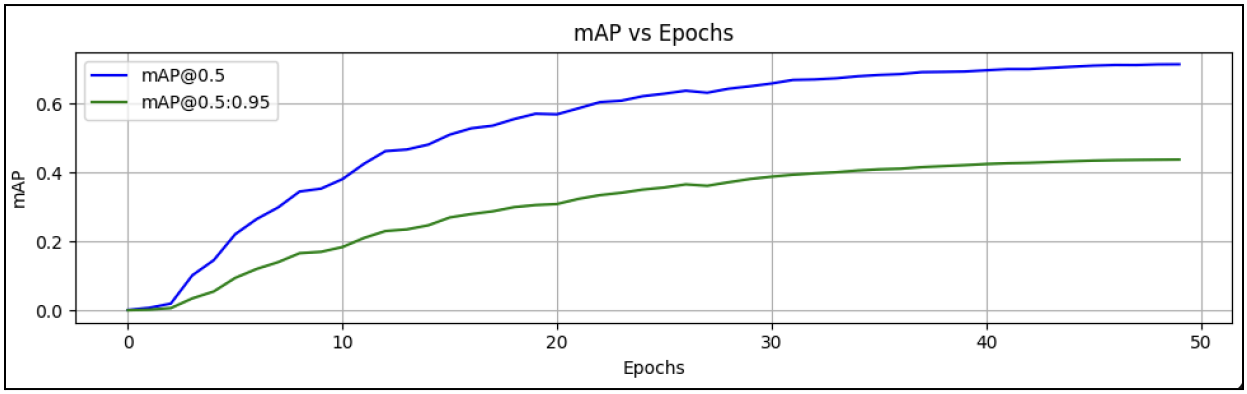
Final Results

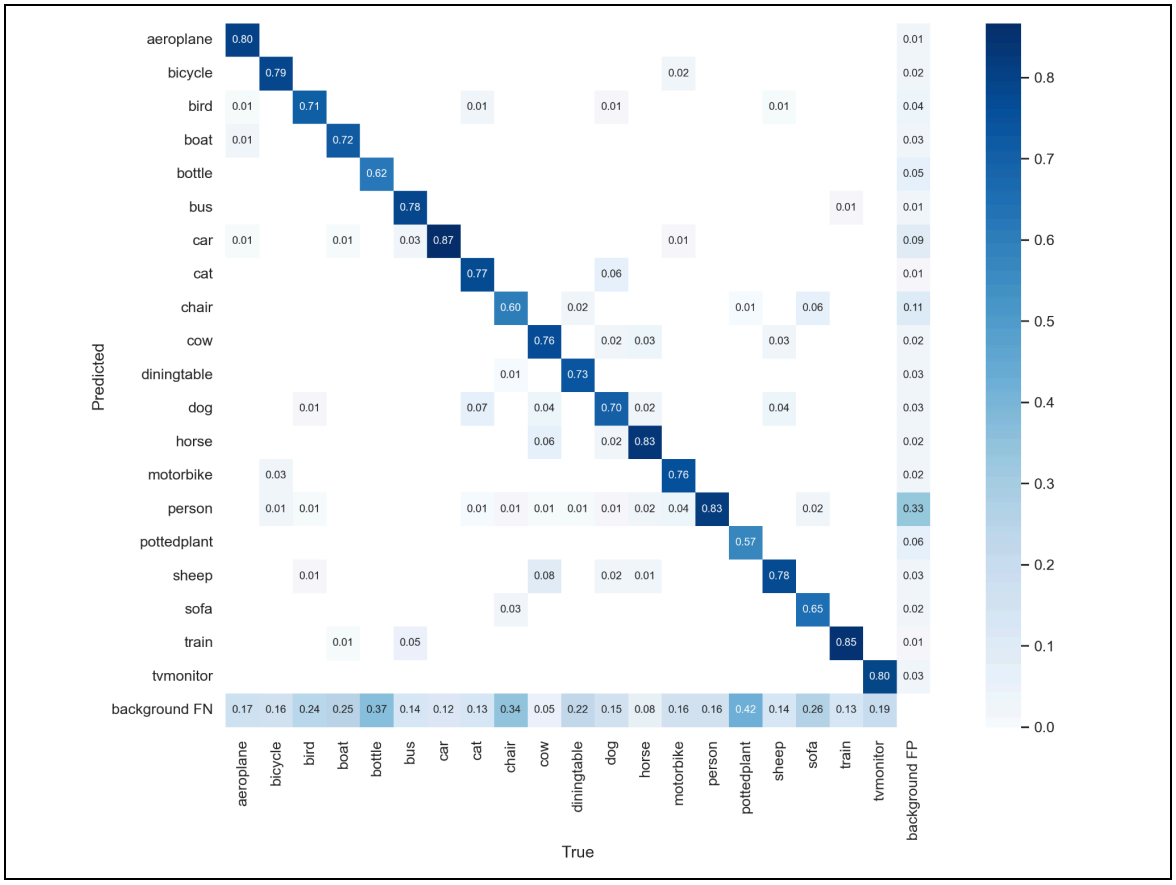
Lauren Spee

Final Training Results:

The only thing I changed from the Preliminary Results was going from training on 25 to 50 epochs. My initial thought was to train the model on 100 epochs, but looking at the metrics afterwards I quickly realized it was overfitting a lot, so I went back down to 50. While 100 was overfitting, 50 seems to be good - it has better metrics than the 25, as seen below.



Throughout the model's training, it saves metrics in a csv, which I used to plot its mean average precision value throughout its training process. mAP measures the average detection accuracy across all the classes, taking into account both recall and precision, meaning it assesses its ability in detecting objects, and if the object is classified correctly. Thus this graph looks exactly as expected: as the model trains and learns, its mAP increases. Though to avoid grossly overfitting the model, I stopped training it as the improvement started to plateau. For models trained on the Pascal VOC dataset, a mAP of 70% or higher tends to be a "good" score, so given the scope of this project, I'm pretty satisfied that at least my mAP@0.5 value made it past 70%. Arguably the mAP@0.5:0.95 is a better overall evaluation due to its more rigorous criteria of requiring predicted and true bounding boxes to have a higher overlap, but for my purposes, I don't deem a very high mAP@0.5:0.95 score critically necessary.



After training the model on the training data, I ran it on the validation set and got this confusion matrix as a result. The diagonal cells represent the frequency that the correct prediction was made, so the level of saturation the line has is a good sign that the objects the model is seeing are being correctly classified, and at a pretty high rate too. This paired with the rather empty other cells means that the model is good at classifying objects it has detected, which is good news.

Though it's worth mentioning that the "background FN" row and columns are fairly saturated as well, meaning the model does miss an, albeit small, still significant amount of objects. This is unfortunate considering this model will be used for a security camera where it's arguably better to have false positives than false negatives. Therefore, if this product ever goes into real development, this will definitely have to be fixed.

Given the time and resources I had at my hands, I'm pleased with the state of my model and weights, and believe it will serve its purpose in my product, at least for a prototype.

However, there is room for improvement. Given enough time and resources, I would like to train this model on a larger, more varied dataset, like COCO. This would allow it to train longer, but on more data to avoid overfitting, and to generally train more thus assumedly get better metrics. Presently, I could train it on a portion of the COCO dataset, but it would be on a similar small amount of data so it wouldn't help

significantly. It might end up actually being worse, given the larger amount of labels that COCO has, thus it would have a proportionally lesser amount of images to then train the model on.
Conversely, training it with pre-set weights would give the model a jumpstart in the right direction, but that seemed against the spirit of this project, so I trained it all from scratch.

Final demonstration proposal:
The functional portion of my code is complete, so all that's left is making a web-app to display my product. I'm strongly leaning towards Flask for this for ease of integration since the backend is in python. My current plan is to create a web-app that allows the user to upload a video (of assumedly security camera footage), and it plays the video, displaying its bounding boxes as it goes, and saving the frames (to be displayed smaller below the video) to show which ones would be sent to the user in notification in the real version of this product. This is the idea, though in practice it might fall into place a little differently due to time and feature constraints. The web-app will be hosted on Heroku, since it seems lightweight and a good site to know how to use.