Northwestern

MSIA 423– Deep Learning

Spring 2022

Final Project Report

Face Localization & Mask Detection

Name	NetID
Lei, Xiuyuan	xlm4182
Xu, Ziru	zxf9764
Yang, Chenxin	cyl9179
Zhao, Ruyang	rzx9163

Abstract

Face masks play a vital role in protecting personal health from respiratory diseases, and they are extremely important to prevent the spread of coronavirus in this age of Covid-19.. Therefore, this project aims to detect and classify mask-wearing in images in order to monitor this behavior. In this report, deep learning algorithms are implemented to first detect human faces and then classify them into 3 categories: with mask, without mask, and mask worn incorrectly. For the localization part, haar feature-based cascade classifiers is our initial attempt. However, noen of the 4 potential models yields satisfactory results, with the best one having 0.47 precision and 0.34 recall. Yolov5 is the best alternative for localization, and the performance on the validation set reaches 0.82 in precision and 0.69 in recall. For the classification part, custom CNN is firstly used and achieved an accuracy of 0.94. Transfer learning model built on Densenet121 is also developed to compare, and it boosts the accuracy to 0.95. Among the 3 classes, the category "incorrectly worn" is significantly more likely to be misclassified, with f1-scores of only 0.5 and 0.57 corresponding to the two models. This is perhaps due to the limited size of training images for this category.

Exploratory Data Analysis

Dataset Overview

This kaggle image dataset contains 4072 objects in 853 images falling into three classes: with mask, without mask, and mask worn incorrectly. It also contains annotation files, each records the name and size of image, the coordinate of bounding boxes of objects in the image, and the class of each object.

Data Exploration

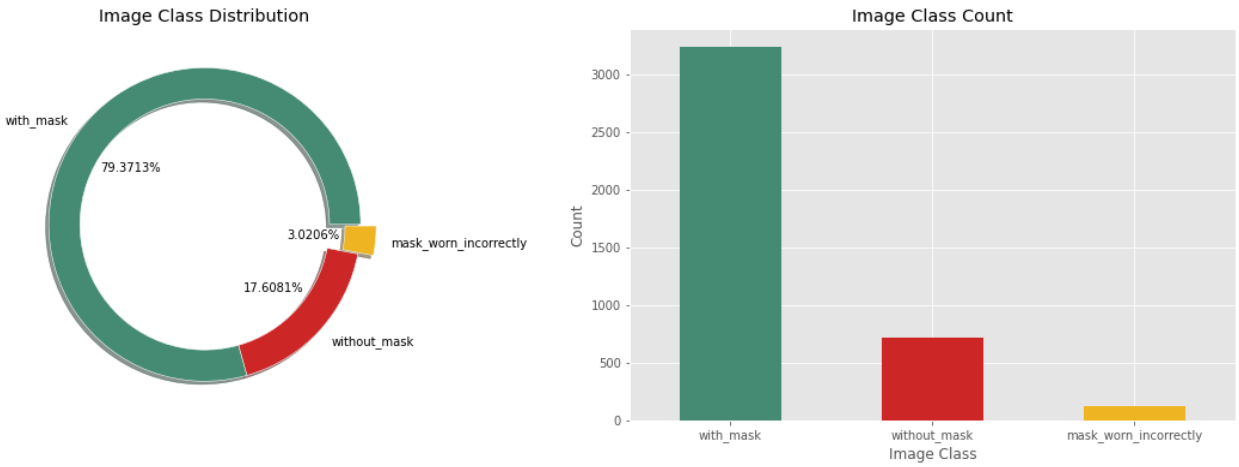

Figure 1. Object Class Count and Distribution

The distribution of image classes is imbalanced, with 79% of objects as with_mask, and only 3% of objects as mask_worn_incorrectly (Figure1). Sample images from each image class are shown in Figure 2.

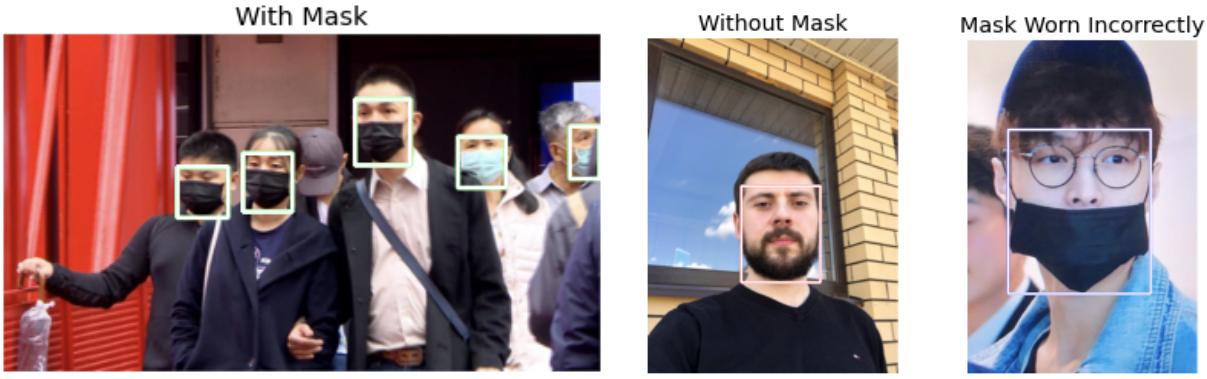

Figure 2. Sample Image from Three Classes

Model Training and Evaluation

Localization Model

Haar

In the beginning, Haar feature-based cascade classifiers are used to detect faces from the images. This method was first proposed by Paul Viola and Michael Jones in their paper in 2001. It is a standard machine learning approach where both positive and negative images are needed to train. During the process, it extracts Haar features from the images, uses Adaboost to select the best features, groups the features into different stages of classifiers and applies them one by one. If a window fails the first stage, it will be discarded. Only the window which passes all stages would be recognized as a face region.

The mechanism above means that the models are pre-trained, and we simply need to fit them and evaluate the performances. Hence, it is easier to implement and achieve our goals. There are four models we need to try individually and choose from. The alt one is a stump-based 20x20 gentle AdaBoost detector; alt2 is tree-based; alt_tree is stump-based and uses a tree of stage classifiers instead of a cascade; default is a stump-based 24x24 discrete AdaBoost detector. They all serve the same purpose of detecting frontal human faces, and thus we decide to implement all of them and compare the performances.

To evaluate localization, the average Intersection over Union(IoU) is calculated as the precision score. IoU indicates the overlap of the predicted bounding box coordinates to the ground truth box. Higher IoU indicates the predicted bounding box coordinates closely resemble the ground truth box coordinates. The percentage of faces that have been detected is counted as recall. The result in Table 1 clearly shows a trade-off between these two metrics.

	Precision	Recall
alt	0.5736	0.2577
alt2	0.5707	0.2847
alt_tree	0.6081	0.0404
default	0.4740	0.3400

Table 1. Precision and Recall of Four Haar Localization Models

The alt_tree model has the highest precision, but it accompanies an unacceptably low recall score. Both alt and alt2 have a lower precision but a much higher recall value. The default model goes along the direction and is the best one among all because a lower precision is usually resulted from drawing a box of a different size instead of missing the objects. Overall, the performance of Haar is not satisfying, and better frontal face detectors need to be explored.

YOLOv5

To improve the localization performance, the YOLO("You Only Look Once") model is used since it is one of the best available models for object detection. It uses a single neural network to process the entire picture, then separates it into parts and predicts bounding boxes and probabilities for each component. It makes predictions after only one forward propagation runs through the network. For this localization problem, YOLOv5, the 5th version of the YOLO model, which was launched by Ultralytics in June 2020, fits our needs well because it allows users to train the Deep Neural Network on a custom dataset.

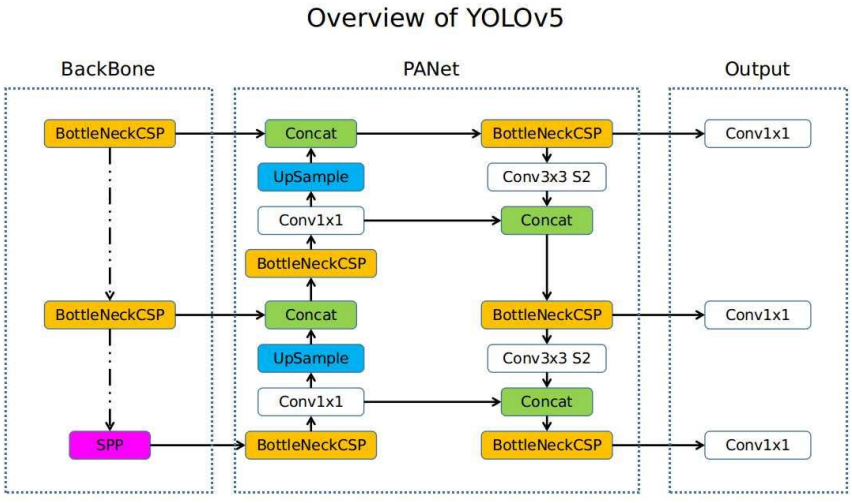

Figure 3. Overview of YOLOv5 Model Architecture

The training environment is set up by cloning the YOLOv5 repository from GitHub. YOLOv5 requires a very specific setup of data folders to work. A data folder is created at the same level as the yolov5 folder and split into images and labels under train, validation, and test. The annotation files in the training data are transformed from pascal VOC XML format to YOLO Darknet txt format that includes the class number of the object in the bounding box, the standardized center pixel of the bounding box, and the standardized width and height of the bounding box. Two YAML files are created to start training the YOLO model, one specifying the path to training data and validation data, the number of classes, and the names corresponding to classes, the other specifying the model architecture. Ultralytics supports several YOLOv5 architectures, named P5 models, which vary mainly by their parameters size: YOLOv5n (nano), YOLOv5s (small), YOLOv5m (medium), YOLOv5l (large), YOLOv5x (extra large). These architecture are suitable for training with images of size 640*640 pixels. A higher series called P6, which is optimized for training with a larger image size of 1280*1280, includes an extra output layer to detect larger objects. They benefit the most from training at higher resolution and produce better results. P5 models are our choice since our dataset is relatively small.

The YOLO model is trained on the training set with specified model hyperparameters such as image size, batch size and the number of epochs, and precision and recall are used to evaluate the model performance. Precision measures how much of the bounding box predictions are correct, and recall measures how much of the true bounding box was correctly predicted. From Table 2, it is clear that all YOLOv5 models have much higher precision and recall scores compared to Harr models. In our case, recall is slightly more important than precision since the goal is to localize all true bounding boxes of faces in the image. Therefore, among all YOLOv5 models, YOLOv5m has the best model performance, given the limited GPU resources.

	Precision	Recall
YOLOv5n	0.9244	0.6428
YOLOv5s	0.7505	0.6387
YOLOv5m	0.7371	0.7737
YOLOv5l	0.9336	0.7442

Table 2. Precision and Recall of YOLOv5 Models

Other model metrics, including mAP_0.5 and mAP_0.5:0.95, are also used. mAP_0.5 is the mean average precision (mAP) at the IoU threshold of 0.5, and mAP_0.5:0.95 is the average mAP over different IoU thresholds, ranging from 0.5 to 0.95. In Figure 4, both mAP metrics of YOLOv5m keep increasing over epochs. At the end of 50 epochs, mAP_0.5 reaches 0.7889, mAP_0.5:0.95 reaches 0.5529, and both have continuous growth trends. Finally, the three-loss metrics YOLO uses are examined: box_loss, obj_loss, and cls_loss. Box_loss is the bounding

box regress loss (mean squared error), obj_loss is the confidence of object presence(binary cross-entropy), and cls_loss is the classification loss (cross-entropy). All three losses decrease over time and approach zero. These metrics show that YOLOv5 models have excellent performance in locating faces in the data.

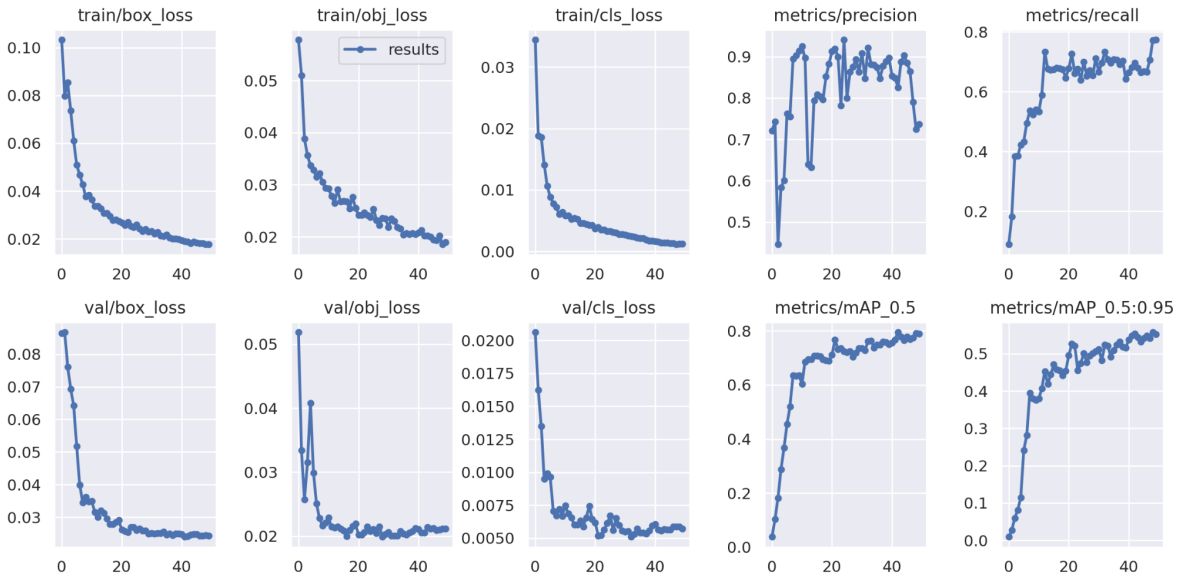

Figure 4. YOLOv5m Model Evaluation Metrics

In addition, while localizing human faces, YOLOv5 models also classify faces into different categories. It can achieve the task so-called "classification with localization," which is a much more challenging task than classification alone. Figures 4, 7, 10, and 13 in the Appendix show each YOLO model's confusion matrix of classifications, and YOLOv5l has the best classification performance. While YOLO models perform well on faces with_mask and without_mask, it performs relatively poorly on objects wearing a mask incorrectly. For instance, the YOLOv5s model either fails to localize the face, as shown in Figure 5 or classifies the object as wearing a mask correctly, as shown in Figure 6.

Figure 5. YOLOv5s Mislocated Example

Figure 6. YOLOv5s Misclassified Example

Classification Model

Customized CNN

Our main task for classification is to categorize people in the images into three categories: with mask, without a mask, and mask worn correctly. Traditional Neural Networks often struggle to prevent overfitting for images with high resolution because the number of parameters in the network could grow very large. As a result, CNN (Convolutional Neural Network) is proposed as the model structure for this multilabel classification task. CNN implements convolutional operations that could learn both low-level features (at initial layers) and high-level features (at deeper layers) with a low risk of overfitting. CNN has the following advantages compared to traditional fully connected neural networks:

1. The relatively small number of parameters: it can prevent the model from overfitting as well as accelerate the training process given the limited computing resources.
2. Parameter sharing: a feature detector (such as a vertical edge detector) that can extract features in one part of the image could probably perform the same task in another part of the image. Thus, sharing the parameters by applying a filter is a very useful design.
3. The sparsity of connection: in each layer, each output value depends only on a small number of inputs.

Our customized CNN model starts with an instance of sequential() and two convolutional layers with maxpooling to downscale the images. Then, the model uses a flattening layer to vectorize the output, followed by dense layers with ReLU activation functions to transition smoothly in the intermediate layers. Dropout layers are also used to prevent overfitting. The output layer uses the

softmax activation function since it can output the probability of each class. The structure of the model is summarized in Figure 7. The model uses Adam as the optimizer to reach convergence faster and reduce computation time. For other hyperparameters, the learning rate is set to be 1e-4 with a decay of 1e-5. The low learning rate in this scenario ensures the training loss and validation error could decrease steadily.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 47, 47, 100)       2800

max_pooling2d_8 (MaxPooling  (None, 23, 23, 100)      0
2D)

conv2d_9 (Conv2D)            (None, 21, 21, 64)        57664

max_pooling2d_9 (MaxPooling  (None, 10, 10, 64)       0
2D)

flatten_4 (Flatten)          (None, 6400)              0

dense_8 (Dense)              (None, 50)                320050

dropout_4 (Dropout)          (None, 50)                0

dense_9 (Dense)              (None, 3)                 153

=================================================================
Total params: 380,667
Trainable params: 380,667
Non-trainable params: 0
```

Figure 7. Customized CNN Model Structure

The model achieves 94% accuracy on the testing dataset. The classification report and confusion matrix are provided below in Figures 8a and 8b.

	precision	recall	f1-score	support
incorrecly wear mask	0.71	0.38	0.50	13
corrrectly wear mask	0.95	0.98	0.97	324
no mask	0.93	0.89	0.91	73
micro avg	0.95	0.95	0.95	410
macro avg	0.87	0.75	0.79	410
weighted avg	0.94	0.95	0.94	410
samples avg	0.95	0.95	0.95	410

Figure 8a. Customized CNN Classification Report

Figure 8b. Customized CNN Model Classification Confusion Matrix

From the classification report, we could see the model works well on predicting people wearing masks correctly and people without masks but works relatively poorly on predicting people wearing masks incorrectly. This poor performance on people wearing masks incorrectly could be due to the lack of enough training data for this category, and varying ways of incorrectly wearing masks makes this category inherently harder to identify.

Looking at misclassified images in Figure 9, we found there are three types of images that are easy to be misclassified: blurry images, images that do not show people's entire faces, and images with mask color close to the skin color.

Examples of misclassified images are shown below in Figure 9.

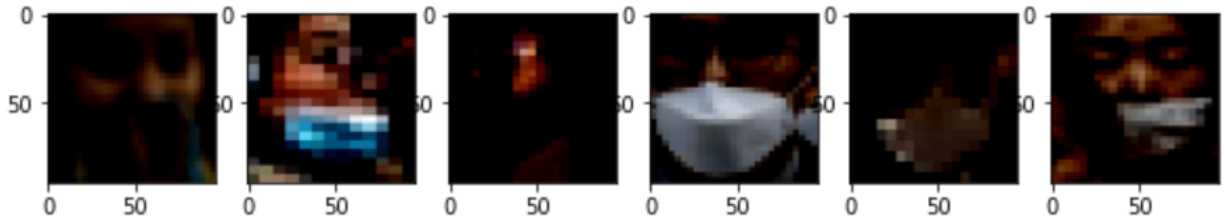

Figure 9. Customized CNN Model Misclassified Images

Transfer Learning

To build upon previous work in this field, we researched many existing pre-trained networks. Densenet turns out to be a good choice because it has the following advantages compared to other pre-trained networks:

1. Good performance compared to other architectures in Imagenet. We tried some popular CNN architectures like VGG, and densenet achieved the best results among all of them.

The output layer of densenet is connected to all prior layers, thus, using features of all complexity levels. As a result, overfitting is less likely to happen when the training sample is not very large. This is a very important advantage since we only have several thousand pictures to train the model, and the pretrained model with complex structures could potentially overfit the data.

2. Improved parameter efficiency makes densenet easy to train. With a similar number of layers, densenet usually has fewer parameters. Consequently, training densenet takes less time, and given the fact that we only have very limited computing resources, this could be a very helpful feature.

The full model starts with two preprocessing lambda layers. The first layer is a resizing layer that resizes the image into 224 by 224 since this is the size of the images on which the densenet121 was trained. The second layer is a preprocess layer that scales and normalizes the input data with respect to the ImageNet dataset.

Two random augmentation layers are added to perform image augmentation. Specifically, a random rotation and random flip layer are used, and these layers will generate real-time random transformation to the input image, which serves a similar function as the ImageDataGenerator. In addition, they are only active during training. It is also worth mentioning that ImageDataGenerator is deprecated now, and performing data augmentation using preprocessing layers is recommended by the office guide of Keras.

Densenet is flattened at the end and followed by several dense layers with dropouts and batch normalization, the purpose of which is to regularize the last few layers, making sure the model could generalize well beyond the training data.

The model achieved 95% overall accuracy, and the confusion matrix is provided below in Figure 10.

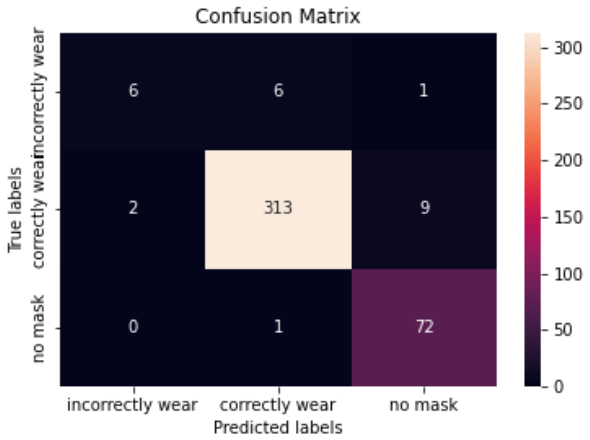

Figure 10. Transfer Learning Model Classification Confusion Matrix

Important metrics of the classification results on the test set are summarized below. Overall, the model works really well on people with masks, having a nearly perfect recall and precision. The model also works reasonably well on people without masks, able to identify almost all the people without masks with a precision about 88%. Identifying people who wear masks incorrectly is a much harder task for the model, the 75% precision is reasonable, but a recall of 46% definitely needs improvements.

	precision	recall	f1-score	support
mask worn incorrectly	0.75	0.46	0.57	13
with mask	0.98	0.97	0.97	324
without mask	0.88	0.99	0.93	73
accuracy			0.95	410
macro avg	0.87	0.80	0.82	410
weighted avg	0.95	0.95	0.95	410

Figure 11. Transfer Learning Classification Report

The poor performance on people who wear masks incorrectly is expected since the training data contain much fewer pictures of people wearing masks incorrectly. Moreover, people with masks or without masks are very distinctive and homogeneous among their own group, whereas there are so many ways to wear the masks improperly.

After examining images that are misclassified, three types of images are found to be more easily misclassified: very blurry ones, images with mask color very close to the skin tone and side pictures, of which the examples are shown below in Figure 12.

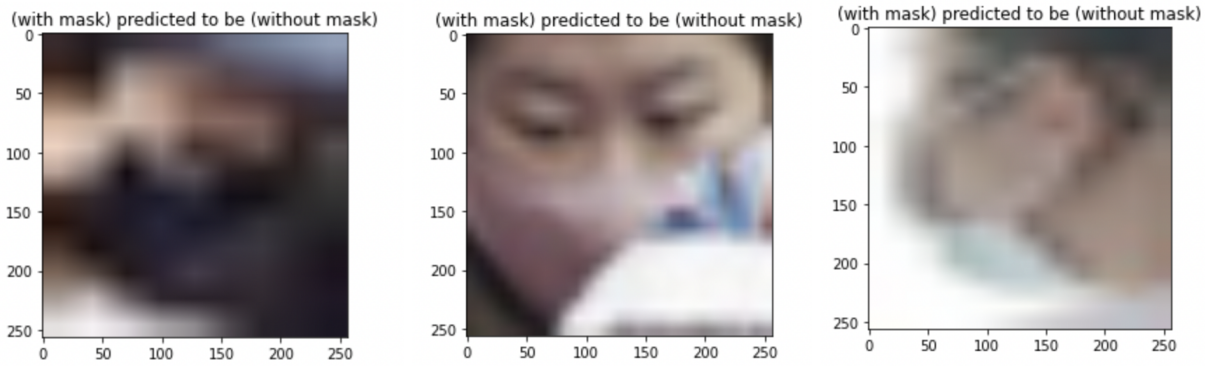

Figure 12. Transfer Learning Model Misclassified Images

Model Operations

Face mask classification has many use cases, especially for now when the pandemic of covid-19 is still plaguing the world. Despite the fact the United States has lifted most of the pandemic-related restrictions, masks are still required in many parts of the world. This model could be of great use, where the mask mandate is in place. For example, if a shopping mall with surveillance cameras wants to enforce the mandate, pictures could be taken regularly and run through our detection and classification algorithm. If people without masks or wearing masks incorrectly are detected, warnings will be generated to notify the security team.

The model will be an independent service running in the cloud and provide a well-defined API for external systems to use it. When using the model, HTTP requests with images could be sent through the API, and the response will be images with bounding boxes and prediction labels. This architecture would insulate the model from the external systems, so it could be maintained and iterated on separately without affecting other services.

To monitor and improve the model in the production environment, we re-train the model by feeding in newly captured images with annotation and labels periodically to update the model parameters, so the model could be up to date. The re-training and update will happen offline when the service is inactive, such as during the time when the malls that deploy it are closed.

Conclusion

For this project, four models have been developed, two for each task (detecting human faces and classifying how they wear masks). Yolo is much more efficient in detecting human faces compared to Haar feature-based cascade classifiers. The best Harr model is the default one with a precision of 0.4740 and recall of 0.3400. The best YOLO model is the YOLOv5m with a precision of 0.7371 and recall of 0.7737.

	Precision	Recall
Harr default	0.4740	0.3400
YOLOv5m	0.7371	0.7737

Table 3. Comparison Between Best Harr and YOLO v5 Models

For classification tasks, the two models have similar performance, with the transfer learning model slightly outperforming the customized CNN model. Both the customized model and the transfer learning model built upon Densenet perform very well in classifying people wearing

masks correctly and without masks (more than 0.9 f1-score) but perform not as well at classifying people wearing masks incorrectly (around 0.5 f1-score).

	Number of Parameters	Accuracy
Customized CNN	380,667	0.94
Transfer Learning	20,117,827	0.95

Table 4. Comparison Between Customized CNN and Transfer Learning Models

Combining YOLOv5m's localization with the transfer learning model's classification could yield a powerful model pipeline that recognizes faces in any given image and determines the mask status.

Limitations & Future Improvements

For face localization, YOLOv5 has very good performance, identifying most of the objects in the image, but it will draw multiple bounding boxes for the same object when a small number of epochs is used(less than 5) as shown in Figure 13. This issue could partially be alleviated by setting a higher threshold for the confidence level, and the threshold value could be further fine-tuned so that most of the faces are recognized with minimal redundant bounding boxes. With a large number of epochs, the problem of redundant bounding boxes is resolved, but the computing resources needed to train YOLO are not trivial. Notably, even though YOLOv5 trains and inferences very fast, it is picky on the environment. GPU is required to train the YOLOv5 model, which could pose problems when GPU is hard to access. For the YOLOv5l model, even the paid Google Colab Pro shows running out of memory error when training it on a batch size of 32, and all YOLOv5 models failed on the Northwestern deepdish servers. For the next step, it is also worth trying to see whether YOLOv5x and P6 models can further improve YOLO performance in both localization and classification if computing power is allowed.

Figure 13. Redundant Bounding Boxes Examples

The limitations of our classification model come from the images with people wearing masks incorrectly because the task is innately harder, and the number of images in this category is limited. To improve the model performance in this category, more images in this category need to be added to the training set. In addition, lower learning rates with more epochs could also be tried if we have access to more computing resources.

References

https://www.kaggle.com/datasets/andrewmvd/face-mask-detection
https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
https://github.com/ultralytics/yolov5
https://towardsdatascience.com/the-practical-guide-for-object-detection-with-yolov5-algorithm-74c04aac4843
https://arxiv.org/abs/1608.06993

Appendix

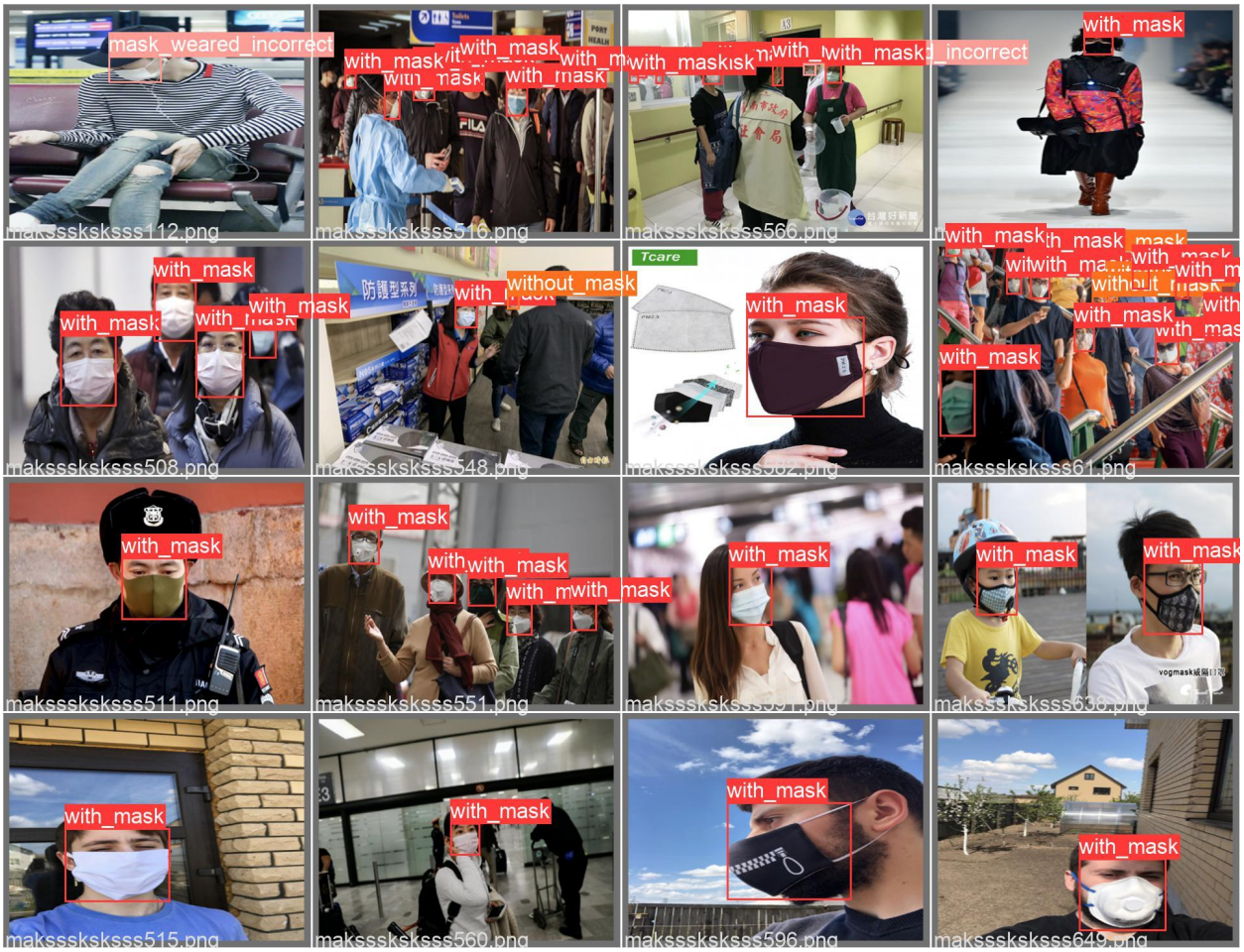

Figure 1. Validation Ground Truth Label

YOLOv5n(batch size =32)

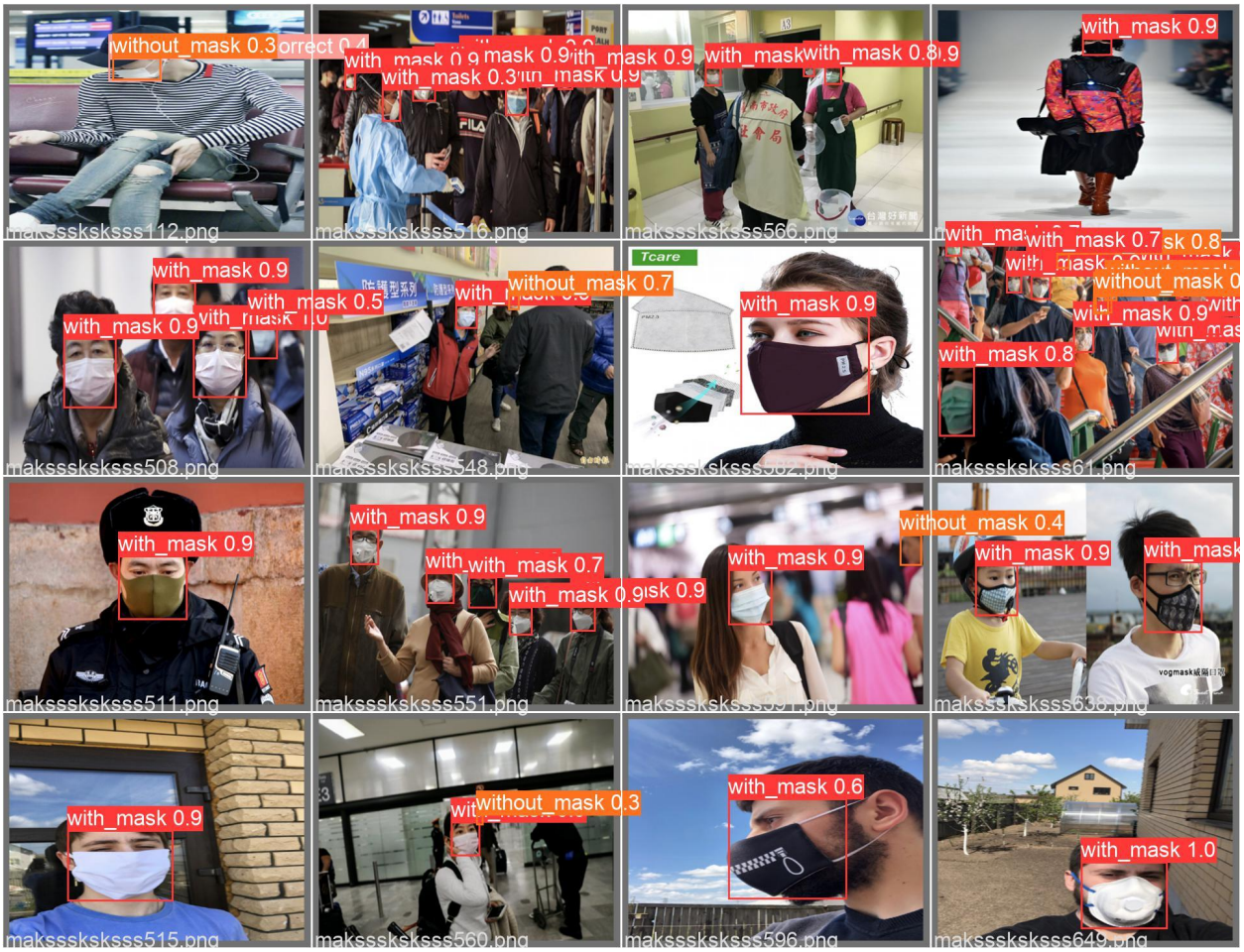

Figure 2. YOLOv5n Prediction Output

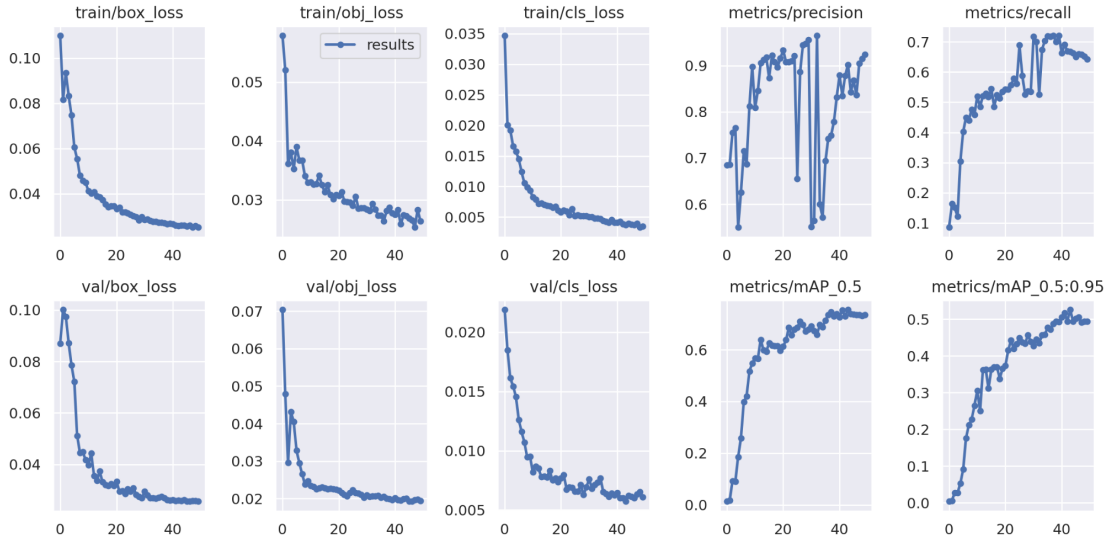

Figure 3. YOLOv5n Model Evaluation Merics

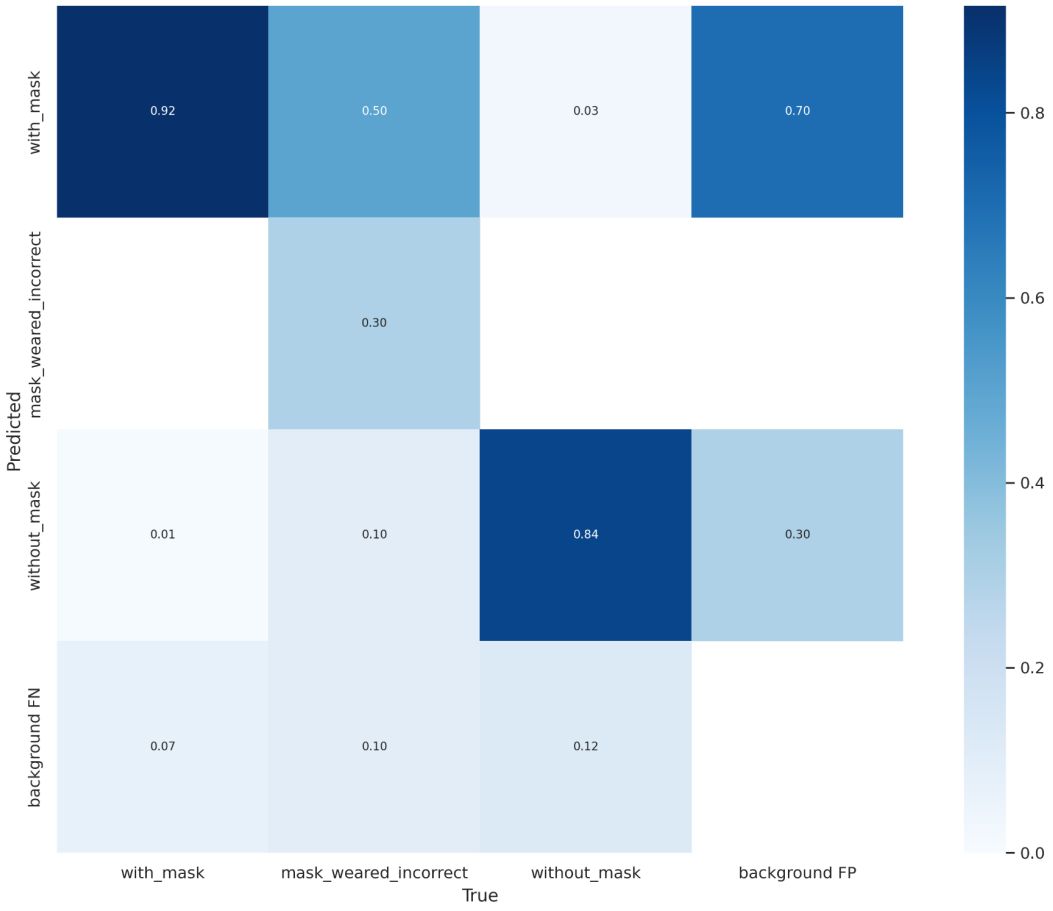

Figure 4. YOLOv5n Model Confusion Matrix

YOLOv5s(batch size =32)

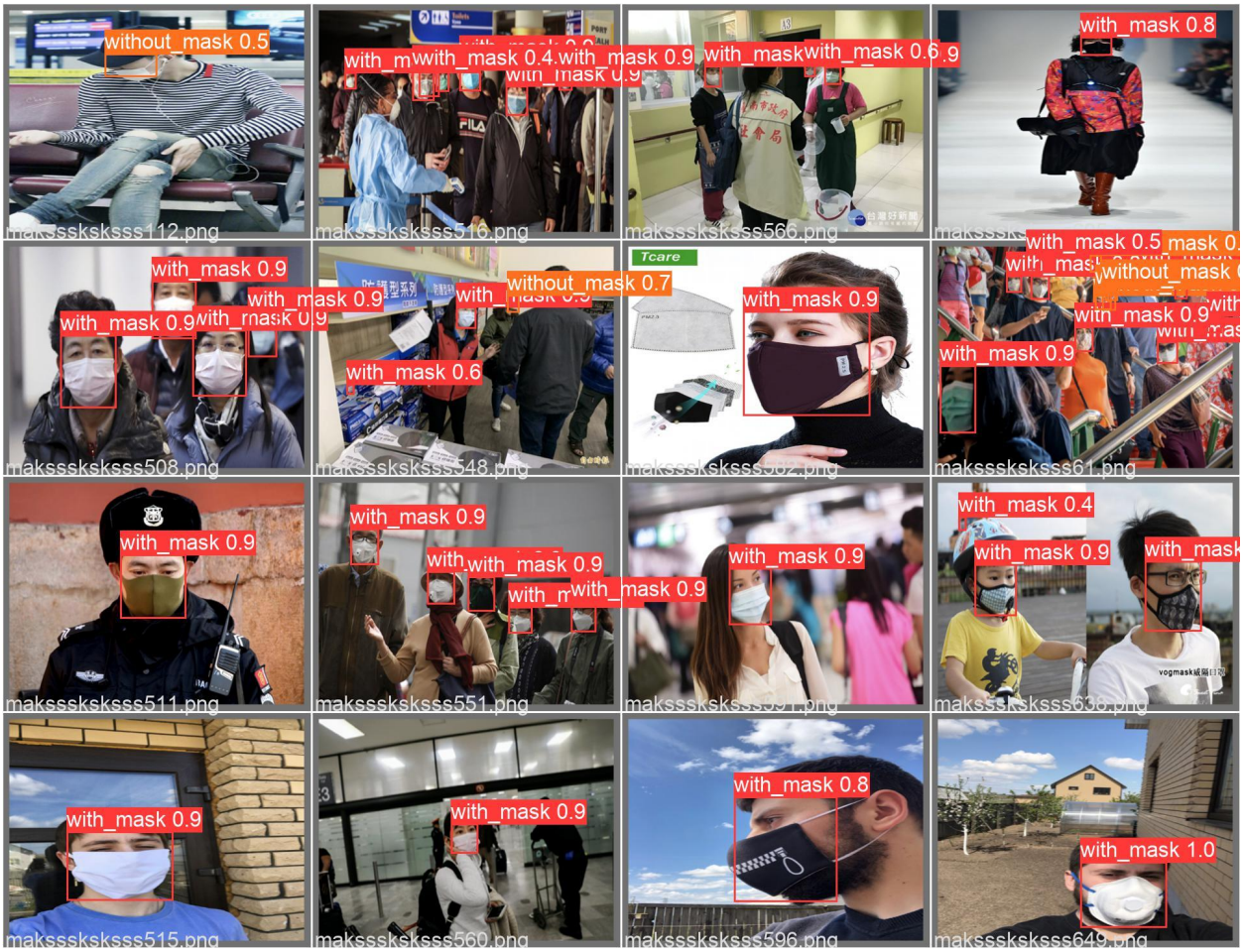

Figure 5. YOLOv5s Prediction Output

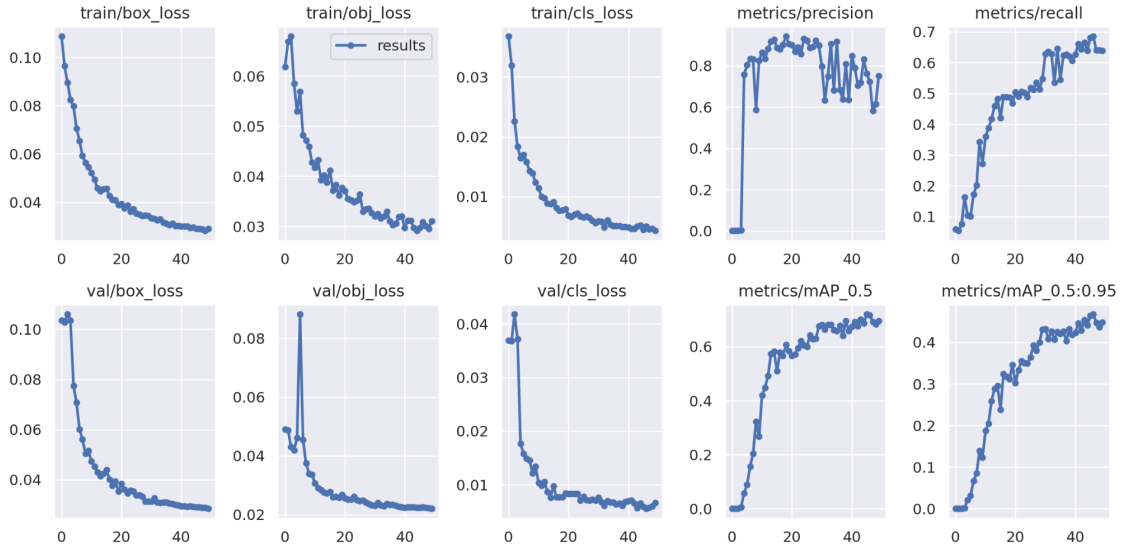

Figure 6. YOLOv5s Model Confusion Matrix

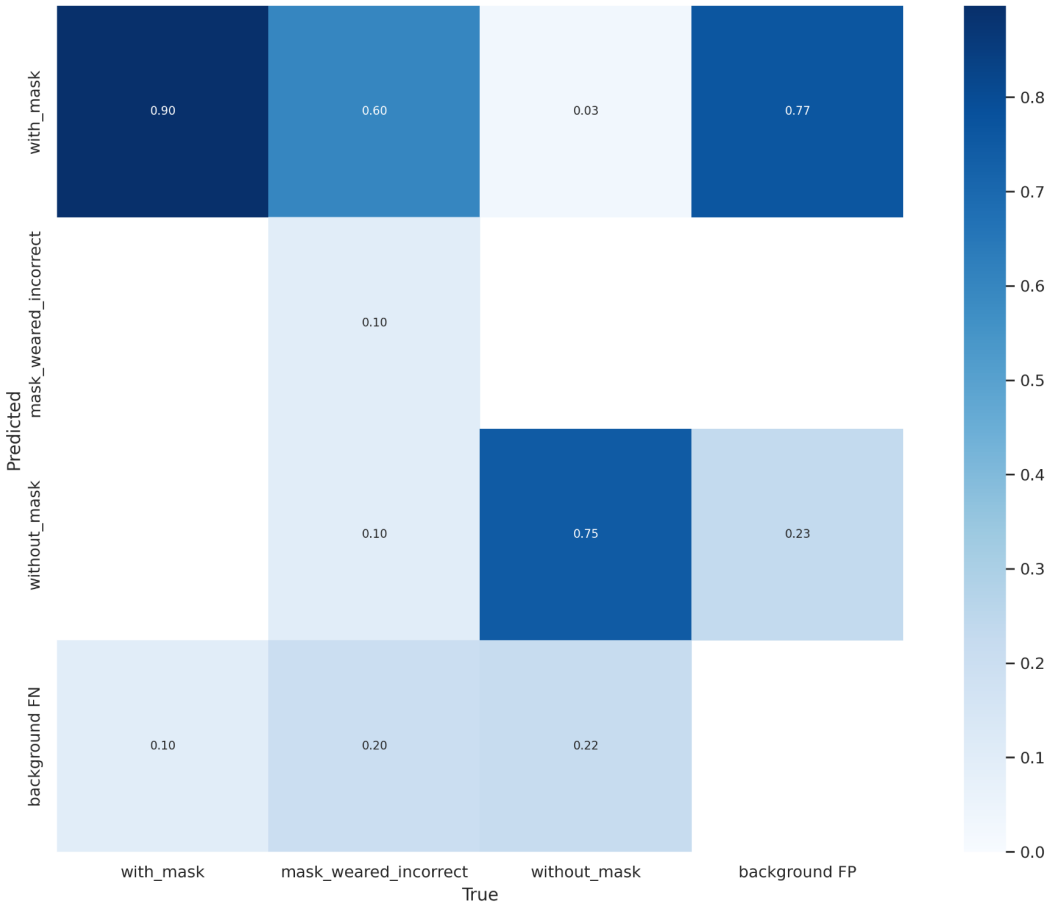

Figure 7. YOLOv5s Model Confusion Matrix

YOLOv5m(batch size =32)

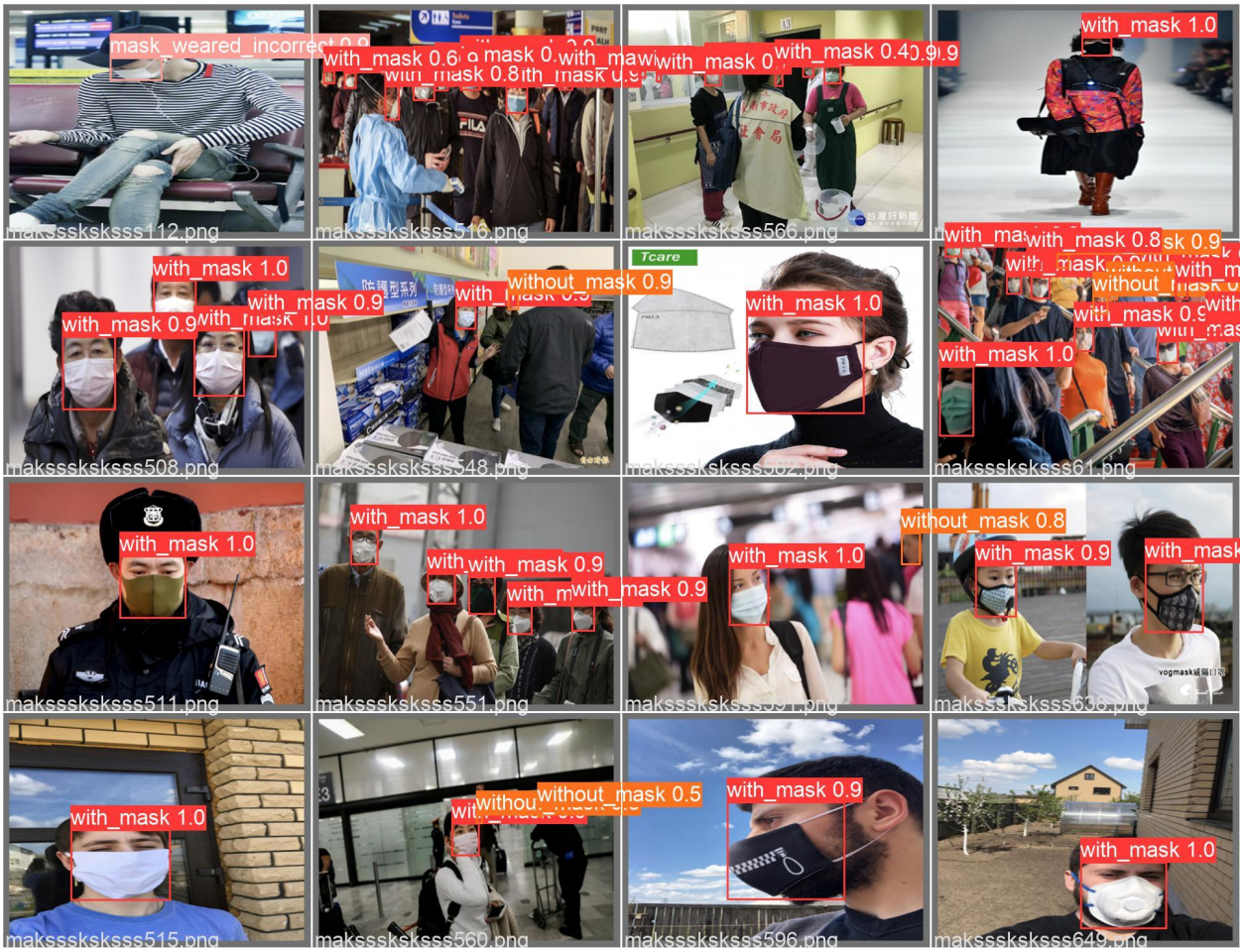

Figure 8. YOLOv5m Prediction Output

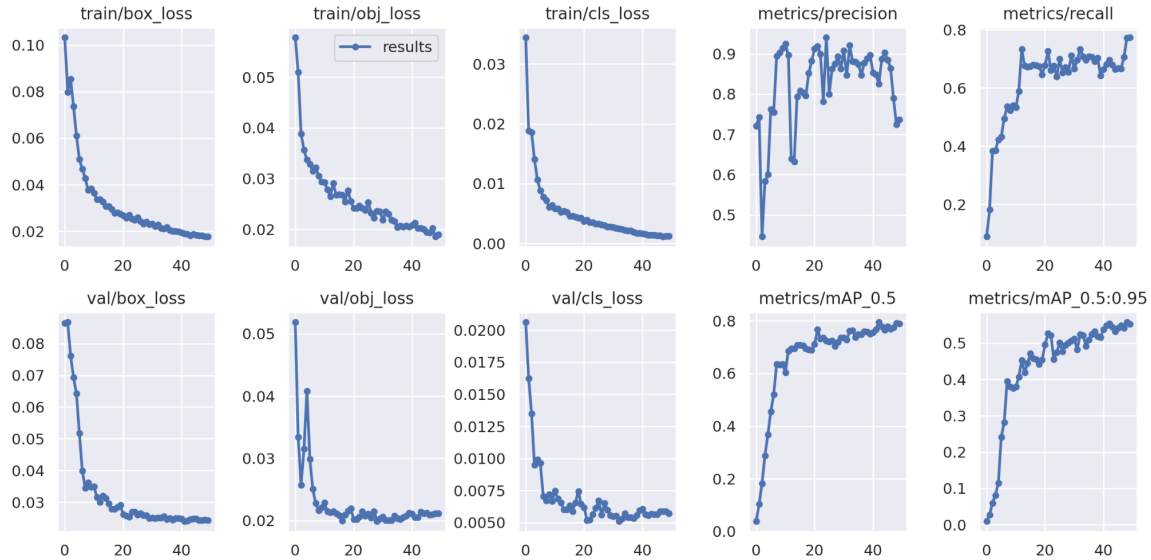

Figure 9. YOLOv5m Model Evaluation Merics

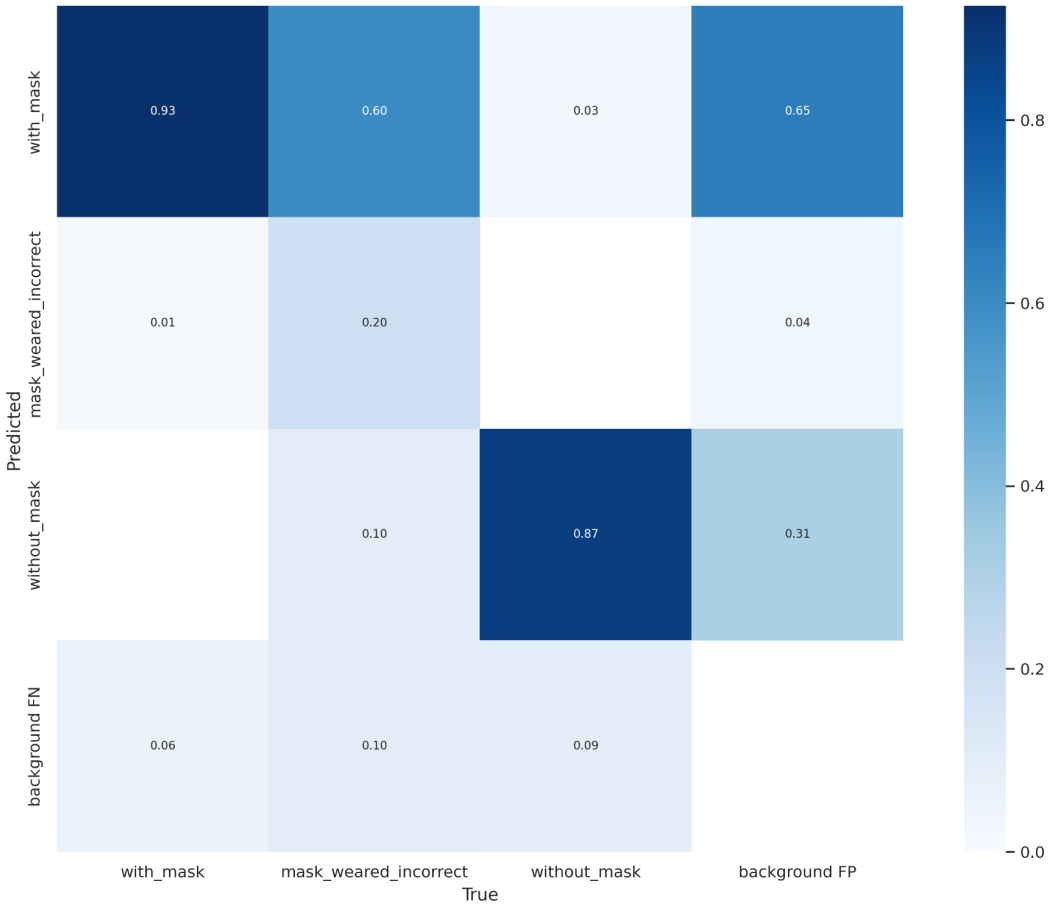

Figure 10. YOLOv5m Confusion Matrix

YOLOv5l(batch size =8)

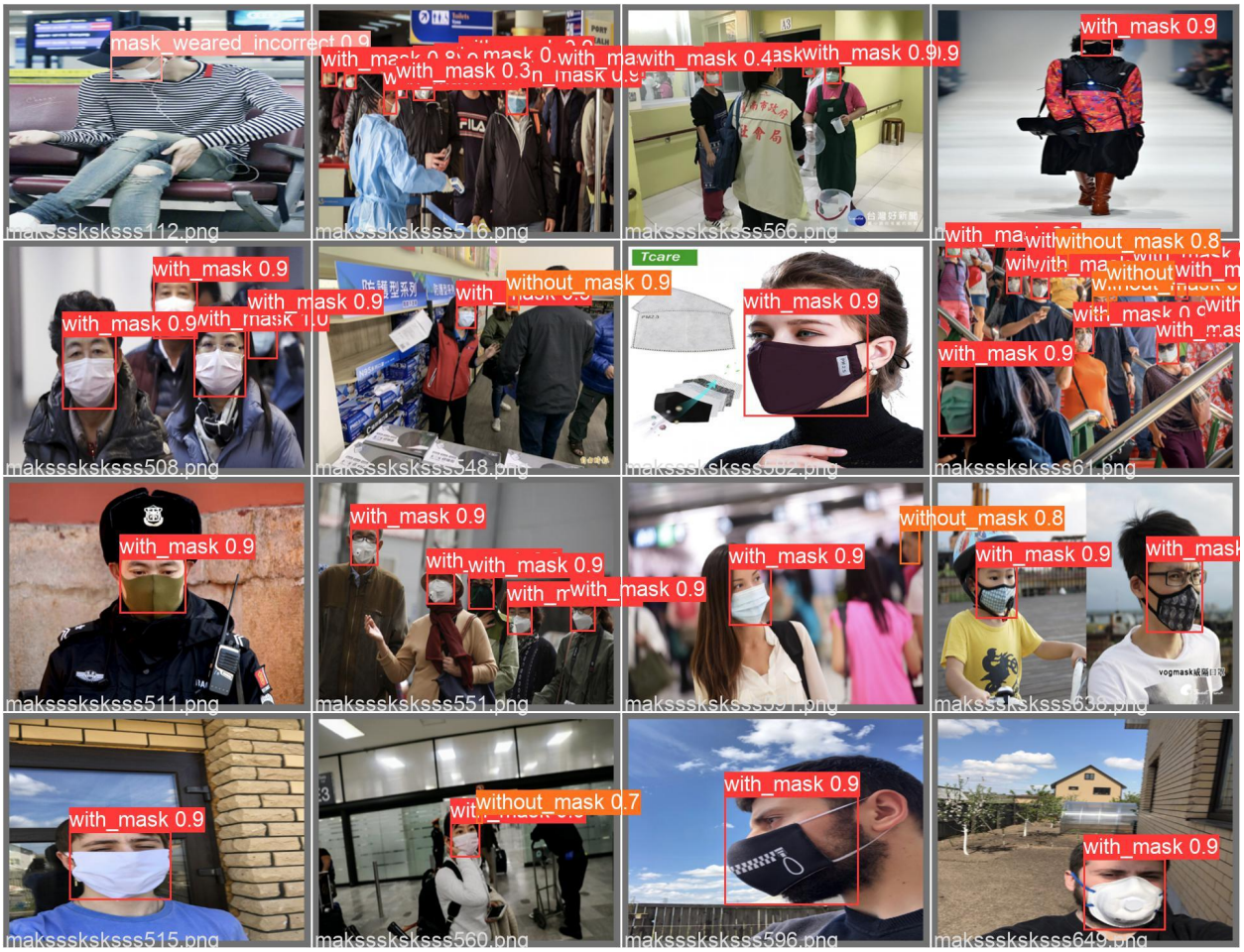

Figure 11. YOLOv5l Prediction Output

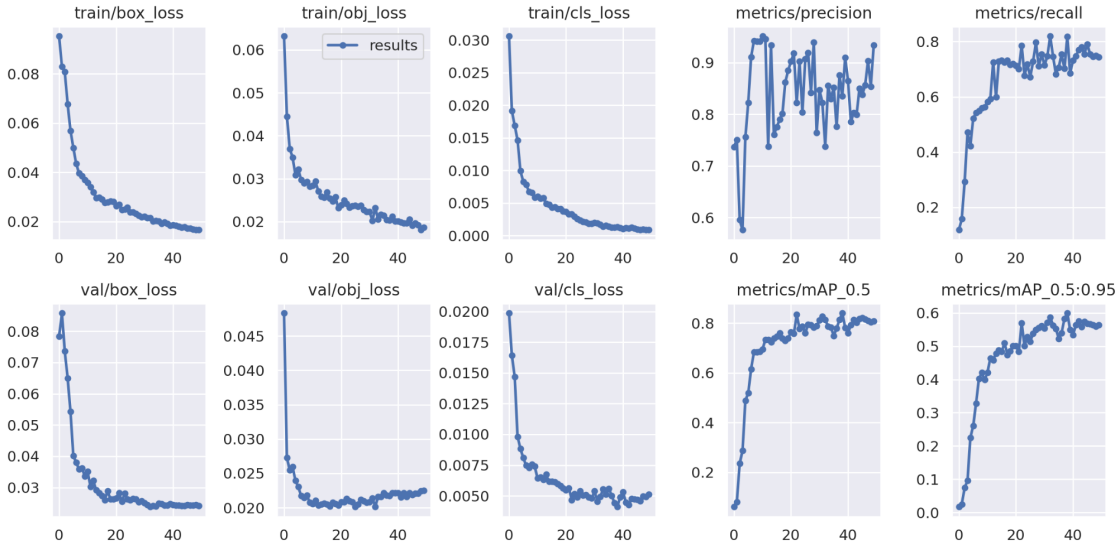

Figure 12. YOLOv5m Model Evaluation Merics

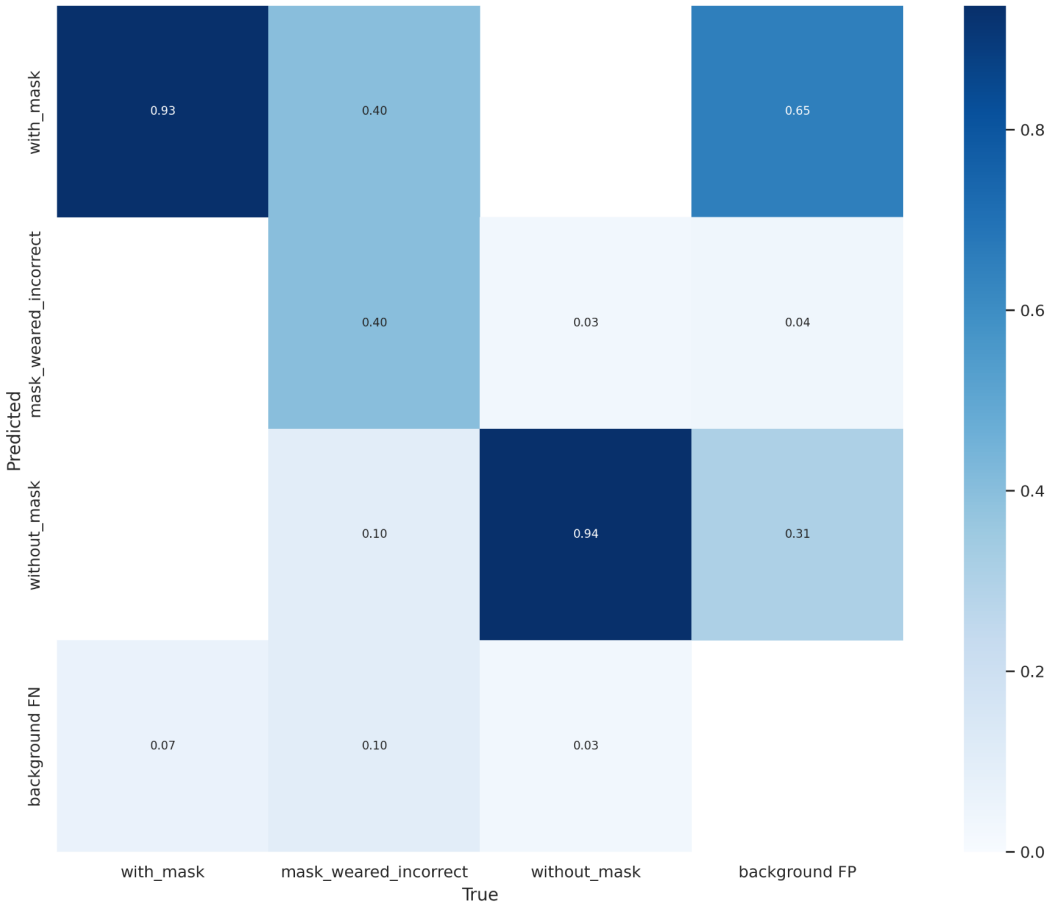

Figure 13. YOLOv5l Confusion Matrix