

Ameer Ali

Gyanesh Tiwari

Yan Kai Lim

<https://github.com/TsarAli/cs584-Final-Project>

Alternate Cognitive Mask Detection Via Machine Learning Techniques

Abstract

Currently we are in the midst of the COVID-19 global pandemic. There is research that proves that wearing masks will help to reduce the spread of COVID-19 to a certain extent and therefore some countries have mandated the need to wear masks in some public spaces. However, there are some problems faced with manual enforcement, especially when we need to monitor compliance in real time, cost and manpower becomes an issue. Our solution to this is making use of an object detection deep learning algorithm to be better able to analyze if a person is compliant to mask wearing.

Introduction

Before we begin, object detection is a computer vision method for detecting instances of objects in images or videos. We will be using deep learning for our solution since deep learning can perform accurately especially when given sufficient data.

More specifically we will be leveraging the YOLOv5 object detection framework as YOLOv5 is better able to analyze massive streams of data in real-time, can easily be implemented and has proven to be very highly accurate even as compared to its peers.

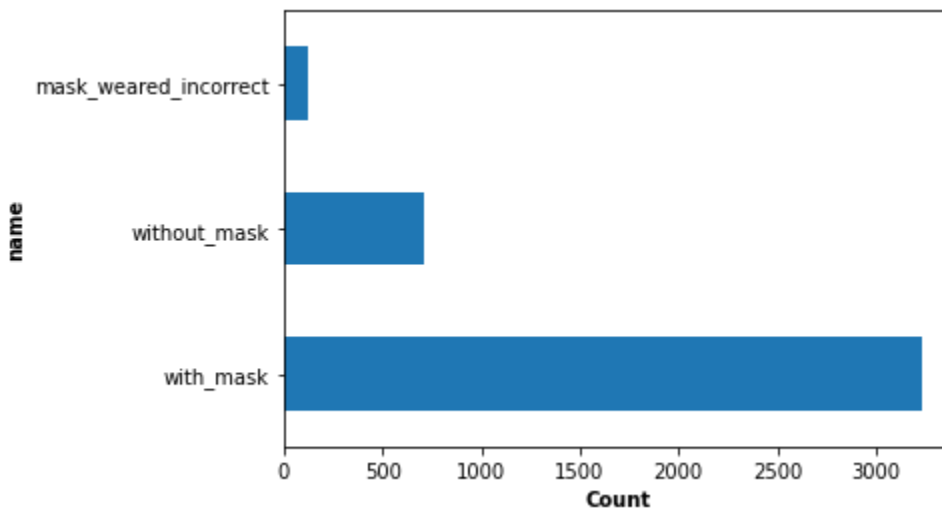
Problem Description

The main problem we are trying to address is the detection of people who are wearing masks, not wearing a mask or wearing a mask incorrectly. An issue that was originally unknown but

became more widespread during the pandemic was face detecting while wearing a mask. Many countries implemented a mask mandate as part of their response to Covid which hampered this field as the mask would limit face detection. An example of this would be a person trying to unlock their smartphone with face detection while they have a mask on. The phone would simply not unlock as the cameras could not recognize the person. The method we used to solve this was using the YOLO algorithm. YOLO is a very fast RCNN based algorithm that can be used to solve this problem.

Data preprocessing and analysis

Our data is taken from Kaggle. [2] The dataset has already been pre annotated so no manual annotation has to be done on our end. The data includes 3 labels including 'With mask', 'Without mask' and 'Mask wear incorrect'. The number of images provided was 853.



Modeling

YOLO makes predictions on an entire image by passing it to the Convolutional Neural Network. It first divides the images into N grids with an equal dimension of SxS. Each grid then predicts B bounding box coordinates relative to the coordinates of the cell. It then returns the bounding boxes above the confidence threshold.

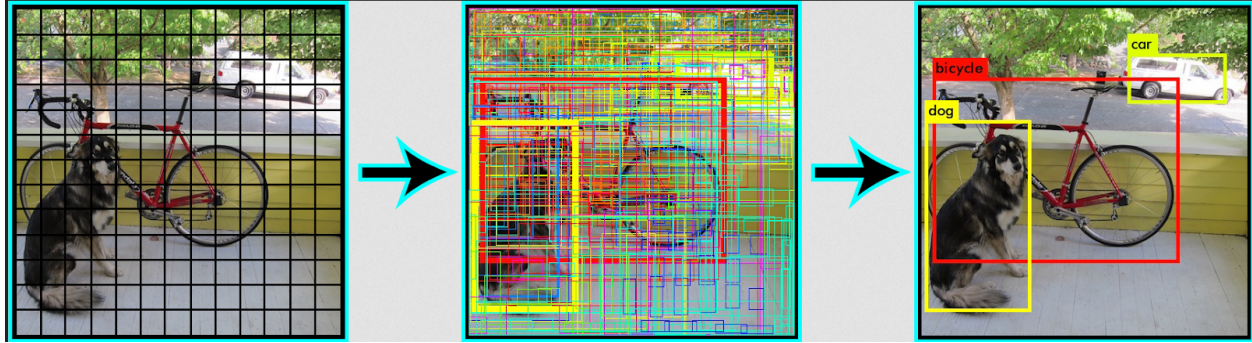


Figure 1: Example of bounding box. Bounding boxes from YOLO are shown with red meaning that there is a high likelihood of an object.

Yolo uses a single regression module in the format of a vector where Y is the final vector representation for each bounding box: $Y = [p_c, b_x, b_y, b_h, b_w, c_1, c_2, c_3, \dots, c_n]$. p_c is the probability score of the grid containing an object, b_x, b_y refers to the x and y coordinates of the center of the bounding box with respect to the grid cell, b_h, b_w refers to the height and width of the bounding box with respect to the grid cell, $c_1, c_2, c_3, \dots, c_n$ represents the class an object in the grid belongs to. YOLO supports having as many classes as required. The architecture of YOLO uses 24 convolutional layers followed by 2 fully connected layers. The input image is resized into 448×448 before going through the convolutional network. First a 1×1 convolution is applied to reduce the number of channels which is then followed by a 3×3 convolution to have the output be cuboidal. The network uses an ReLU activation function with the final layer using a linear activation function. YOLO also uses regularization techniques such as batch normalization and dropout to prevent overfitting.

For the hyperparameters, we set our batch size as 16 with 300 epochs as shown.

```
!python /content/drive/MyDrive/cs584/Project/yolov5/train.py --img 320 --batch 16 --epochs 300 --data /content/drive/MyDrive/cs584/Pr
```

Evaluation

There are 2 main evaluation metrics that we use to evaluate the model trained, Intersection over Union (IOU) and Mean Average Precision (mAP).

IOU is the metric used to calculate the localization accuracy and localization errors. To calculate IOU, we take the intersection area between the bounding box for a prediction and the ground truth bounding boxes of the same area. We then calculate the union between the two boxes as well. Then we take the intersection divided by the Union which gives the ratio of the overlap to the total area.

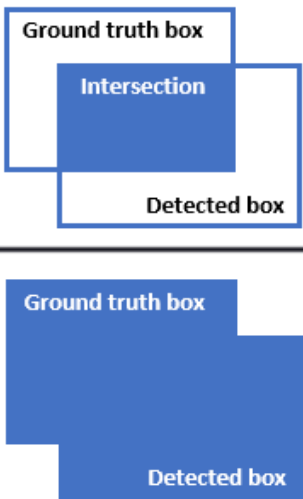
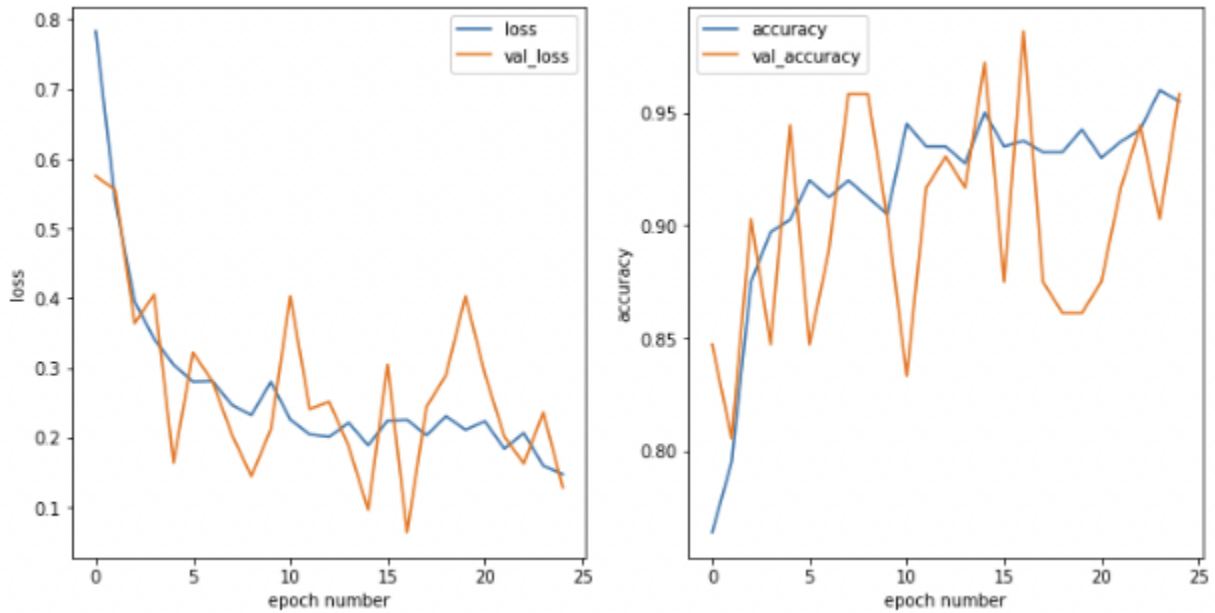
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{\text{Intersection}}{\text{Ground truth box} \cup \text{Detected box}}$$


Figure 2: IOU visualization.

YOLOv5 also uses non-max suppression. It is a technique that is 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 the highest probability is selected and another bounding box which has a higher overlap with the selected bounding box (has an IOU greater than 0.3) is suppressed.

YOLOv5 also uses average precision (AP) and mean average precision (mAP). The average precision (AP) is a way to summarize the precision-recall curve into a single value representing the average of all precisions. To calculate the mAP, start by calculating the AP for each class. The mean of the APs for all classes is the mAP.

Results



Conclusion and Future Work

We can see that YOLOv5 works well for our use case even when given a small dataset. We can easily embed the model that we train onto python based web applications in order to leverage the power of YOLOv5.

References:

- Figure 1: Bandyopadhyay, H. (2022, October 7). Yolo: Real-time object detection explained. V7. Retrieved November 9, 2022, from <https://www.v7labs.com/blog/yolo-object-detection#h4>
- Figure 2: Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016, May 9). You only look once: Unified, real-time object detection. arXiv.org. Retrieved November 9, 2022, from <https://arxiv.org/abs/1506.02640v5>