Face Mask Detection

Saurav Kant Kumar
Illinois Institute of Technology
skumar66@hawk.iit.edu

Abstract

Face masks play a crucial role in protecting the health of individuals against respiratory diseases. It is also one of the few precautions available for COVID-19 in the absence of immunization. With this dataset, it is possible to create a model to detect people wearing masks, not wearing masks, or wearing masks improperly. This work utilizes deep learning based algorithms for object detection.

1. Introduction

Object Detection is a technique used to locate the presence of objects with a bounding box and types or classes of the located objects in an image.

There are a number of traditional and deep learning based approaches for object detection but Deep Learning based algorithms almost always outperform traditional algorithms if enough compute and data is available.

There are two types of frameworks available in deep learning object detection models. The first framework is regression-based and consists of models like MultiBox, AttentionNet [11], SSD [6], YOLO [8] family of algorithms. The second framework is region proposal based and it consists of models like SPP-NET [3], RCNN [2] [1] [9] family of algorithms.

This work uses YOLO v5 [8][4] to detect whether a person is wearing mask, not wearing mask, or the mask is worn incorrectly. YOLO reads the image only once hence it is very fast compared to region-based algorithms but it's performance is slightly lower than regions-based algorithms.

2. Data Source

The Data set is taken from kaggle. [7] There are total 853 pre-annotated images. The annotation data is available in MS COCO format which needs to be converted to appropriate format before fitting a model.

(a) With Mask (b) Without Mask (c) Mask weared incorrect

Figure 1: Sample Input

(a) With Mask (b) Without Mask (c) Mask weared incorrect

Figure 2: Sample output from YOLOv5s

3. EDA

There are images with many boxes and with different classes per image. Let's check the distribution of the bounding boxes per image.

Figure 3, shows that the classes are highly imbalanced. Almost 80 percent of the labels in the images belong to the class "with mask". Almost 17 percent of the labels in the images belong to the class "without mask". Almost 3 percent of the labels in the images belong to the class "mask weared incorrect"

Figure 4, shows that most images have less than 5 bounding boxes per image but there are images with over 100 boxes per image.

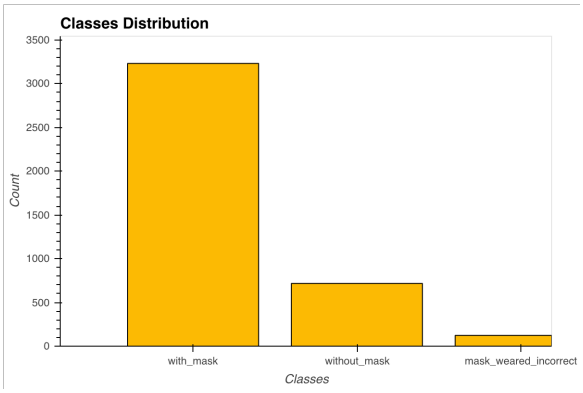

Figure 3: Class Distribution

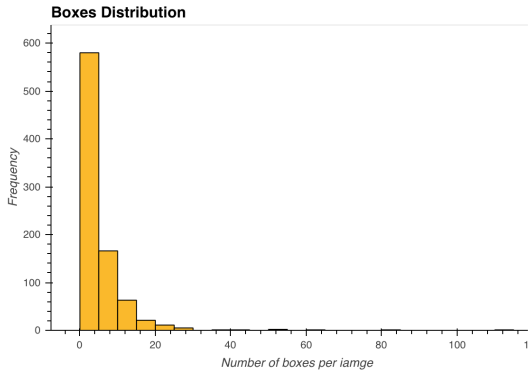

Figure 4: Frequency of boxes per image

4. YOLO (You Only Look Once)

YOLO (You Only Look Once) is a family of models that ("PJ Reddie") Joseph Redmon originally coined with a 2016 publication [8]. YOLO models are infamous for being highly performant yet incredibly small – making them ideal candidates for realtime conditions and on-device deployment environments.

Redmon research team is responsible for subsequently introducing YOLOv2 and YOLOv3, both of which made continued improvement in both model performance and model speed. In February 2020, Redmon noted he would discontinue research in computer vision.

In April 2020, Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao introduced YOLOv4, demonstrating impressive gains.

The approach involves a single neural network trained end to end that takes a photograph as input and predicts bounding boxes and class labels for each bounding box directly. The technique offers lower predictive accuracy (e.g. more localization errors), although operates at 45 frames per second and up to 155 frames per second for a speed-optimized version of the model.

The model works by first splitting the input image into a grid of cells, where each cell is responsible for predicting a bounding box if the center of a bounding box falls within the cell. Each grid cell predicts a bounding box involving the x, y coordinate and the width and height and the confidence. A class prediction is also based on each cell.

For example, an image may be divided into a 7×7 grid and each cell in the grid may predict 2 bounding boxes, resulting in 94 proposed bounding box predictions. The class probabilities map and the bounding boxes with confidences are then combined into a final set of bounding boxes and class labels. Figure 5 summarizes the two outputs of the model.

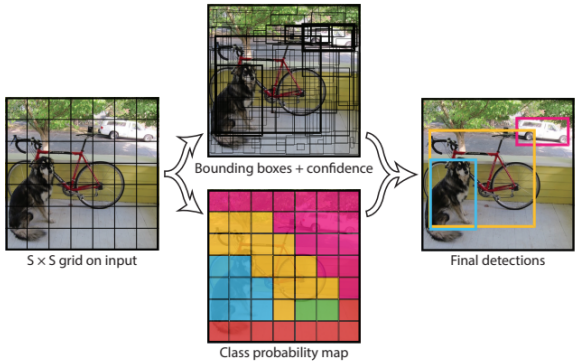

Figure 5: Summary of Predictions made by YOLO Model

If the image is divided into an SxS grid and each cell in the grid may predict B bounding boxes, and there are three labels(C=3) in the image(with mask = 1, without mask=2, mask worn incorrectly=3) then the output of the cells in the grid can be encoded in a tensor of size SxSx(5B+C). It is illustrated in figure 6.

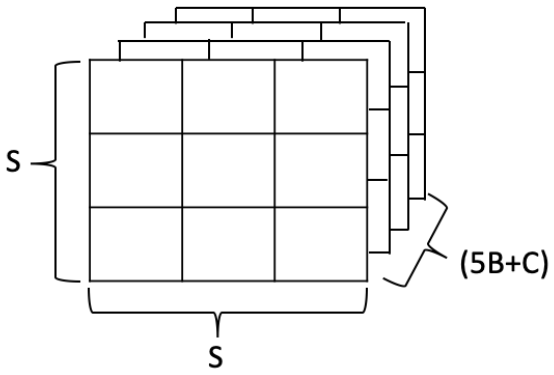

Figure 6: Encoding Multiple Bounding Boxes

5. YOLOv5

5.1. Introduction

Glenn Jocher released YOLOv5 with a number of differences and improvements. (Notably, Glenn is the creator of mosaic augmentation, which is an included technique in what improved YOLOv4.) The release of YOLOv5 includes four different models sizes: YOLOv5s (smallest), YOLOv5m, YOLOv5l, YOLOv5x (largest).

First, this is the first native release of models in the YOLO family to be written in PyTorch first rather than PJ Reddie's Darknet. Darknet is an incredibly flexible research framework, but it is not built with production environments in mind. It has a smaller community of users. Taken together, this results in Darknet being more challenging to configure and less production-ready.

Because YOLOv5 is implemented in PyTorch initially, it benefits from the established PyTorch ecosystem: support is simpler, and deployment is easier. Moreover as a more widely known research framework, iterating on YOLOv5 may be easier for the broader research community. This also makes deploying to mobile devices simpler as the model can be compiled to ONNX and CoreML with ease.

Second, YOLOv5 is fast – blazingly fast. In a YOLOv5 Colab notebook, running a Tesla P100, we saw inference times up to 0.007 seconds per image, meaning 140 frames per second (FPS)! By contrast, YOLOv4 achieved 50 FPS after having been converted to the same Ultralytics PyTorch library.

Third, YOLOv5 is accurate. In our tests on the blood cell count and detection (BCCD) dataset, we achieved roughly 0.895 mean average precision (mAP) after training for just 100 epochs. Admittedly, we saw comparable performance from EfficientDet and YOLOv4, but it is rare to see such across-the-board performance improvements without any loss in accuracy.

Fourth, YOLOv5 is small. Specifically, a weights file for YOLOv5s is 27 megabytes. Our weights file for YOLOv4 (with Darknet architecture) is 244 megabytes. YOLOv5s is nearly 90 percent smaller than YOLOv4. The biggest YOLOv5 implementation, YOLOv5l, is 192 MB. This means YOLOv5 can be deployed to embedded devices much more easily.

The YOLOv5s is the fastest one, but the AP is relatively less accurate. However, this model is also a good choice if the focus of object detection is on larger targets and less complex scenarios, which are speed oriented. The other three networks of YOLOv5 series are based on YOLOv5s to continuously deepen and widen the network, the AP is also continuously improved, but the speed will become slower.

The input of YOLOv5 adopts the same way of mosaic data enhancement as YOLOv4. Mosaic refers to the method of CutMix data enhancement [12]. However, CutMix only takes use of two images for splicing, while mosaic data enhancement utilizes four images, which are randomly scaled, cropped, and resized. In addition, YOLOv5 is also optimized in terms of adaptive image scaling.

5.2. Architecture

As YOLO v5 is a single-stage object detector, it has three important parts like any other single-stage object detector.

- Model Backbone
- Model Neck
- Model Head

Model Backbone is mainly used to extract important features from the given input image. In YOLO v5 the CSP Networks [10] are used as a backbone to extract rich in informative features from an input image.
Model Neck is mainly used to generate feature pyramids. Feature pyramids help models to generalized well on object scaling. It helps to identify the same object with different sizes and scales. Feature pyramids are very useful and help models to perform well on unseen data. In YOLO v5 PANet [5] is used for as neck to get feature pyramids.
The model Head is mainly used to perform the final detection part. It applied anchor boxes on features and generates final output vectors with class probabilities, objectness scores, and bounding boxes. In YOLO v5 model head is the same as the previous YOLO V3 and V4 versions.

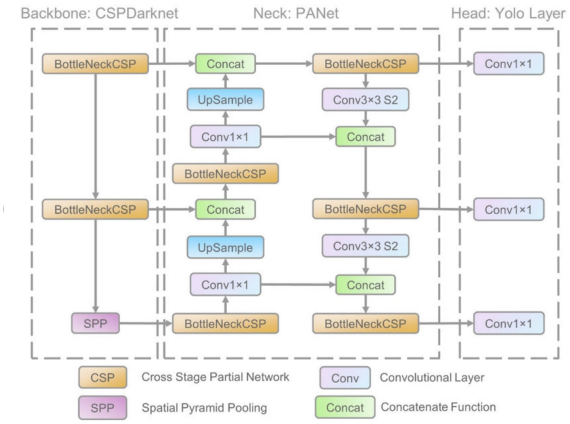

Figure 7: YOLOv5 Architecture

5.3. Activation Function

YOLO v5 authors decided to go with the Leaky ReLU and Sigmoid activation function. In YOLO v5 the Leaky ReLU activation function is used in middle/hidden layers and the sigmoid activation function is used in the final detection layer.

5.4. Optimization Function

In YOLO v5, the default optimization function for training is SGD. However, you can change it to Adam by using the "— — adam" command-line argument.

5.5. Cost Function or Loss Function

In the YOLO family, there is a compound loss is calculated based on objectness score, class probability score, and bounding box regression score. Ultralytics have used Binary Cross-Entropy with Logits Loss function from PyTorch for loss calculation of class probability and object score. We also have an option to choose the Focal Loss function to calculate the loss. You can choose to train with Focal Loss by using "fl gamma" hyper-parameter.

6. Data Preparation for YOLOv5

6.1. Convert annotation to appropriate format

YOLOv5 expects annotated data in txt format. Hence input xml annotated data is converted to txt format.

Figure 7 shows sample input annotation for an input image. Here, xmin, ymin are the coordinates of the top left corner and xmax, ymax are the coordinates of the bottom right corner of the bounding box.

```
<annotation>
    <folder>images</folder>
    <filename>maksssksksss313.png</filename>
    <size>
        <width>400</width>
        <height>400</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>with_mask</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>143</xmin>
            <ymin>39</ymin>
            <xmax>309</xmax>
            <ymax>257</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 8: Sample input annotation

Figure 8 shows the converted output annotation format corresponding to the input annotation. The first value indicated the class of the object(0 for with mask, 1 for without mask, 2 for mask worn incorrect). Second and Third values indicate the normalized x and y coordinates of center of the bounding box. Fourth and Fifth values indicate the normalized height and width of the bounding box.

```
0 0.5649999873712659 0.36999999172985554 0.41499999072402716 0.5449999878183007
```

Figure 9: Sample output annotation

6.2. Data Split

Input images and corresponding converted annotations are split and stored into train, test, and validation sets. There are 678 entries in train set, 85 in validation set, 85 in test set.

6.3. Configure yaml file for training

facemask.yaml file is configured with paths to train and validation images, number of classes, and the names of classes.

7. Evaluation Metrices

7.1. Intersection over union(IOU)

It is an evaluation metric used to measure the performance of an object detector on a particular dataset.

In order to apply Intersection over Union to evaluate an object detector we need:

- The ground-truth bounding boxes (i.e., the hand labeled bounding boxes from the testing set that specify where in the image our object is).
- The predicted bounding boxes from our model.

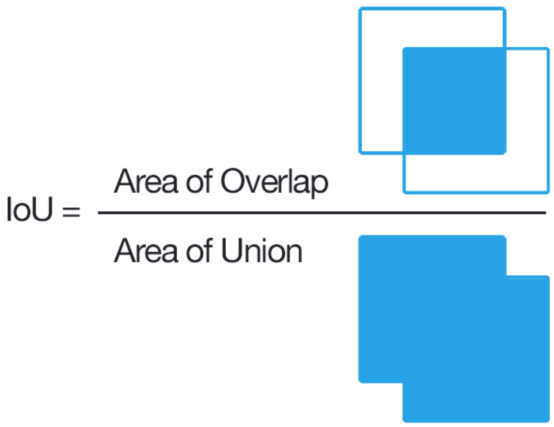

Figure 10: Computing the Intersection over Union

IoU is 0 when there is a no overlap between the predicted and ground-truth boxes. The IoU is 1 when the 2 boxes fit each other completely.

To objectively judge whether the model predicted the box location correctly or not, a **threshold** is used. If the model predicts a box with an IoU score greater than or equal to the **threshold**, then there is a high overlap between the predicted box and one of the ground-truth boxes. This means the model was able to detect an object successfully. The detected region is classified as Positive (i.e. contains an object).
The mAP compares the ground-truth bounding box to the detected box and returns a score. The higher the score, the more accurate the model is in its detections.

7.2. Non-Max Suppression

It is a technique used to suppress redundant bounding boxes when there are multiple detections of the same object. In order to perform non-max suppression the bounding box with highest probability is selected and other bounding boxes which have higher overlap with this bounding box(IOU greater than 0.3) are suppressed.

For example in Figure 11, there are 3 bounding boxes detected for the same class. Among these 3 bounding boxes the bounding box in maroon color has the highest probability(0.96), the bounding box in green color has probability value 0.72, and the bounding box in blue color has probability value of 0.72. Here we measure the IOU value between maroon box and green box and between maroon box and blue box. If these IOU values are greater than certain **threshold** value, these green and blue boxes are suppressed. The output after non max suppression is shown in figure 12.

7.3. Average Precision (AP):

The average precision (AP) is a way to summarize the precision-recall curve into a single value representing the average of all precisions. The AP is calculated according to the next equation. Using a loop that goes through all precisions/recalls, the difference between the current and next recalls is calculated and then multiplied by the current precision. In other words, the AP is the weighted sum of precisions at each threshold where the weight is the increase in recall.

7.4. Mean Average Precision (mAP):

To calculate the mAP, start by calculating the AP for each class. The mean of the APs for all classes is the mAP.

Figure 11: Multiple bounding box detection

Figure 12: Output after non-max suppression

$$AP = \sum_{k=0}^{k=n-1} [Recalls(k) - Recalls(k+1)] * Precisions(k)$$
$$Recalls(n) = 0, Precisions(n) = 1$$
$$n = Number\ of\ thresholds.$$

Figure 13: Computing AP

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$$AP_k = \text{the AP of class } k$$

$$n = \text{the number of classes}$$

Figure 14: Computing mAP

8. Results and Discussions

The Precision Recall curve illustrates Average Precision scores for all three classes and mAP score for the algorithm at IOU threshold 0.5.

- Average Precision for "With Mask" class = .882
- Average Precision for "Without Mask" class = .738
- Average Precision for "Incorrect Mask" class = .522
- Mean Average Precision for the algorithm = 0.714

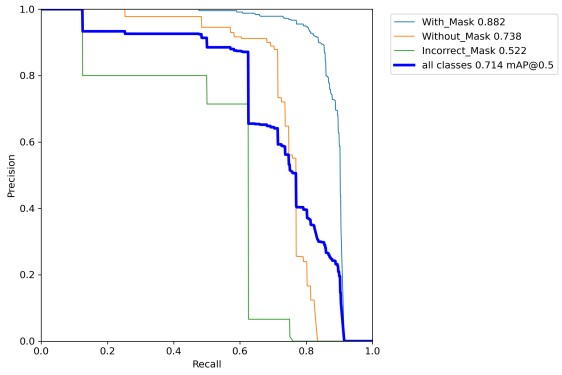

Figure 15: Precision Recall curve @ IOU=0.5

The Average Precision value for the category "With Mask" is maximum because the number of training samples for this class are maximum. The Average Precision value for the class "Incorrect Mask" is minimum because the number of training samples for this category is minimum.The Average Precision value for the class "Without Mask" is in between the above two categories because the number of samples for this class is between the above two categories.

Figure 16 shows that Training Loss for Classification decreases continuously with each epoch.

Figure 17 shows that Validation Loss for Classification decreases initially, fluctuates for some time, and shows minimum value around 230 epochs.

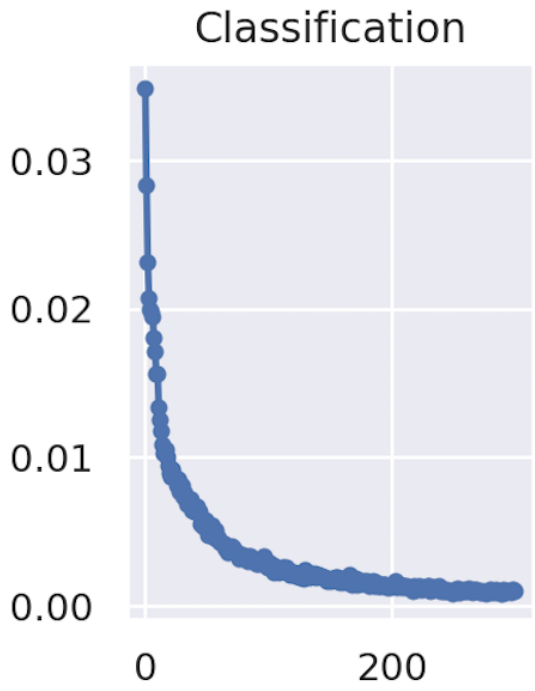

Figure 16: Training Loss for Classification

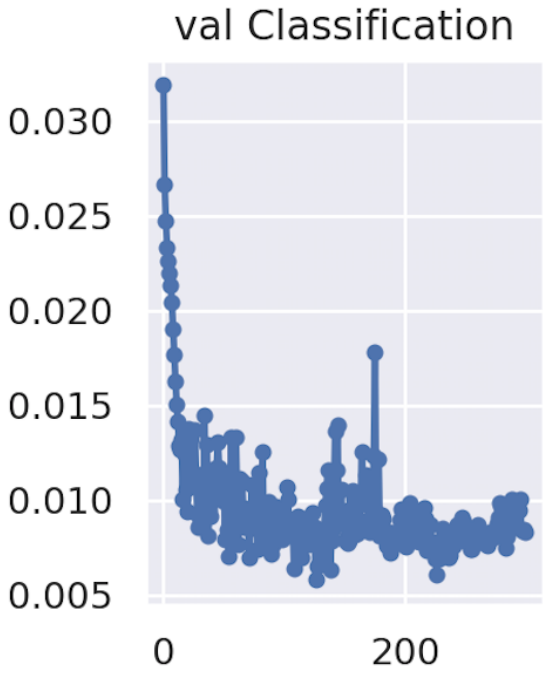

Figure 17: Validation Loss for Classification

Figure 18 shows that Training Loss for Objectness decreases continuously with each epoch.

Figure 19 shows that Validation Loss for Objectness

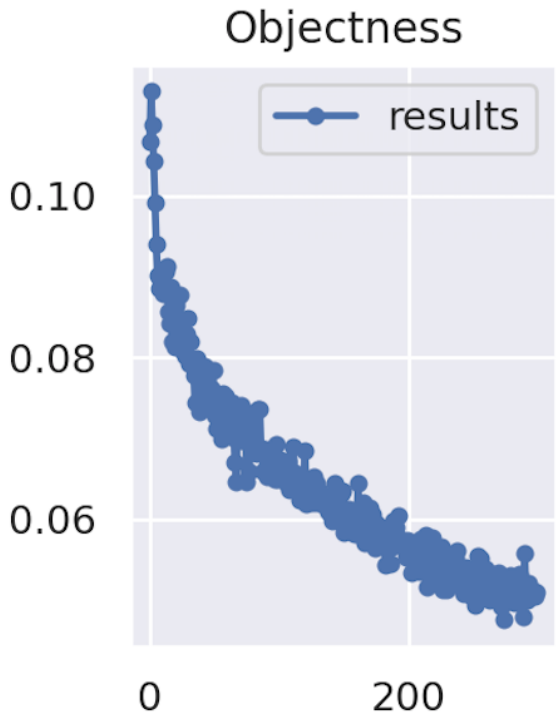

Figure 18: Training Loss for Objectness

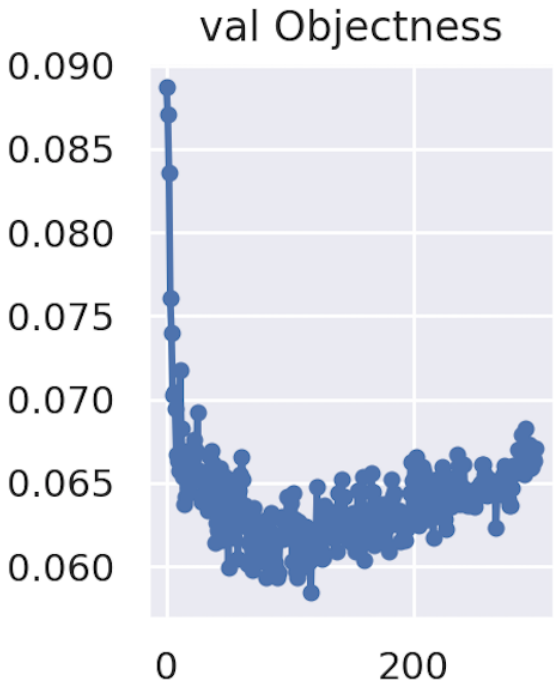

Figure 19: Validation Loss for Objectness

decreases initially, reaches minimum around 120 epochs and then starts increasing.

9. Conclusion and Future Work

In this paper, YOLOv5 model was employed to implement object detection of face masks. Experimental results show that YOLOv5 algorithm is superior in almost all indicators.

In this project, the number of samples for class "Without Mask" and "Mask Weared Incorrect" are relatively small due to which few objects of these categories are not detected. In order to improve the performance more data of these two classes could be added.

The speed to YOLOv5 is very fast and it is able to perform detection in real-time. But when the size of object is small it sometimes fail to detect it. To overcome this shortcoming, i plan to implement Faster-RCNN [9] on this dataset and create an ensemble to make inference.

References

[1] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[2] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[3] Kaiming He et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916.

[4] Glenn Jocher et al. *ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements*. Version v3.1. Oct. 2020. DOI: 10.5281/zenodo.4154370. URL: https://doi.org/10.5281/zenodo.4154370.

[5] Shu Liu et al. "Path aggregation network for instance segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8759–8768.

[6] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[7] *Mask Dataset*. URL: https://makeml.app/datasets/mask.

[8] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[9] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *arXiv preprint arXiv:1506.01497* (2015).

[10] Chien-Yao Wang et al. “CSPNet: A new backbone that can enhance learning capability of CNN”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 390–391.

[11] Donggeun Yoo et al. “Attentionnet: Aggregating weak directions for accurate object detection”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2659–2667.

[12] Sangdoo Yun et al. “Cutmix: Regularization strategy to train strong classifiers with localizable features”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6023–6032.