Mask Object Detection (*Lab 4 CS 5330*)

John David Maunder
maunder.j@northeastern.edu

Abstract

For the lab4 research project, we will be using the "Mask Dataset" dataset from MakeML to train a preconfigured neural network to see if people are wearing a mask, and if they are, are they wearing this mask correctly. Sars-Covid-2 has required people to wear a mask to minimize the transmission of this virus. Therefore, building a detection tool to see if people are wearing a mask correctly in a public setting is necessary to maximize the usefulness of public health measures. The dataset used to for this mask detection Neural Network is the "Mask Dataset" from MaskML. The Mask Dataset contains 853 images which are broken up into three categories labelled "with mask", "Without mask" and "Mask wearied incorrect". The dataset includes images of people in large groups in a public space, images of people in a small group setting, individual photos of a person's face ("a selfie") and various photos of people from advertisements (a picture of a picture). Before training, data cleaning was undertaken to make sure the images included in the dataset were representative of how this neural network would be used in a public setting. After this cleaning, augmentation methods were used to increase the number of training images and were used to make the training images more robust. These training images were used as the final dataset to train the YoloV3 Darknet and the YoloV5 convolutional neural networks (CNN's).

1. Introduction

In March 2020, the World Health Organization declared that Sars-Covid2 was a virus that warranted the label of a "pandemic". Due to the severity of this disease, multiple governments worldwide including the Canadian government implemented a lockdown which essentially prohibited people from congregating in a public setting. Even with these restrictions, people still had to travel to essential places like doctor appointments, the grocery store or work. Therefore, to minimize the spread of covid, governments required people to wear masks in public places to mitigate the spread of covid. Therefore, to ensure people are complying with these health requirements a video recognition system can be used to identify whether people are wearing a mask. This will allow government officials to reconfigure their policies to ensure health requirements are being met. To create an image recognition system that can determine whether someone is wearing a mask we can use the Mask Dataset, dataset from MakeML as our training dataset for the Yolov3 darknet and Yolov5 CNNs. To make our models more efficient, we can clean the dataset and preform data augmentations on this dataset. Subsequently, we will setup Yolov3 and YoloV5 in google collab. We will then run these models until MAP (Mean Average Precision) is maxed. Lastly, we will look at the results and determine if the algorithms need to be fined tuned and re-run again.

2. Data Cleaning and Augmentation Process

2.1. Data Cleaning

Since the dataset is provided for us, we cannot add any new images to the dataset. Therefore, the only way we can increase the accuracy of our neural network is to add images through augmentation, and to remove non representative images. Before we can augment the data, we need to clean the data to make sure it is representative of the images we are trying to capture in the real world. Cleaning the dataset will first involve importing the dataset into Roboflow. We used Roboflow to augment and clean our data since it's the easiest free online tool I found to preform these tasks. Once the dataset is in Roboflow, we can go through the dataset and determine whether a certain image should be used as a training, validation or testing image. Since we want our image recognition system to be used in a public area, we will want to remove photos that aren't representative of this area. For example, images 1 and 2 were removed from this dataset since they are advertisements (photos of photos) and are not real photos taken in a public setting. These photos will most likely cause the algorithms to miscalculate what a person looks like as the texture of an advertisement will be different that what a person will look like in a public setting (non photoshopped image). This decision criteria caused 15 images to be removed from the dataset. We also removed certain images where the people in the image were too small to be recognized. For example, in image 3 the number of pixels in the image is not enough to tell if the people in the background have masks on or not. This was removed from the dataset since it was difficult to determine if the people in the background had masks on or not. Since a human cant tell whether a person has a mask or not, it is not advised to put it in into the neural network as there isn't a real ground truth. While we are removing images from the training set, we are also letting certain images remain. Images whose background is slightly blurry were left in as they were used to make the testing dataset more robust. In a real-world setting, not every single person will be facing the camera at an optimal angle. Because of this, we will want to include images that aren't perfect so the CNN can reweight its neurons based on these blurry images. For example, if we look at images 4-5, most of the people in the image can be easily seen as wearing a mask or not. For example, the people in the background who are circled in the blue outline are blurry. Because of this, it is hard to tell if they are wearing a mask or not. For the sake of this lab, we did not put any bounding boxes on these people even though they may or may not be wearing masks. By not labelling these parts of these images, we are adding noise to the dataset, which in turn will make the dataset more robust going forward. We also left in images where people took selfies. We left these images in, even though its not representative of real-world images because these images are useful in helping the model train. These selfie images are essentially a zoomed in image of a normal person walking by a stationary camera. This zoomed in image will allow the CNN to form patterns which in turn can be used to find the same pattern in a person farther away from the camera. This is important since people who are farther away from the camera will have less pixels. The lack of pixels will make it hard for the CNN's to determine a pattern which it can use to classify people []. The last step in cleaning up the dataset before we move on to augmentation is to make sure all images have the correct bounding boxes. Since determining the ground truth for any image must be done by humans, it is always advisable to double check to ensure the bounding boxes are correctly labelled. This process resulted in labelling more images in the background than the initial dataset had. We also removed other images for simplicity sake. For example, in image 6, the person in the image is covering his mouth with his hand. The initial dataset had this labelled as with-mask but this image should be removed since this person isn't wearing a mask. We could have labelled this image as not wearing a mask correctly, but since it's the only image in the dataset where the person is not using a proper mask, it is beneficial for the CNN to not include this image in the training set. Once we have decided which images to included in the dataset, our next step is to determine the training, validation, and testing breakdown. Based on the provided guidelines our breakdown must be 70-15-15. Since our training, validation, split is predefined for us, we can move on to the augmentation step.

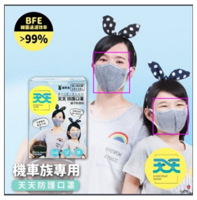


Figure 1. Image 1

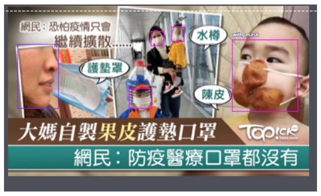


Figure 2. Image 2

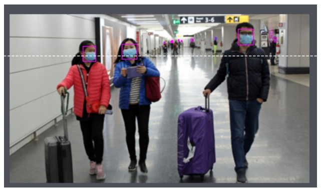


Figure 3. Image 3

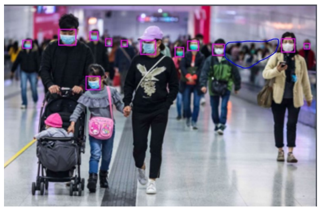


Figure 4. Image 4

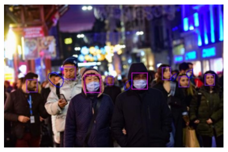


Figure 5. Image 5

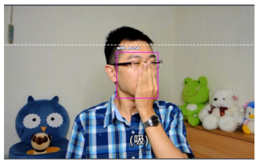


Figure 6. Image 6

2.2. Augmentation Step

We preformed the following augmentation steps on our dataset.

1. Saturation (20 percent increase and decrease)
2. Shear (20 percent increase and decrease)
3. Brightness (20 percent increase and decrease)
4. Exposure (20 percent increase and decrease)
5. Rotation (20 percent increase and decrease)

When we preformed the above augmentation steps (both increasing and decreasing the values by 15 percent), our model underperformed our expectations. The reason for this was due to the shearing and rotation augmentations. When the image was rotated, the pixels where the image was originally located was converted from normal pixels to blackness. This is an issue when effectively building a neural network for two reasons. The first issue is this will not represent what we want the neural network to do in the real world. It is highly unlikely that a stationary camera will be in an area where the camera will have a blind spot (a spot where the live image taken by the camera will yield a black pixel space). Since this situation is unlikely to happen even if the camera is rotating on a fixed plane, we will not want to include images where there is black patches as this unrepresentative of the situation we are trying to achieve. The second issue with having an image with black pixels is the neural networks will be filtering and pooling based on the black pixels. Since the black pixels will be grouped and pooled this will results in non accurate training features being learnt. These two augmentation methods led to a decrease in MAP score for both neural networks. After determining that shearing and rotation would not be great augmentation methods, we retrained the neural networks using the following augmentation methods:

1. Saturation (15 percent increase and decrease)
2. Brightness (20 percent increase and decrease)
3. Exposure (15 percent increase and decrease)
4. Mosaic

Saturation, brightness, and exposure were used to augment the data since the lighting in various locations will vary rapidly. For example, if a camera is looking at a person during the day or during the night the amount of brightness that will be casted onto an individual instance will vary greatly. Therefore, by applying brightness and darkness filters to the dataset this will increase the likelihood that the neural network will be trained on data that is more represented of what would happen in the real world. Furthermore, since every camera has a different configuration and quality, having a CNN that is trained on data with various lighting (in terms of brightness and visibility) will increase the likelihood that CNN will perform better in a real-world setting. We included the mosaic augmentation method due to the heat maps listed below. In images 7-9, the location of the people in the three various classes appeared in the center of the image. The mosaic augmentation will move the bounding boxes of our three classes from being in the center of the images to the corners of the image making the testing images more spread out. This will make our testing data more robust. The last augmentation step we performed on our data was image size reduction as this will increase training speed. We shrunk the image to (40x31). Reducing the size of the image will increase training speed. The result of this augmentation step ended up yielding a 0

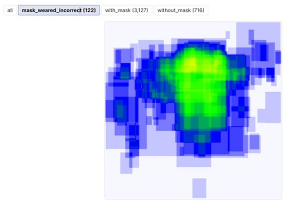


Figure 7. Image 7

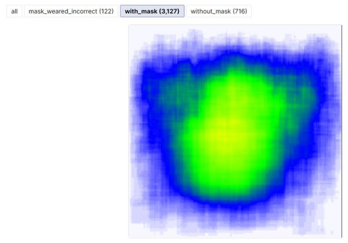


Figure 8. Image 8

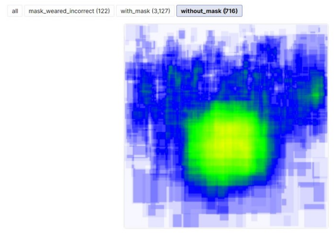


Figure 9. Image 9

3. Convolutional Neural Network

We can use various platforms to train a machine learning model. We can train this model on a local host, a virtual host, or we can use a third party to train our data. We want a third party to be able to run and configure our code. This requirement prohibits us from using a local or third-party host as training platform options. Google collab also has many beneficial features. Google collab makes it easy to download complex models. It also allows the use of the CUDA coding language. CUDA makes it easier to run code on the GPU, which in turn will speed up training time since most training is just matrix multiplication.

3.1. Yolov3

We first trained our data on Yolov3. I chose to use YoloV3 as our first neural network since I have experience using this CNN before. Furthermore, the Darknet backbone of this CNN combined with Yolo's "one look pass" allows this network to be accurate and fast. Yolov3 was able to accurately classify images in a real world setting at a 30 frames per second threshold[3]. Yolov3 can yield three different feature vectors of three different sizes. These three feature vectors are an improvement over the previous iterations of the Yolov3 architecture. The smaller feature vectors allow for smaller objects located at various regions (not the center) to be picked up and identified during training. This is very important for our work since a lot of people will be wearing masks in the background of our training images. Therefore, Yolov3 will not only be able to classify large images but smaller images as well [3]. Since we had our data split up into training, validation and testing prior to importing our data in google collab (due to Roboflow) there was no need to use prewritten code. We just wrote a basic python script to load the data into a "config" file so Yolov3 could find our testing and validation data. After, we made our own custom config" file. We had to change some of the hyperparameters of the file so the neural network can work with three classes instead of 80 (which is the default class value for Yolov3). Once the config file was done, we decided on what preconfigured weights to use. We decided to use the non trained weights for our testing since I wanted this network to be trained from scratch.

We initially tried to tune the following hyperparameters:

1. Batches/Subdivisions
2. Learning Rate
3. Max Batches

Batches/Subdivisions will be the number of images we pass to our network before the weights are updated. A lower batch will mean less images trained per each iteration of our model. This in turn will decrease our training time [2]. There are advantages and disadvantages to a smaller or larger batch size. The smaller batch size will pass less of the same image meaning its less likely our model will overfit, but there also is an issue our model won't see the images enough which mean it will underfit. The Learning rate should be set to "0.001". I initially tried to set this rate to 0.1, but this learning rate became to slow to train the model and thus the model was not finding the local optimum. This resulted in the model not converging. This lack of convergence results in our training error following to below zero, which meant our model learnt nothing. This can be supported by our model not making any predictions once the model finished training. Max batches is Yolov3's version of "epochs" (how many times we pass the data through the neural network). It is advised to pass 2000 times the number of classes you have. We ran 6000 max batches and found that the model would maximize its "MAP" score around 5000. Yolov3 allows you to save the best weights, so if the model starts to deprecate in performance after X number of iterations, the best weights will be saved.

3.2. Yolov5

We chose to use Yolov5 in addition with Yolov3 darkent since its an easy to use high preforming out of the box CNN that takes little time to train. Yolov5 is a relatively new CNN (made in April 2020). It was also combined with ultra-analytics and roboflow to make this model super easy to train.

The hyperparameters for Yolov5 are as follows:

1. Batch size
2. Epochs
3. Config file
4. Initial weight

The only hyperparameter we changed is the epoch size since we wanted to run the model for at least 200 iterations. We left batch size, config size and initial weight size untouched since we did not need to configure the model to run our training set. Since we wanted to run this model from scratch non-pre-configured weights is ideal. The results of training, and inference can be seen in images 10-12 and the inferences can be seen in 13-15. In our inference our Yolov5 model preformed quite well. It was able to get all classes. It did struggle with the advertisement picture as it completely missed both people wearing masks.

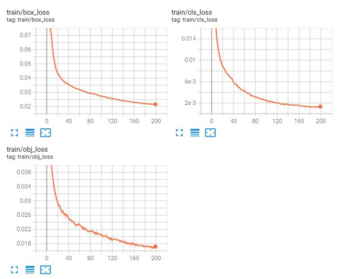


Figure 10. Image 10

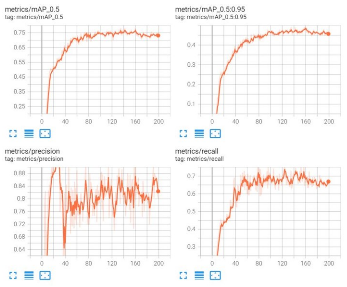
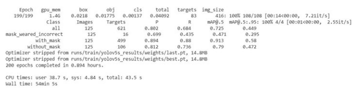


Figure 11. Image 11



```
   Epoch   gpu_mem       box       obj       cls     total   targets  img_size
 199/199      1.4G    0.0218   0.01775   0.00137   0.04092        83       416: 100% 108/108 [00:14<00:00,  7.21it/s]
               Class      Images     Targets           P           R      mAP@.5  mAP@.5:.95: 100% 4/4 [00:01<00:00,  2.55it/s]
                 all         125         621       0.802       0.684       0.725       0.449
mask_weared_incorrect          125          16       0.699       0.435       0.471       0.295
           with_mask         125         499       0.894        0.88       0.913        0.58
        without_mask         125         106       0.812       0.736        0.79       0.472
Optimizer stripped from runs/train/yolov5s_results/weights/last.pt, 14.8MB
Optimizer stripped from runs/train/yolov5s_results/weights/best.pt, 14.8MB
200 epochs completed in 0.894 hours.

CPU times: user 38.7 s, sys: 4.84 s, total: 43.5 s
Wall time: 54min 5s
```

Figure 12. Image 12

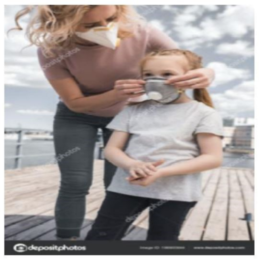


Figure 13. Image 13

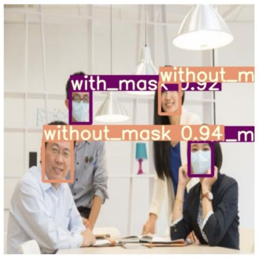


Figure 14. Image 14

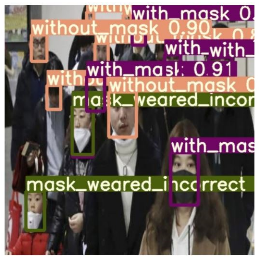


Figure 15. Image 15

4. Results

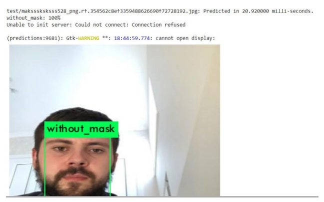
Both CNN's Map scores were brought down greatly by the partly worn mask class. The reason for this is this class had very little training data. In addition, the difference between wearing a mask correctly and wearing a mask incorrectly is two nostrils showing. Since this is a small amount of space to differentiate between two classes, the CNN's had a hard time correctly identifying this one class. Furthermore, if someone has a mask down around their chin, this may not be picked up in an image since a person's head could be blocking half the mask. This reduction in visibility significantly hurt training for this one class. The Mask grouped preform very well. The only issue this class faced was when there were people wearing masks in the background. Since there is less data about these individuals sometimes the CNN's struggled to pick up this class. The non mask group preformed decently well. There was a lot less data for the non mask class than the masked class. If there was more data for this class, the MAP score would have been higher for both CNN's. Again, this can be improved by increasing the amount of data for this class. Lastly both these classes sometimes struggled when a person's head was turned but you could see the outline of the mask in the photo. The CNN's may be picking this up when training and validating. This is hard to fix since the angle at which someone is turned may affect the ground truth. For example, if they were turned at a 45-degree angle, the person may be showing enough of a mask or no mask to be labelled in one of three classes. Since we did not deal with people's heads turn around, this may be affecting the ground truth and the overall score of the CNN's.

4.1. Yolov3

The inference time for each photo was very quick. The algorithm was able to accurately determine whether multiple people were wearing masks in 20ms (see image 16-18). As discussed above the precision and recall scores (image 19 for both wearing and not wearing masks was well done. But as can be seen in image 18, Yolov3 classified the man in the coat as not wearing a mask when it should have been classified as incorrectly wearing a mask. As can be seen in these scores, the algorithm correctly stated that there was a person with an incorrect mask, but the like hood was less than the like hood that the person was not wearing a mask. Lastly, the precision and recall scores showed that we only had 6 true positives and 3 false negatives for this one class. This clearly shows we needed more training data for this one class. The recall score was higher than the precision score for this algorithm. This is most likely due tot he fact that this algorithm was over guessing. The inference results for Yolov3 can be seen in images 20-23.

4.2. Yolov5

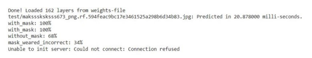
For yolov5 the precision outperformed the recall metric. The reason for this is this algorithm was more conservative when making a guess. For example, in the image 13, both the mother and daughter have masks on, but the neural network didn't make a guess. This resulted in at least two false negatives. Lastly, the probability scores for each person in the training images shows that Yolov5 is certain of a decision when it's deciding. This can be backed up by the MAP score being only slightly better than the MAP 0.5-0.9 score. Since this detection algorithm is not life or death, we generally want both precision and recall scores to be high. If people aren't wearing masks and the algorithm is not detecting this, this is bad since this will increase the like hood that someone will spread covid 19. We also don't want the algorithm to mislabel people as wearing masks when they aren't as this will mis inform health councils about mask wearing.



```
test/maksssksksss528_png.rf.354562c8ef3359488626690f72728192.jpg: Predicted in 20.920000 milli-seconds.
without_mask: 100%
Unable to init server: Could not connect: Connection refused

(predictions:9681): Gtk-WARNING **: 18:44:59.774: cannot open display:
```

Figure 16. Image 16



```
Done! Loaded 162 layers from weights-file
test/maksssksksss673_png.rf.594feac9bc17e3461525a298b6d34b83.jpg: Predicted in 20.878000 milli-seconds.
with_mask: 100%
with_mask: 100%
without_mask: 68%
mask_weared_incorrect: 34%
Unable to init server: Could not connect: Connection refused
```

Figure 17. Image 17

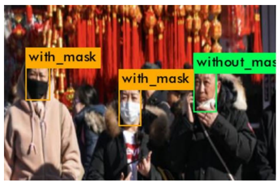
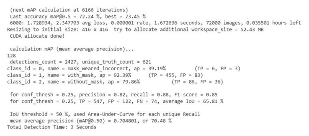


Figure 18. Image 18



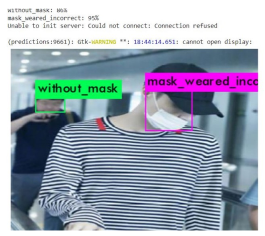
```
 (next mAP calculation at 6166 iterations)
 Last accuracy mAP@0.5 = 72.24 %, best = 73.45 %
 6000: 1.728934, 2.347703 avg loss, 0.000001 rate, 1.672636 seconds, 72000 images, 0.035501 hours left
Resizing to initial size: 416 x 416  try to allocate additional workspace_size = 52.43 MB
 CUDA allocate done!

 calculation mAP (mean average precision)...
128
 detections_count = 2427, unique_truth_count = 621
class_id = 0, name = mask_weared_incorrect, ap = 39.19%          (TP = 6, FP = 3)
class_id = 1, name = with_mask, ap = 92.39%      (TP = 455, FP = 83)
class_id = 2, name = without_mask, ap = 79.86%           (TP = 86, FP = 36)

 for conf_thresh = 0.25, precision = 0.82, recall = 0.88, F1-score = 0.85
 for conf_thresh = 0.25, TP = 547, FP = 122, FN = 74, average IoU = 65.81 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.704801, or 70.48 %
Total Detection Time: 3 Seconds
```

Figure 19. Image 19



```
without_mask: 86%
mask_weared_incorrect: 95%
Unable to init server: Could not connect: Connection refused

(predictions:9661): Gtk-WARNING **: 18:44:14.651: cannot open display:
```

Figure 20. Image 20

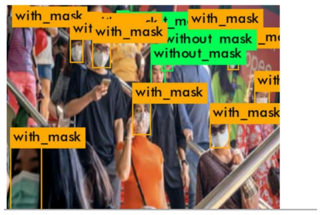


Figure 21. Image 21

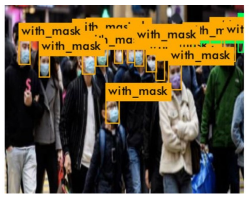


Figure 22. Image 22

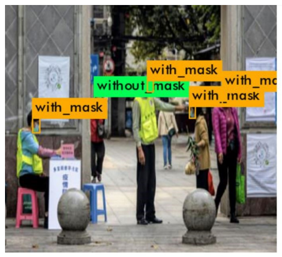


Figure 23. Image 23

5. Conclusion and Future Work

The results from both Yolov3 and yolov5 were well done. I do believe the results could be higher if we made a couple of changes to the model.

1. Get images from a more diverse background
   (a) Include people with different skin tones
   (b) Include people with different facial structures and facial features
2. Acquire more images from a camera in a public setting
3. Acquire more images of people wearing their mask incorrectly
4. Acquire more images of people not wearing a mask in a public setting

By making the above adjustments to our dataset, our model will be trained on a more robust dataset which will in turn improve our results. We can also improve our model going forward by tuning our hyperparameters. We fined tuned our hyper parameters a bit, but by running a script to adjust the hyperparameters we can get a more well-tuned model. For example, by playing with the learning rate, we are more likely to get a model that will find the global minimum when optimizing the weights of our model. In addition, by playing with various batch sizes, we can make sure our model is not over or under fitting. We only trained our model on two different but closely related neural networks which are both one pass CNN's. We could also try training and modelling our data on different CNN architectures like ResNet, googleNet and VGG. By using various architectures, we can see which model is the best at detecting our three classes [1].

References

[1] Various types of convolutional neural network, author=Chatterje, Himadri, journal=Data Versity, link=https://towardsdatascience.com/various-types-of-convolutional-neural-network-8b00c9a08a1b, year=2019.

[2] J. Brownlee. How to control the stability of training neural networks with the batch size. *Machine Learning Mastery*, 2019.

[3] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.