of YOLOv5 bounces back and forth between 0.6 and 0.9 during training, while YOLOv8 has a much more stable performance.

Moreover, mAP0.5(mAP50) is commonly used to evaluate the performance of an object detection algorithm. It stands for "mean average precision at an intersection over union (IoU) threshold of 0.5", which measures the average precision of the model at an IoU threshold of 0.5. The IoU threshold means that the predicted bounding box is considered correct if it overlaps with the ground truth bounding box by at least 50 percent. A higher mAP0.5 score indicates better object detection performance. From the graphs of YOLOv5 and YOLOv8(figure 12 and 13), YOLOv8 shows a constant superior mAP0.5 value over YOLOv5 with a final value of around 0.83, while the value is 0.76 for YOLOv5.

8. Evaluation of Algorithms

8.1. Customized Algorithm

- The data preparation process and implementation is more complicated, especially the choice of hyperparameters for feature extractor model.
- One major advantage is that it grants us with large freedom for design. For instance, all structure could be potentially changed including OpenCV face detection model, feature extractor, evaluation metric, and so on.
- A major limitation is their dependence on OpenCV for human face detection. This can lead to reduced accuracy when applied to faces with masks, as the traditional OpenCV is trained using only frontal faces without masks. Additionally, these models require hyperparameter tuning for each image, as the number of human faces in an image can vary. If we do not address this issue, the models may not be suitable for real-world applications since we would need to crop the images first before testing the models.

8.2. YOLOv5

- YOLOv5 is relatively easy to use and can be fine-tuned on custom datasets with just a few lines of code, which makes it an attractive option for small scale projects.
- YOLOv5 can detect face masks in a variety of settings and orientations, making it useful for detecting masks on faces at different angles or orientations.
- YOLOv5 does not have built-in support for advanced data augmentation techniques, which can limit its ability to generalize to different datasets and conditions. YOLOv5 may produce false positives in certain situations, such as when there are objects that are similar in appearance to face masks, or when there are occlusions or other environmental factors that make it difficult to detect masks accurately. This happened when testing the demo and the model recognized a tote bag as a face.

8.3. YOLOv8

- YOLOv8 has quick and easy implementation. It allows for simple data prepossessing without image cropping and it outputs multiple evaluation graphs; but in the mean time, the black-box implementation limits the degree of freedom in the modification of the model, changing the backbone feature extractor, for example, and selection of evaluation metrics.
- A noticeable advantage of YOLOv8 over YOLOv5 is that it deals with imbalanced data more successfully. It has a 31 percent increase in accuracy when predicting for incorrectly wear mask.

- One major problem is that YOLO detection draws bounding boxes for different classes, which frequently results in repeated prediction for different classes, i.e. repeated bounding boxes.

9. Future Works

- One of the major problem with the dataset is that it is largely imbalanced. Among the 853 images, 3232 faces are wearing mask, 123 weared incorrect, and 717 without mask. This means a random guess of face with mask will result in an accuracy of 0.79, and after training, the models indeed display stronger ability to identify face with mask. Therefore, a larger dataset with balanced number of faces between three classes will help the model better learn to classify faces without proper mask wearing.
- For simplicity, accuracy is the sole evaluation metric during the project. However, other metrics like recall rate or precision could be also utilized to evaluate the model for different purposes. Moreover, beside the evaluation of image classification, face detection is also an important component in this project. Metrics like mAP50 could be used to compare the effectiveness of using an OpenCV face detection tool and YOLO algorithm.
- As mentioned above, it occurs sometimes that YOLO made duplicated predictions for different classes. We aim to investigate it further and try to eliminate this drawback in the future.
- To overcome the limitation of OpenCV mentioned above, a new detection model can be trained on human faces with mask. This will ensure the model's ability to detect face with mask in an image.

10. Data Availablity

The dataset used in this report can be found at `https://www.kaggle.com/datasets/andrewmvd/face-mask-detection`.

11. Contributions Statement

Based on his previous project on YOLOv2, Haoran proposed to work on an object detection task. Due to the significant impact of mask policies on our daily lives, we decided to develop a mask detection system. Haoran suggested using different versions of YOLO and a custom algorithm with a backbone CNN network, and conducted an extensive literature review on past object detection algorithms. For the coding part, Haoran was primarily responsible for implementing and training ResNet50v2 and YOLOv8. He also helped to improve the readability of the code, debug other parts of the project, and wrote helper functions to process input data for YOLO and generate predictions. He held regular Zoom meetings to improve communication and coordination between team members. In terms of writing, Haoran mainly contributed to the abstract, related work, method, model architecture, dataset, evaluation, and future work sections of the report. He also reviewed, revised, and provided suggestions for other parts of the report.

Shukai Yin first helped to search for relevant data and open-source codes to support the project. He then reviewed several research papers about COVID-19 and the applications of machine learning models for COVID-19 prevention beyond face masks. For the coding aspect of the project, Shukai Yin developed helper functions to prepare the dataset, display images with bounding boxes, and crop images. He also utilized OpenCV Haar Cascade and trained VGG19 to detect face masks. Additionally, Shukai Yin reviewed ResNet50v2 to determine whether the model required any improvements and created figures to display accuracy and loss for both VGG19 and ResNet50v2. Shukai Yin also contributed significantly to the writing portion of the project, including the Abstract, Introduction, Methods, and part of the Evaluation sections.

He collaborated closely with the other group members to review and revise each other's work. Furthermore, Shukai Yin proofread the report as a whole and provided valuable feedback to identify areas that required further revision or additional information based on the grading scheme.

Xinghao Dong discussed the problem formations with the group members and helped to balance the complexity and feasibility of the project. He also searched for different image datasets on Kaggle and read related paper and reports which finally contribute to the idea of using OpenCV and YOLO to detect face masks. In the coding part, Xinghao is mainly responsible for deploying the YOLOv5 model and developing real time detection demos to check the prediction abilities of the models. He also helped to identify errors in other members' codes including functions that label images incorrectly. He was also responsible for running the codes as a whole, testing new features and collecting results. For writing the report, he contributed to the write-ups of the Abstract, Related Work, Dataset, Results, and Conclusion and Future Works sections of this report; He then collaboratively reviewed and revised the other parts of the report with others.

References

[1] Dai, J., Li, Y., He, K., Sun, J.: R-FCN: object detection via regionbased fully convolutional networks. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 379–387. Curran Associates, Inc. (2016)

[2] N. R. Longrich and S. K. Sheppard, Publicuse of masks to control the coronavirus pandemic, preprints.org, 2020.

[3] R. Vaishya, M. Javaid, I. H. Khan, and A. Haleem, "Artificial intelligence (ai) applications for covid-19 pandemic," Diabetes & Metabolic Syndrome: Clinical Research & Reviews, vol. 14, no. 4, pp. 337–339, 2020.

[4] Redmon, Joseph, et al. "You Only Look Once: Unified, Real-Time Object Detection." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, doi:10.1109/cvpr.2016.91.

[5] Simonyan, K., and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." 3rd International Conference on Learning Representations (ICLR 2015), Computational and Biological Learning Society, 2015, pp. 1–14.

[6] T. Li, Y. Liu, M. Li, X. Qian, and S. Y. Dai, "Mask or no mask for covid-19: a public health and market study," PloS one, vol. 15, no. 8, article e0237691, 2020.

[7] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[8] He, Kaiming et al. "Identity Mappings in Deep Residual Networks." ArXiv abs/1603.05027 (2016): n. pag.

[9] Redmon, Joseph, and Ali Farhadi. YOLOv3: An Incremental Improvement. 2018, http://arxiv.org/abs/1804.02767.

[10] Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587. 2014.

[11] Ren Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems 28 (2015).

[12] He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In Proceedings of the IEEE international conference on computer vision, pp. 2961-2969. 2017.

[13] T.-Y.Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 2999-3007, doi: 10.1109/ICCV.2017.324.

Appendix A: Real Time Detection Illustrations

With the trained model (YOLOv5 with 100 epochs), the real time detection can be carried out via a webcam utilizing the OpenCV library. In the demo, we have demonstrated its ability to detect people wearing masks, not wearing masks, and wearing masks incorrectly. When it comes to different types of face masks, its ability is not affected. A video demo is available online at `https://youtu.be/2HGkQCB9v3w`.

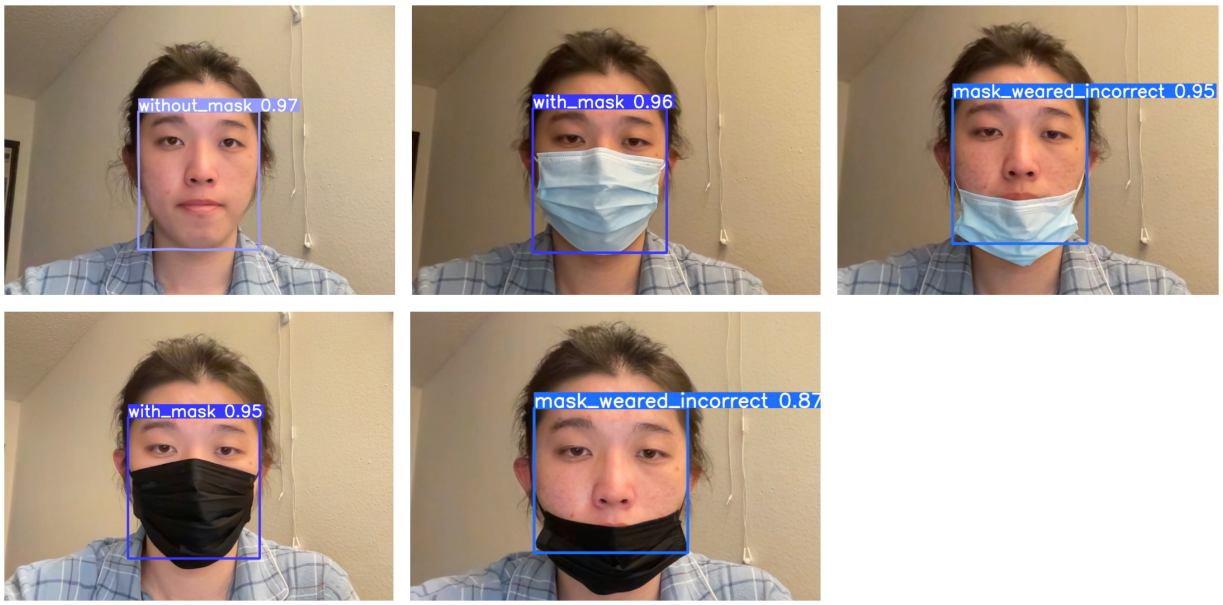


Figure 5: Real Time Detection

Appendix B: Model Outputs

In this appendix, we present the example output from different algorithms. We can see that our models have done a very descent job in predicting classes of with mask and without mask. Looking at the figures below, we can also notice the lack of targets of classes mask worn incorrectly, which could possibly explained by the relatively low accuracy when detecting not correctly worn masks. Toward the end I also report the running output of YOLO.

Figure 6: VGG19

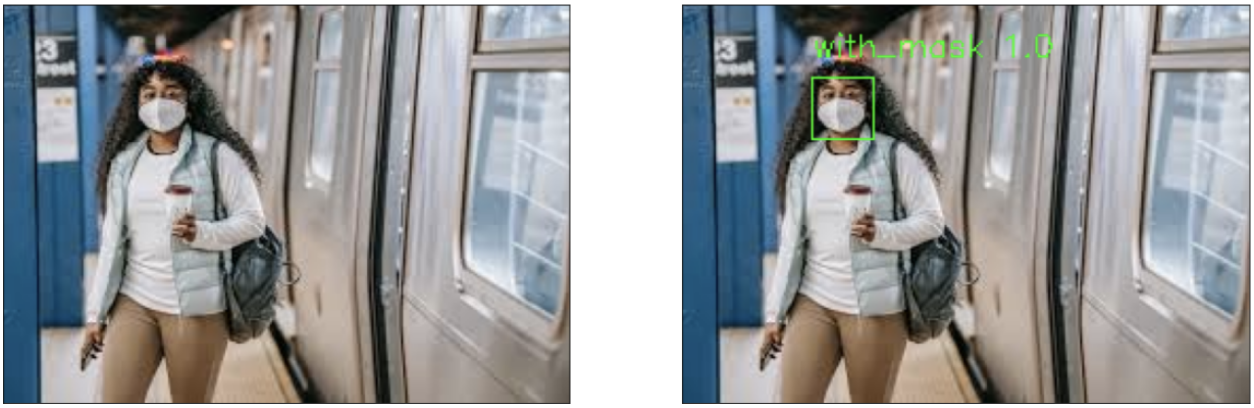


Figure 7: ResNetV2

Figure 8: Batch Labels for YOLOv5

Figure 9: Batch Predictions for YOLOv5

Figure 10: Batch Labels for YOLOv8

Figure 11: Batch Predictions for YOLOv8

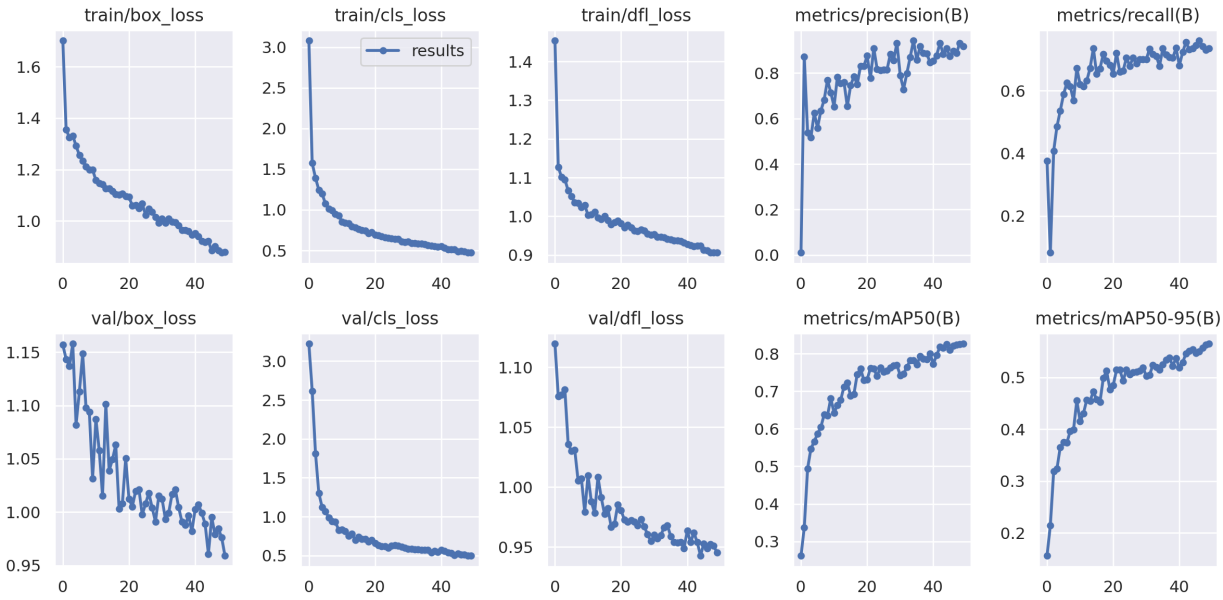


Figure 12: Running output for YOLOv8

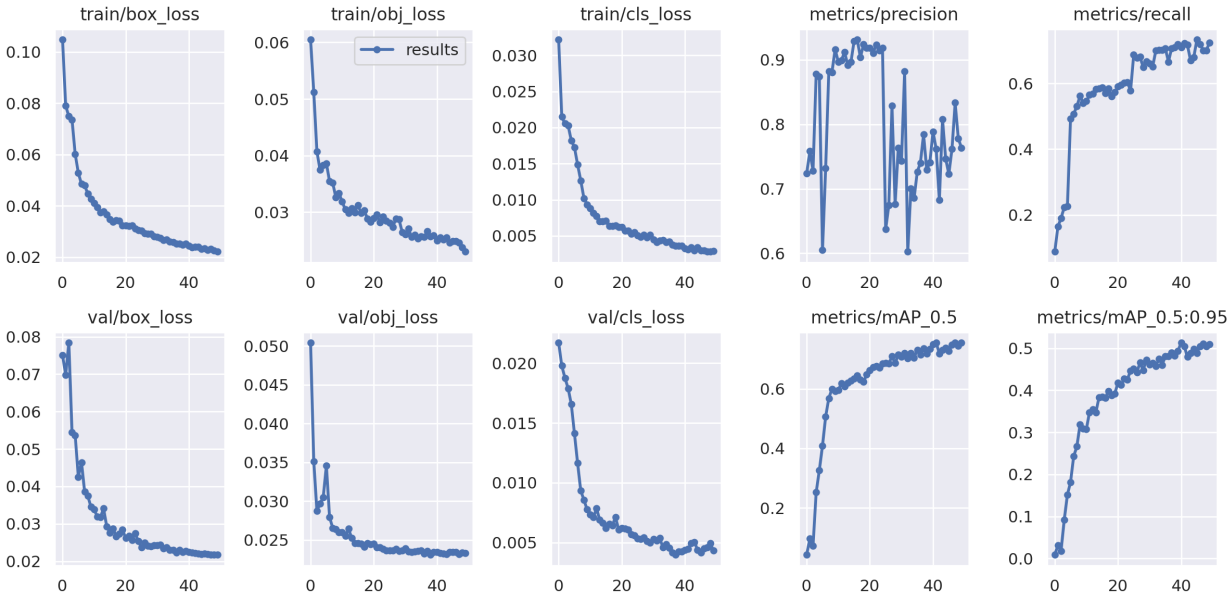
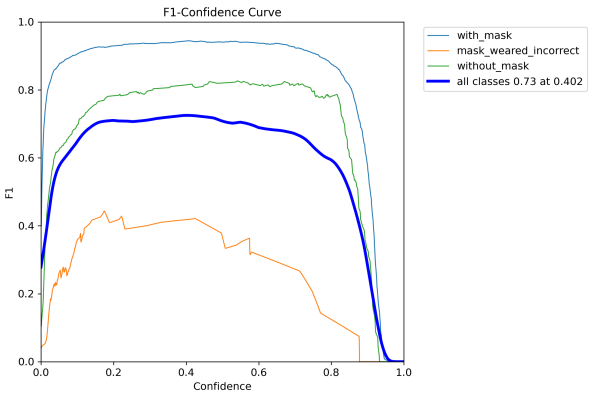
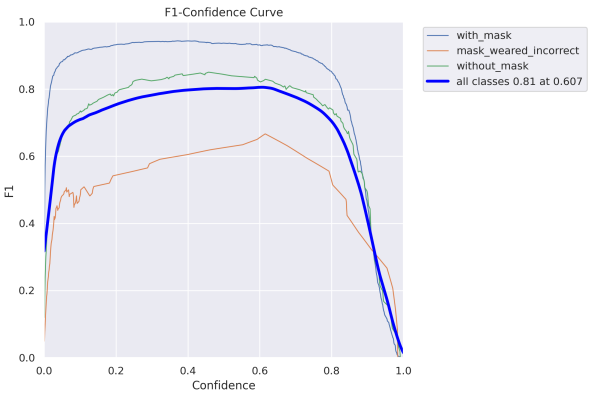


Figure 13: Running output for YOLOv5

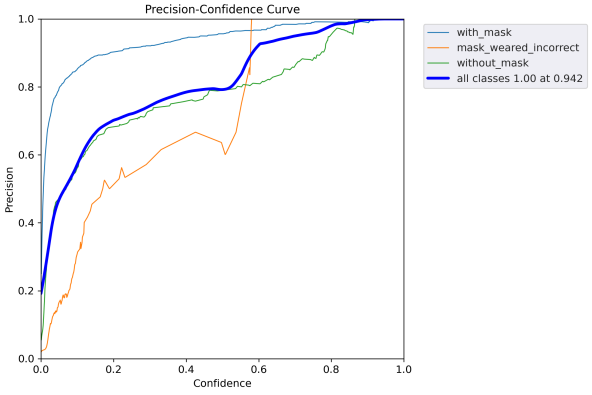


(a) YOLOv5

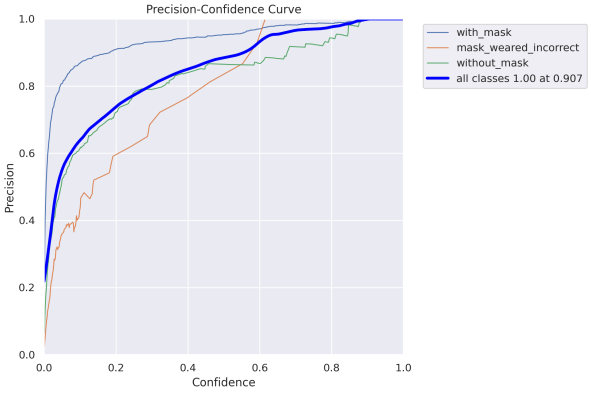


(b) YOLOv8

Figure 14: Comparison of F1 curves against confidence threshold for YOLOv5 and YOLOv8



(a) YOLOv5



(b) YOLOv8

Figure 15: Comparison of Precision against confidence threshold for YOLOv5 and YOLOv8