APS360 Project Report:
Real-Time Face Mask Detector

Prepared by

Team 16
Jiachen (Jason) Zhou 1003300545
Yuxuan (Sherry) Chen 1002942587
Zhiwei (Brian) Liu 1003493007
Salar Hosseini 1003142020

Prepared for

Prof. Sinisa Colic and Teaching Team
Dec. 8, 2020

Word Count: 2461

Introduction

The World Health Organization (WHO) dashboard shows that COVID-19 has globally infected over 60 million people and caused 1.5 million deaths, and is continually getting worse [1]. The most effective way to combat further spread of viruses is to urge people to wear face masks in indoor public spaces [2]. Enforcing this health regulation brings disruptions to work flow for public service providers. Moreover, manual inspection is neither efficient nor effective and comes with a cost of additional human resources. To assist with the enforcement, we propose a face mask detection pipeline that is to be deployed on surveillance systems in public spaces (e.g. airports, shopping malls, etc.) to alert inspectors of any individuals' violation of the regulation. Designed for both accuracy and speed, our detector uses machine learning to localize human faces from a live video feed with bounding boxes and classify whether they are wearing face masks or not.

As Convolutional Neural Networks (CNNs) are commonly used to tackle object detection problems, we adopted a deep learning architecture with a CNN backbone to perform facemask detection. After comparing multiple existing object detection frameworks, we chose Faster R-CNN [3] for its unequalled detection accuracy. The overall pipeline is shown in Figure 1.

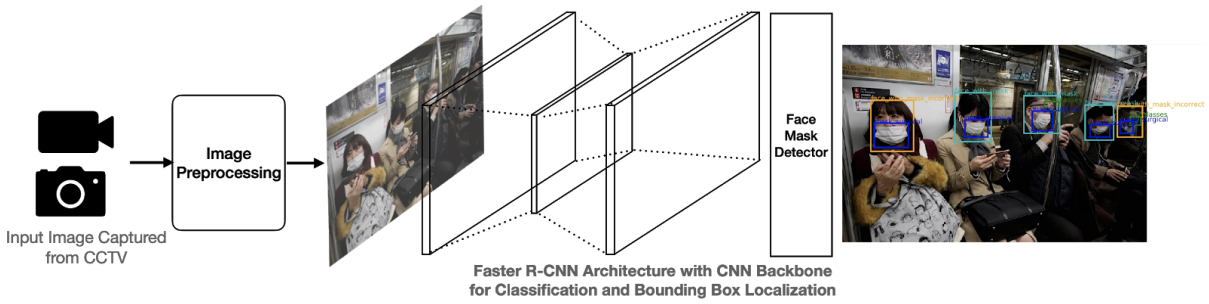


Figure 1. Illustration of the overall pipeline, consisting of image capture from a video stream, data preprocessing, and a Faster R-CNN [3] architecture based detector for face masks.

Background

From the literature, Faster R-CNN is a two-stage detector which achieves high accuracy at a processing speed of ~5 frames/second [3]. It consists of an initial Region Proposal Network stage which generates possible object regions from a feature map, and a second stage called a Fast R-CNN detector [4] which simultaneously predicts bounding boxes and object labels, as shown in Figure 2 [3].

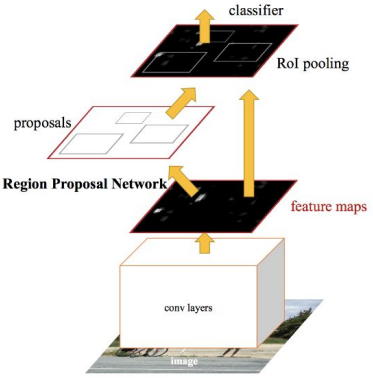


Figure 2: The Faster R-CNN object detection model [3].

Another popular model is YOLOv3 [5], a one-stage object detector that has lower accuracies than Faster R-CNN but faster inference at ~30fps [5]. YOLOv3 model divides the image into a 2D grid of cells pertaining to possible objects, reduces the spatial dimension to a desired grid size, and outputs features for a regression layer to predict bounding boxes and class labels, as shown in Figure 3 [6]. However, YOLOv3 has limitations on the vicinity of objects due to the grid size restriction.

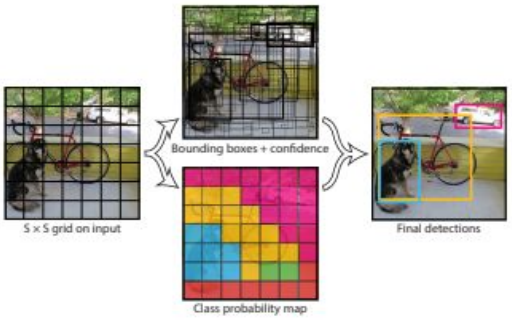


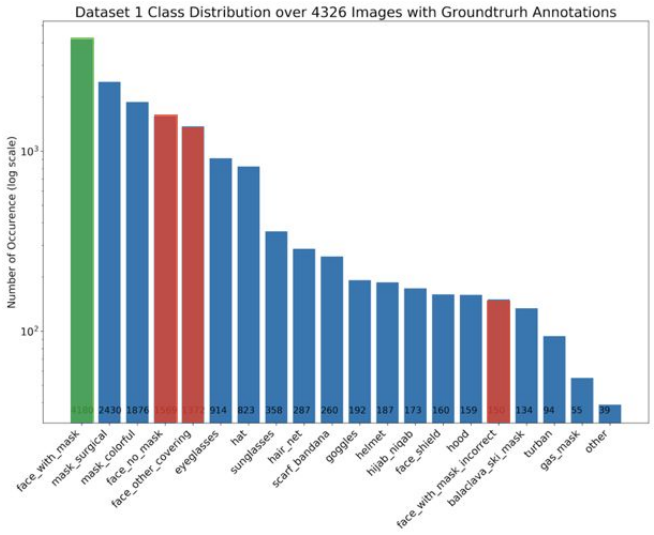
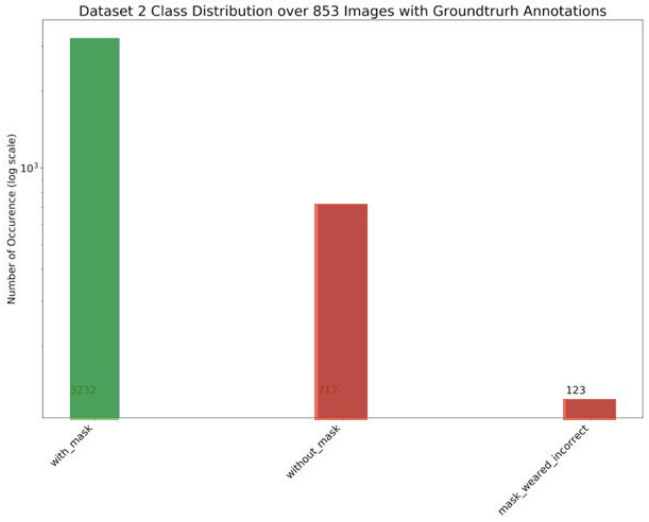
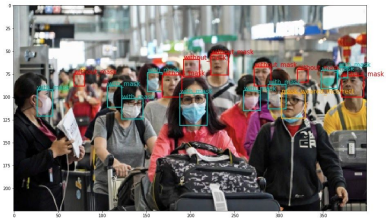
Figure 3: The YOLO object detection model [6].

Data Processing

The two datasets used are from Kaggle [7][8], as seen in the table below. The first dataset consists of 4326 annotated images of people wearing various masks, while the second dataset has 853. The bounding boxes for faces are given for both the datasets, indicating whether a face is face_with_mask, face_with_mask_incorrect, or face_without_mask. For dataset 1, there is an additional label of face_other_covering, which we grouped together with face_without_mask. Since we focus on detecting whether a mask is worn properly, we also grouped face_with_mask_incorrect into face_without_mask, effectively making it detection with a binary classification of more balanced classes (face_with_mask vs. face_without_mask). Dataset 1 also includes labels for mask types and other items, such as mask_surgical, ski_masks, eyeglasses, etc., which were safely removed.

We used 70% of dataset 1 for training and 15% for validation. For testing, we held out the remaining 15% of dataset 1 and all of dataset 2, for testing transferability to unseen data.

Table 1. Datasets Statistics and Examples of Images and Annotations

	Class Distribution	**Data Example**
Dataset 1	 Positive samples: face_with_mask Negative samples: face_no_mask, face_other_covering, face_with_mask_incorrect (for training, validation, and testing)	 Cyan: face_with_mask Orange: face_with_mask_incorrect Red: face_without_mask ~~Blue: mask_surgical~~ (ignored)
Dataset 2	 Positive samples: with_mask Negative samples: without_mask, mask_worn_incorrect (only for testing)	 Cyan: with_mask Orange: mask_worn_incorrect Red: without_mask

Another useful pre-processing step was data augmentation for increasing robustness to scale and orientation changes of objects. Following the Faster R-CNN [3] approach, we randomly resized input images by their shortest edge to one of {640, 672, 704, 736, 768, 800} pixels, and flipped the images horizontally with a 50% probability, as demonstrated in Figure 4.

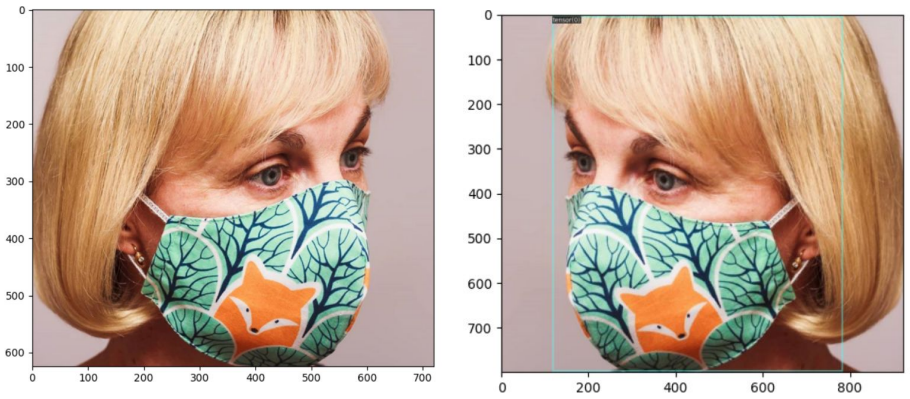


Figure 4: Raw training image (left) and after a resize to 800px with horizontal flip (right)

Architecture

The final architecture, depicted in Figure 5, is a variation of the Faster R-CNN from [3] which uses a Feature Pyramid Network (FPN) [9]. YOLOv3 [5] was not considered for its poor performance for objects in close vicinity, and another one-stage detector, RetinaNet [10] was experimented with, but yielded lower accuracies and speed.

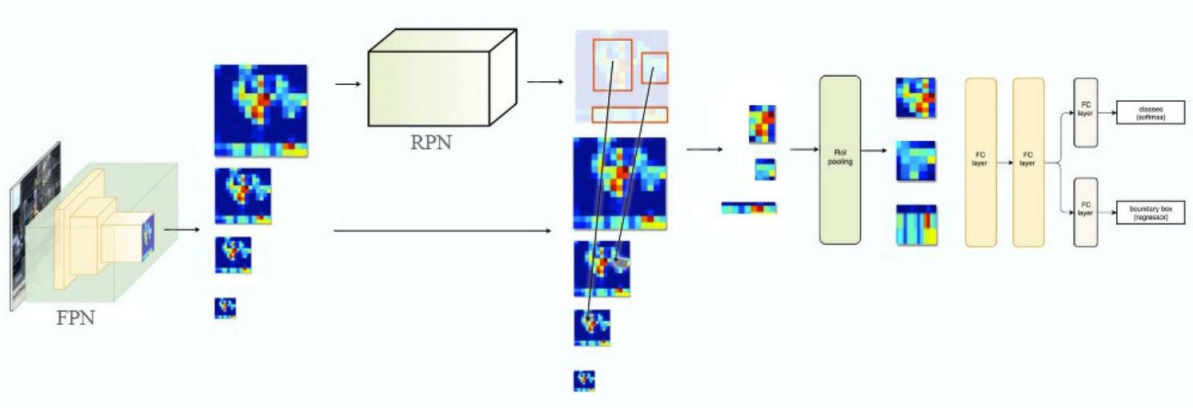


Figure 5: Faster R-CNN with a FPN backbone [11]

The first stage of the model is the FPN, which has the generic structure shown in Figure 6. It is composed of a feedforward ResNet-50 encoder [12] (bottom-up pathway) followed by 4 feature maps of increasing scale (top-down pathway) which are produced by upsampling the previous output by 2x and performing 3x3 convolutions with stride and padding equal to 1. In addition, there is a skip connection going to each feature map in the top-down pathway which performs element-wise addition.

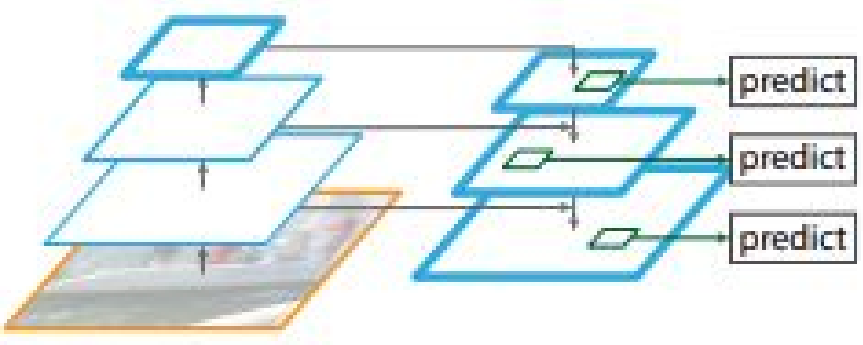


Figure 6: A Feature pyramid network composed of a bottom-up and top-down pathway [9]

The FPN results in 5 feature maps of increasing scale, which are used to generate region proposals of sizes $\{32^2, 64^2, 128^2, 256^2, 512^2\}$ respectively. For each feature map, a Region Proposal Network (RPN) slides a single-stride 3x3 kernel to generate anchor boxes with the aforementioned areas at aspect ratios of (1:1, 1:2, 2:1) as shown in Figure 7. For each anchor, the RPN predicts the possibility of it being the background or foreground, as well as the possible offsets.

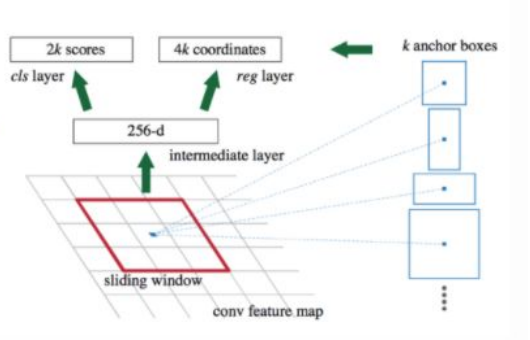


Figure 7: The sliding window used in the Region Proposal Network [3].

Then, as depicted in Figure 8, the proposals are sent to a ROI Pooling layer that applies Max-Pooling on every region to reduce the feature maps to the same size (7x7), and then sends the channels to a classifier and regressor to detect the occurrence of objects.

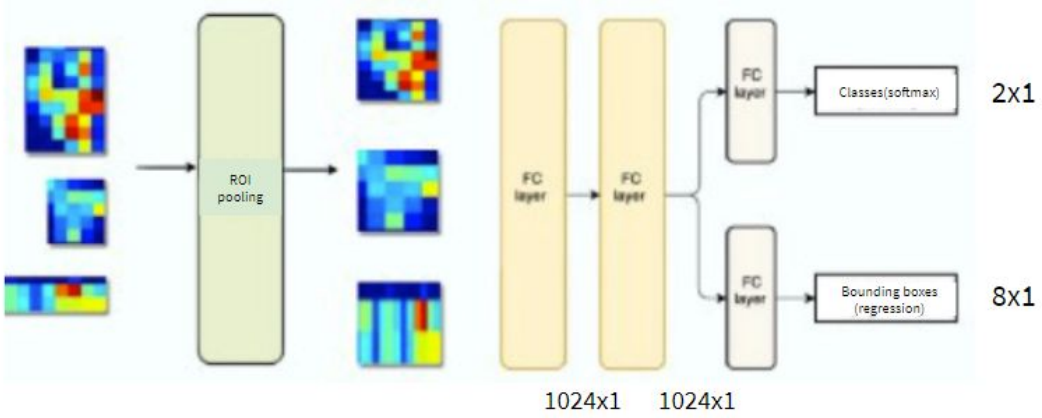


Figure 8: Region of Interest (ROI) pooling on region proposals from the RPN, followed by fully connected layers leading to separate class and bounding box predictions [11].

We initialized a Detectron2 implementation of the model [13] with pretrained weights on MS-COCO [14], and fine tuned the network end-to-end on the face mask dataset.

Baseline Model

To construct a baseline model, we replaced the FPN backbone with a simple 2-layer CNN, as shown in Figure 9. This replaces the 5 feature maps of varying sizes generated by the FPN with a feature map of fixed size generated by the CNN. Since each feature map corresponds to one anchor size, the baseline model only has one fixed anchor size. The rest of the pipeline is the same as our model with the exception that only a single feature map is used by the RPN to generate region proposals used for classification. The modified pipeline is shown in Figure 10, where one can see that the baseline model is essentially our model with key features of the FPN backbone removed. Hence, our baseline model is a suitable candidate for an ablation study. The Detectron2 configuration is shown in Figure 11.

```
@BACKBONE_REGISTRY.register()
class BaselineCNN1(Backbone):
  def __init__(self, cfg, input_shape):
    super().__init__()
    # Baseline model
    self.conv1 = nn.Conv2d(3, 32, 5)
    self.pool = nn.MaxPool2d(2, 2)
    self.conv2 = nn.Conv2d(32, 64, 5)

  def forward(self, img):
    x = self.pool(F.relu(self.conv1(img)))
    x = F.relu(self.conv2(x))
    return {"p2": x}

  def output_shape(self):
    return {"p2": ShapeSpec(channels=64, stride=1)}
```

Figure 9. CNN backbone implementation in Detectron2

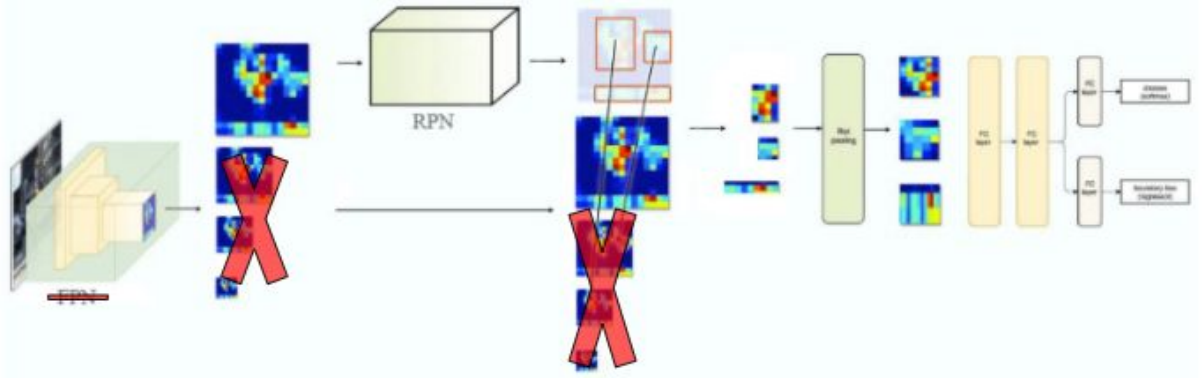


Figure 10. Modified pipeline featuring a single feature map with fixed anchor size replacing the feature maps generated by the FPN [11]

```
# Load base config file
cfg = get_cfg()
config_file_path = "Base-RCNN-FPN.yaml"
cfg.merge_from_file(model_zoo.get_config_file(config_file_path))

# Replace FPN backbone with CNN backbone
cfg.MODEL.BACKBONE.NAME = 'BaselineCNN1'
cfg.MODEL.RPN.IN_FEATURES = ["p2"]
cfg.MODEL.ANCHOR_GENERATOR.SIZES = [[256]]
cfg.MODEL.ROI_HEADS.IN_FEATURES = ["p2"]
```

Figure 11. Replacing FPN backbone with CNN backbone in Detectron2.

Quantitative Results

During training, the model was trained with the SGD optimizer and evaluated using the cross entropy loss for class prediction and the smooth L1 loss [15] for bounding box regression. The resulting combined (added) losses from the training and validation splits of dataset 1 are shown in Figure 12a and 12b. Figure 12c shows the plot legend with the hyperparameters that were tuned, where {'b', 'lr', 'proposal', 'iter'} represent {batch size, learning rate, number of proposals sampled from the RPN, number of training iterations} respectively.

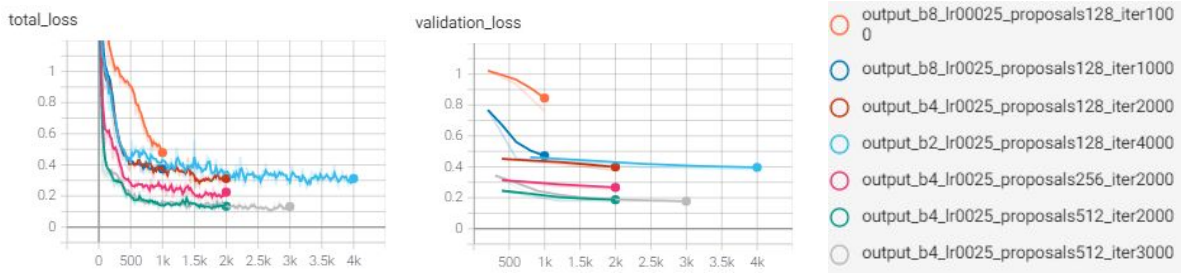


Figure 12: (a) Training loss (b) Validation loss (c) Legend with hyperparameter settings

In the same order, the best parameters were found to be {4, 0.0025, 512, 3000}, with the resulting training curves shown in Figure 13.

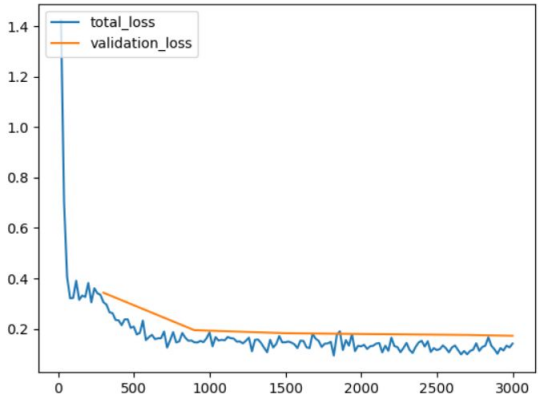


Figure 13. Training and Validation Curve for Proposed Model

Our holdout data consists of the test split from dataset 1 and all of dataset 2. We evaluated our trained model on this data with two metrics: mean average precision (mAP) [16] and inference time. The final results for mAP are calculated using the equations shown below and are reported in Table 2. The precision-recall curve at 0.5 Intersection over Union (IOU) is shown in Figure 14. Lastly, the confusion matrix generated by the test set prediction is shown in Figure 15 and indicates that the model predicted a high number of true positives and negatives and very few false positives and negatives.

$$precision = TP / (TP + FP) \quad (1)$$

$$recall = TP / (TP + FN) \quad (2)$$

$$AP = 1/11 \sum_{recall_i} precision(recall_i) \quad (3)$$

$$mAP = 1/2 \, (AP_{with\ mask} + AP_{no\ mask}) \quad (4)$$

Table 2: mAP@IOU{0.5, 0.75}

	mAP@0.5 (%)	**mAP@0.75 (%)**	**Inference Time (s/frame)**
Faster- RCNN	84.55	71.54	0.07

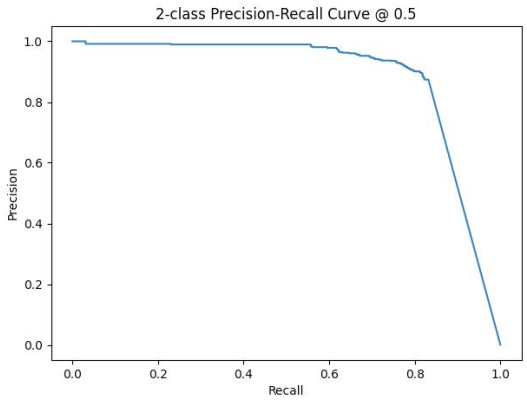


Figure 14. Precision-Recall Curve

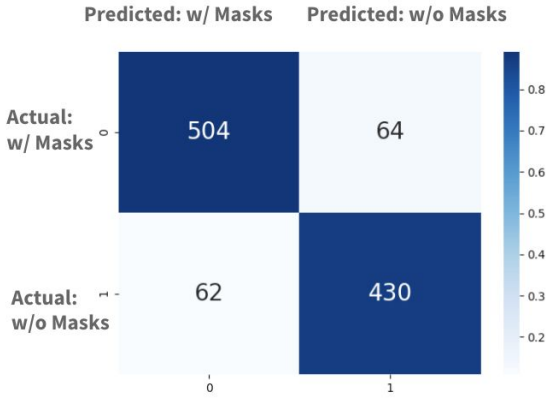


Figure 15. Confusion Matrix on Test Set

Qualitative Results

We randomly selected several images from the test set to visualize our model's performance. In most of the cases, our model is quite robust as it is scale and orientation invariant. For the images in Figure 16, it is shown that the model successfully localized all faces present, including those out-of-focused human faces in the background.

Figure 16. Test Set Visualization on Model Prediction

However, we acknowledge that there are certain limitations of our model. For instance, in Figure 17, our model incorrectly classifies other types of face coverings (e.g. bandana, scarf, etc.) as face masks. This might be caused by the low number of hard negatives in the dataset (i.e. images including face coverings other than masks), and can be addressed by adding more occurrences of such negative samples representing different types of face coverings in the training.

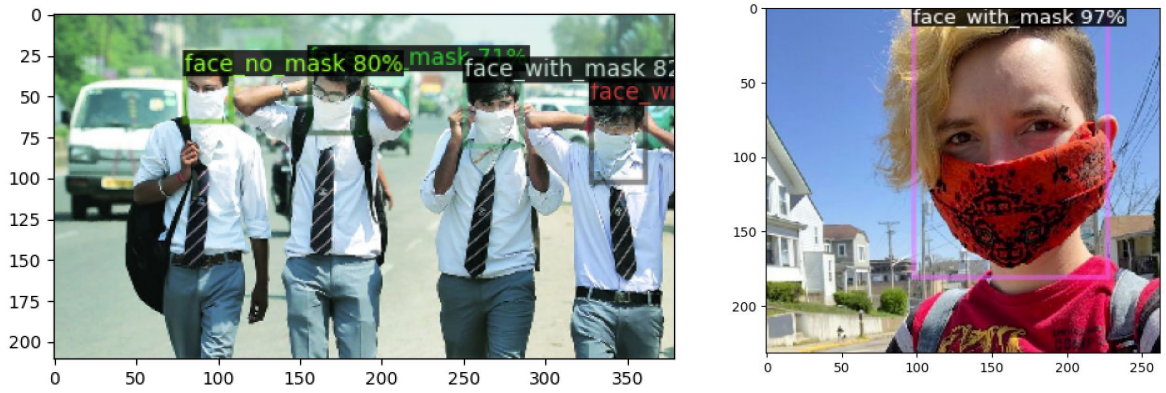


Figure 17. Visualization on Failed Cases

Model Evaluation on New Data

To evaluate our model on additional unseen data, we evaluated its performance on a live video feed. The video feed was captured by screen recording a live Zoom team meeting, as seen in Figure 18. This ensures that the data being fed to our model for inference is unseen. Since our model should have the capacity to detect multiple faces, we used a composite of our webcams to make the problem more difficult. In each frame, our model should detect four faces and provide label predictions and bounding boxes for each face. For visualization, we saved the frames labelled by our model to a video for playback.

Figure 18. Video feed sample with class and bounding boxes predictions from model inference; green bounding boxes indicate the presence of masks.

From the video, we observed that the model was able to consistently provide accurate predictions and bounding boxes with high confidences of around 90% - 99%. In addition, the average inference speed per frame was 70ms, which means our model can process ~14 frames per second (fps). Interestingly, despite the presence of sharp transitions between our webcams, our model can still accurately identify our faces and perform robustly in the presence of unnatural noise. Moreover, when taking off our masks, our model can quickly change its prediction when it sees an exposed nose and mouth, even though the mask is not completely off, as shown in Figure 19. This is desirable as it can allow users to quickly respond to illegal mask usage.

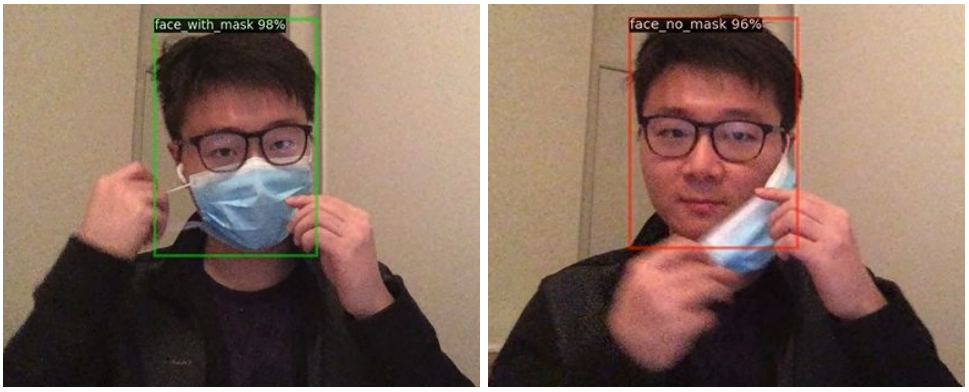


Figure 19. Change of prediction when taking off the masks

However, our model occasionally struggles when a mask is partially worn, as seen in Figure 20. In this example, our model predicted two overlapping faces with contradicting predictions. The red bounding box correctly predicts a face with incorrectly worn masks, while the green bounding box incorrectly predicts an additional "face" with a mask on. One idea to mitigate this is to take the prediction with the highest confidence when we have instances that have high overlap. However, we need to be careful of overlapping faces if we take this approach.

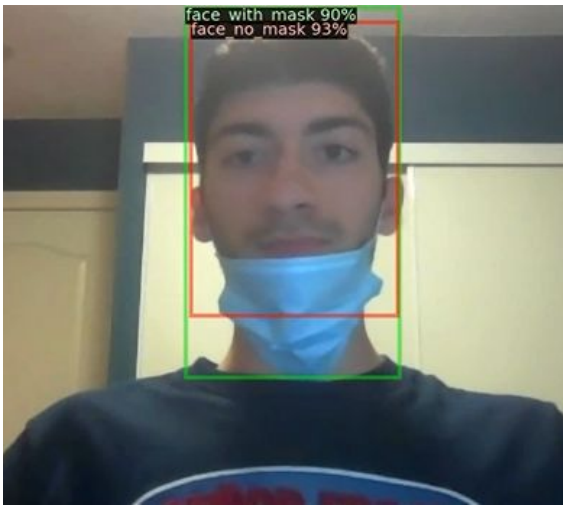


Figure 20: Partially worn masks caused the model to predict two overlapping faces with conflicting predictions, each with high confidence

Discussion

In general, our model performed well compared to the baseline model. By inspecting the predictions made by the baseline (Figure 21), we can easily see that the bounding boxes do not encompass faces accurately and the class prediction accuracies are around 20-30%, while our final model generated much more accurate bounding box and class predictions. The poor performance of the baseline was likely due to the fixed anchor size used, which caused improper regions of interest to be generated. Furthermore, we observed that without multiple

feature layers of varying scales, such as those generated by the FPN network, our model's performance would be greatly compromised.

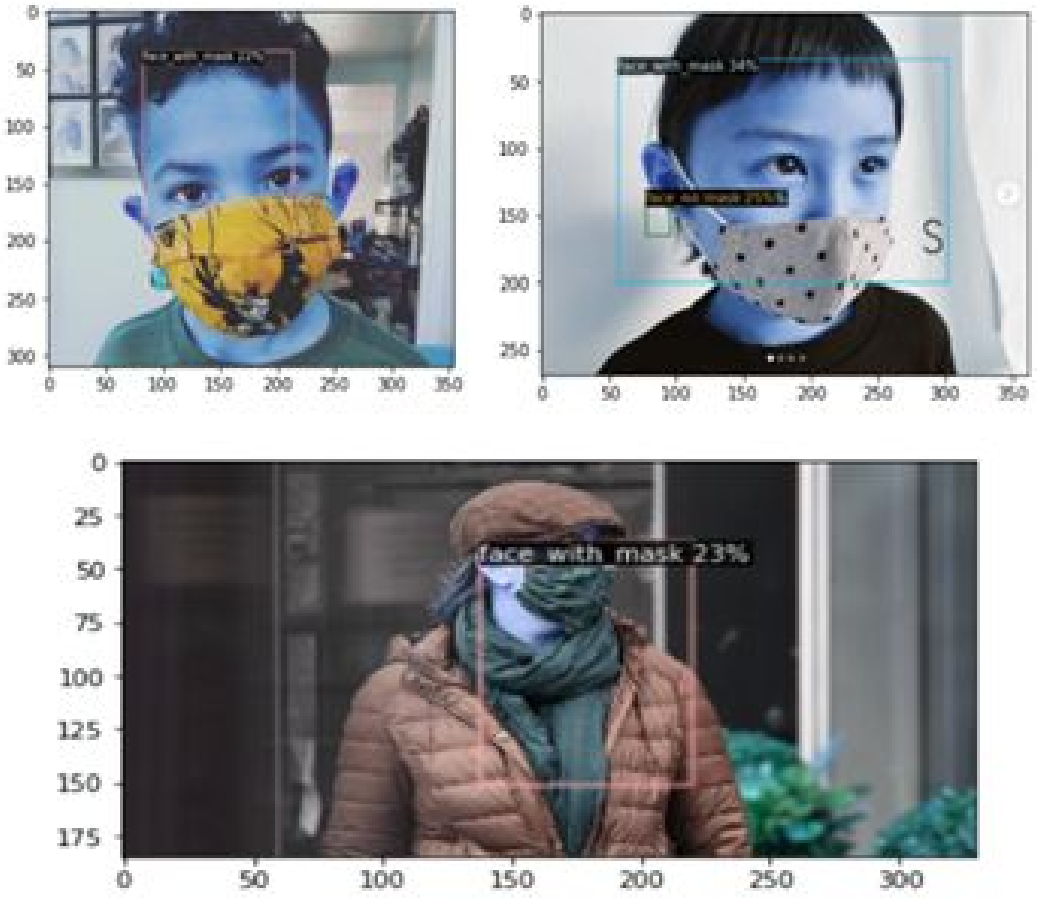


Figure 21: Visualization of Baseline Model Prediction

In addition to the network's architecture, we learned that initializing our model with the pretrained weights from the COCO dataset [14] improved the learning efficiency and robustness of our model. Since the facemask dataset was relatively small, it would achieve poor accuracy if our model was trained on the dataset from scratch. However, with pretrained weights loaded, our model could reuse the low-level features learned from COCO [14] and generalize from the object detection task to the face mask detection task. Leveraging the learned representations, our model was able to exceed our expectations on the dataset and live video. For instance, when visualizing our model's prediction results, we were surprised to find that our model was able to accurately detect blurry human faces in the backgrounds that were not even annotated with ground truth labels. In the video demo, we observed that our model could accurately detect our faces from image composites and performed robustly in the presence of noise.

Lastly, the model architecture permits high inference speeds without compromising accuracy. In our demo, our model achieved 14fps on a live video feed, which is desirable for industry since ~15fps maintains sufficient video quality while minimizing storage costs [17]. Thus,

our model should be readily deployable on existing surveillance cameras for real-time inference.

Ethical Consideration

Our model requires access to surveillance for deployment. This is less of a concern in areas such as airports where surveillance already exists, but it is unrealistic that one can have widespread camera coverage in some general public space. Further, one may need to incorporate additional information, such as geolocation or face recognition, to identify and track individuals identified as a risk by our system. The need for surveillance and personal information pose privacy issues. For instance, most data generated or collected by AI surveillance will not be covered under HIPAA[1] [18]. Personal information, such as cell phone geolocation, that reveal crucial information about COVID19 cannot be used. Even if policies allowed exceptions for the purpose of healthcare, such privacy violations are unlikely to be remedied in the aftermath of the pandemic [18]. It will also be important to make our system robust to differences in skin colour and the presence of traditional attire. To achieve this, we would need to ensure that our training data is diverse enough so that our model can learn to be insensitive to these differences.

Challenges Faced

One significant challenge was detecting multiple occurrences of human faces in the same image with various scales and poses and outputting accurate bounding boxes in addition to correct labels for each. This makes the problem more difficult than simple classification. Our model performed quite well in this aspect, mainly as a result of the Faster-RCNN architecture and its ability to produce ROIs that were close-by or overlapping. The resulting model beats the baseline by a large margin and it has shown to be robust both quantitatively and qualitatively. Detecting faces of varying scales and orientations was overcome through the multi-scale features computed from the feature pyramid network and the augmentation techniques that were employed. However, there is still room for improvement, as the detector occasionally misclassified certain types of face coverings as face masks. These issues will need to be addressed since a high false positive rate could lead to undetected public risks.

Furthermore, since there is usually a tradeoff between performance and speed, it is quite challenging to achieve real-time inference (>20fps) without compromising the detection accuracy. Currently, our best model achieves an inference speed of 14fps using 1 GPU, which would be sufficient to be deployed in most surveillance cameras to perform live predictions smoothly. To further close the gap, we will explore more light-weight model variations in order to achieve better real-time inference. Overall, the promising results of our model makes it on track for real-world deployment.

[1] HIPAA: U.S. Health Insurance Portability and Accountability Act of 1996 (HIPAA) Privacy Rule offers protection for certain individually identifiable health information

References

[1] World Health Organization, “WHO Coronavirus Disease (COVID-19) Dashboard,”. [Online]. Available: https://covid19.who.int/?gclid=CjwKCAiA_Kz-BRAJEiwAhJNY7wmse49jTEM2LWRhXe--sCPtSDIBU5x0oKnOOPrKvWezx3s7IJnNvhoC0j4QAvD_BwE [Accessed: 6-Dec-2020]

[2] City of Toronto Council, “By-law 541-2020 To impose temporary regulations requiring the wearing of masks or other face coverings within enclosed public spaces.” Toronto, ON (2020).

[3] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[4] R. Girshick, “Fast R-CNN,” in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[5] J. Redmon, A. Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[7] Kaggle.com. 2020. Face Mask Detection Dataset. [Online] Available: https://www.kaggle.com/wobotintelligence/face-mask-detection-dataset [Accessed: 18-Oct-2020].

[8] Kaggle.com. 2020. Face Mask Detection. [Online] Available: https://www.kaggle.com/andrewmvd/face-mask-detection [Accessed: 18-Oct-2020].

[9] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and ´ S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[10] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal Loss for Dense Object Detection,” *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[11] . Hui, “Understanding Feature Pyramid Networks for Object Detection (FPN),”. [Online]. Available: https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c [Accessed: 5-Dec-2020].

[12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” arXiv:1512.03385, 2015.

[13] Y. Wu, A. Kirillov et al., “Detectron2,” 2019. [Online]. Available: https://github.com/facebookresearch/detectron2 [Accessed: Nov. 14, 2020]

[14] T.-Y. Lin, M. Maire, et al., “Microsoft coco: Common objects in context,” In European conference on computer vision, pages 740–755. Springer, 2014.

[15] "SmoothL1Loss — PyTorch 1.7.0 documentation", *Pytorch.org*, 2019. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html [Accessed: 07- Dec- 2020].

[16] T. C. Arlen, “Understanding the mAP Evaluation Metric for Object Detection,” *Medium*, 01-Mar-2018. [Online]. Available: https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3. [Accessed: 08-Dec-2020].

[17] J. Honovich, "Frame Rate Guide for Video Surveillance", *IPVM*, 2014. [Online]. Available: https://ipvm.com/reports/frame-rate-surveillance-guide. [Accessed: 07- Dec- 2020].

[18] C. Shachar, S. Gerke and E. Adashi, "AI Surveillance during Pandemics: Ethical Implementation Imperatives", *Wiley Online Library*, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/hast.1125. [Accessed: 07- Dec- 2020].