Final Project Progress Report

Definitions

Team name: ***Cover Your Nose***

Team members: ***Samuel Wilkins, Shiqin Yan, Xiaoyan Zhao, Zhehui Liang***
Note: Once one person uploads the report to Gradescope, please add all other team members to the submission within the Gradescope interface (top right on your submission).

TA name: ***Carlos Perez-Ruiz***

1 Main goal of our project

Our project aims for judging whether a mask is being wore properly. Thus the process is roughly described as follows:

1. Take an image or a video as input.
2. Construct a facial detector, and return several small boxes with human faces and confidential probabilities.
3. Now construct a mask detector. (Or, merge steps 2 and 3 to create an object detection model, maybe with a transfer-learned backbone of ResNet-50, and a custom-trained extension that localizes faces and acts as a binary classifier for each bounding box, mask vs. no-mask.)
   - If there is no mask detected, then return ”**Please wear a mask**” .
   - If we detect a mask, then we need to justify whether the mask is worn properly. Thus we need to construct a nose detector and (possibly also a mouth detector).
     - If no noses or mouth are detected, then return ”**you are wearing a mask properly.**”
     - If we detect a nose or mouth in the face box, then return ”**Though you are wearing a mask, please cover your nose (mouth)**”

2 Our dataset

In our previous proposal, we have listed several dataset for mask detection, which we also listed here:

(a) Face Mask Detection dataset1 on Kaggle

(b) Face Mask Detection dataset2 on Kaggle

(c) Face Mask dataset on roboflow

(d) Face Mask Detection video dataset

But there are problems here. Most of those images don't display noses or mouths since they are hidden by masks, thus we need another dataset for face recognition, which is listed here:

- **Wider Face dataset**. The dataset is from this weblink, which contains 32,203 images with 393,703 normal faces with various illumination, pose, occlusion etc.
- **MAFA dataset**. This dataset is from this weblink, which contains 30,811 images and 34,806 masked faces, but some faces are masked by various kinds of things rather than physical masks. Here I list several images from this dataset that are pretty tricky, as the noses and mouth are hided by hands, veils, scarfs, etc.

(a) Nose and mouth covered by hands

(b) Nose and mouth covered by veils

(a) Whole face covered by masks

(b) Part of the Nose and mouth covered by a book

Besides, we might also use the COCO dataset as listed in the SSD model (subsection 5.1.3), which is listed here

3 Do you actually have the compute

Yes. Currently, we have tested our code on single images locally, but we will run our model on our GCP instances, using existing credits, after we have built up the training and testing (validation) dataset.

4 Is there any code that you don't have access to

So far we don't have this type of worries.

5 Is there any area that might need help?

At the moment, we're trying to decide on the best object detection model (either Faster-RCNN, SSD, the lighter weight AIZOO model). I think at the moment, we're very much still developing our understanding of how to train an object detection model where the labels aren't just a single class, but a number of class, bounding box pairs. We found this as a recommended model associated with our first Kaggle dataset (which has labels in the PASCAL VOC format, which is convertible to JSON if need be), and seems to give some reasonable insight into how we might implement a training pipeline with our given data. However, this is still somewhat shaky ground for the team.

6 Some progress that we have made

At a high level, we have currently 'skipped to the end,' if you will, by building out our desired pipeline using pre-trained models. Our next and final substantive step will be to replace the pre-trained mask detector / localizer with our own version.

Specifically, we first use the pre-trained AIZOO mask detector (6.1.2) to retrieve bounding boxes and a mask vs. no mask classification for each face in a given input image. Then, we use each bounding box to crop a facial region of interest (ROI) out of the image, which we feed into HAAR nose and eye detectors. Then, our final prediction depends on a few simple rules. If the classification was "No Mask," we indicate failure (prompting the user to put on his or her mask). Else, i.e. a mask was detected, and there is a validated nose present*, we indicate the intermediate classification "Exposed" (prompting the user to pull up their mask). If no validated nose is detected, we return the proper "Mask" classification (indicating that the user is wearing the mask correctly). *Because the Haar detectors can sometimes be overzealous in identifying noses, if we are able to detect eyes (which seem more stable), we filter out any ’noses’ that are not spatially below the eyes detected in the image. Of course, if the eye detector is unable to locate any eyes, we lose the benefit of this spatial validation. As a result, we have been able to properly classify the 6 images in our initial test set. The bounding boxes on the left are mostly obscured, but they are indeed green, and their blue counterparts on the right indicate that both a mask and a nose was detected:

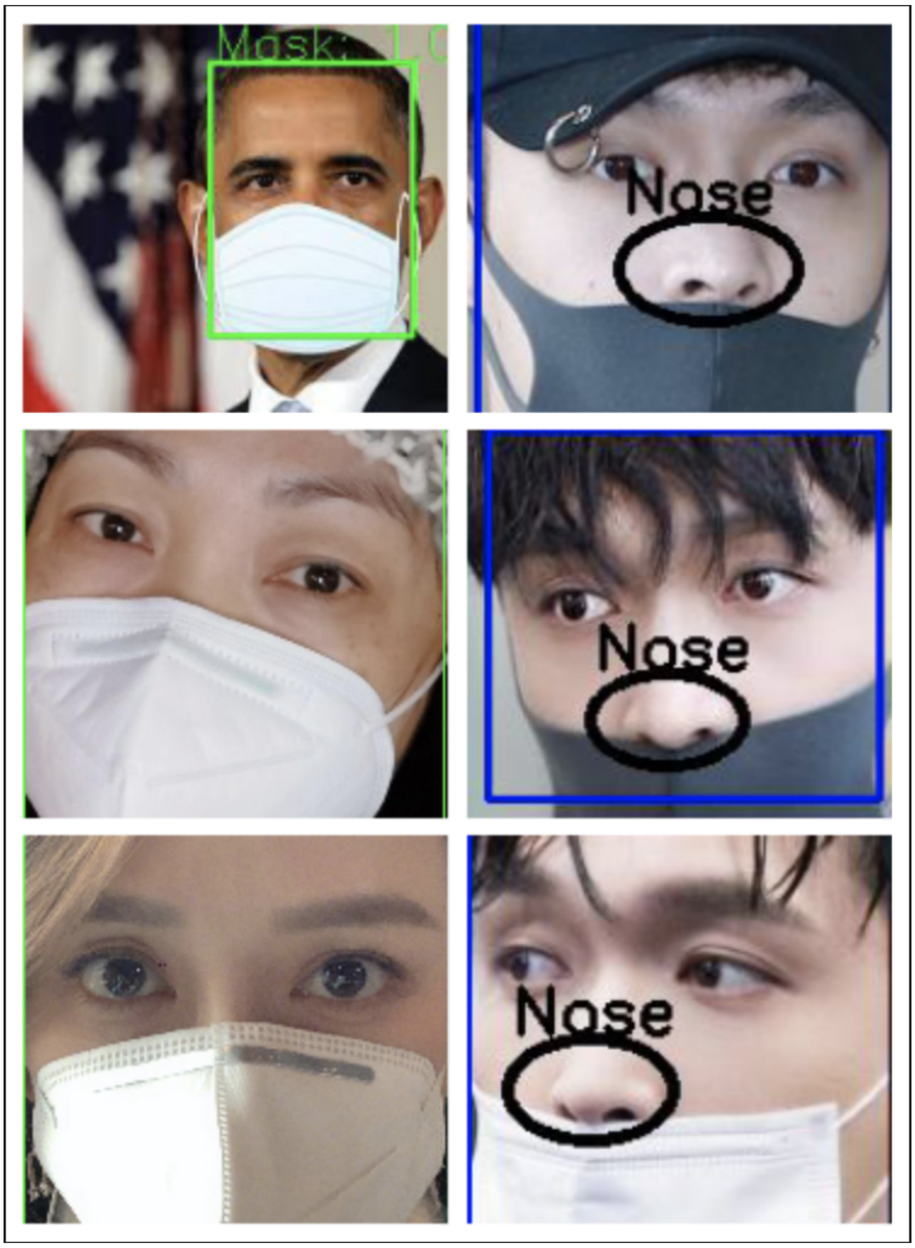


Figure 3: Initial test results (no nose vs. nose)

Again, this pipeline is effective, but currently contains entirely pre-trained models pending our own implementation of a mask (object) detector.

6.1 Some of those existing models that we might use

6.1.1 Haar Cascades feature detection

Haar Cascade classifier is an effective object detection approach which was proposed by Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001. This is basically a machine learning based approach where a cascade function is trained from a lot of images both positive and negative. Based on the training it is then used to detect the objects in the other images.

There are huge individual .xml files with a lot of feature sets and each xml corresponds to a very specific type of use case, which are saved in the OpenCV library. This link provides some useful xml file for feature detection.

6.1.2 AIZOO Mask Detector

This model (with link over here) is a pre-trained CNN model with super small number of parameters (about 1M). This is a weblink that we can play with it online.

The following shows some running results on this AIZOO model. We can see that it is also suitable for multi-faces detection in a single image, but as you can see this model is not stable. Two similar facial boxes might have absolute different mask-wearing result.

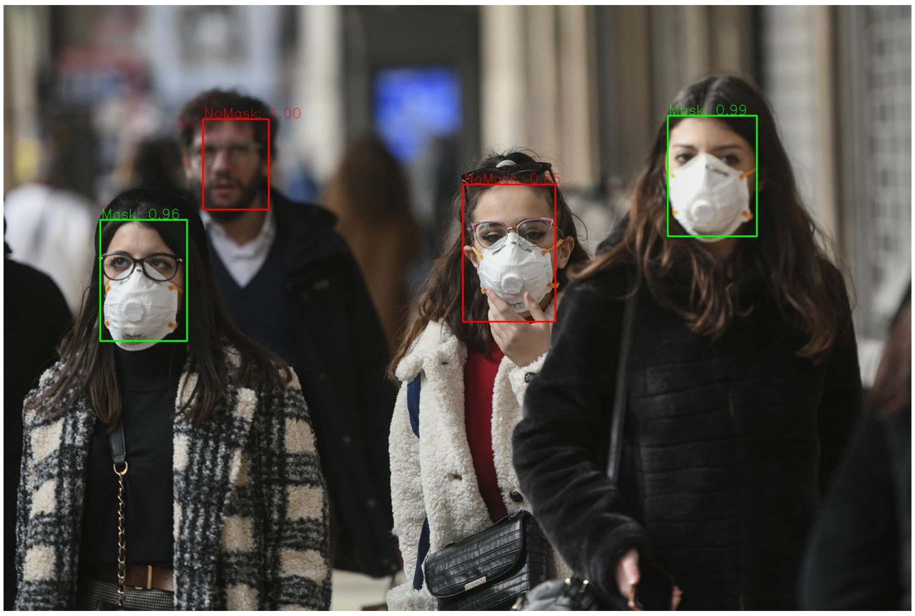
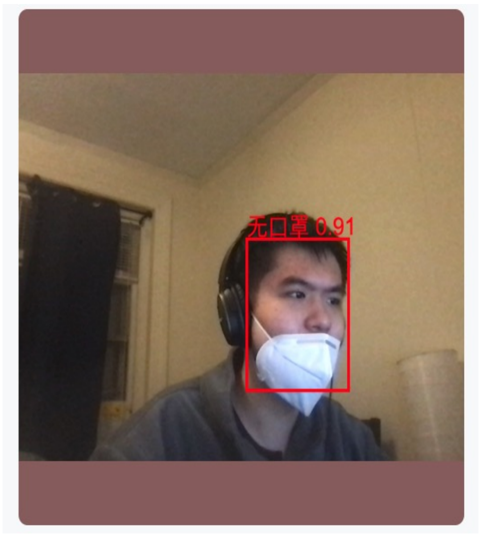
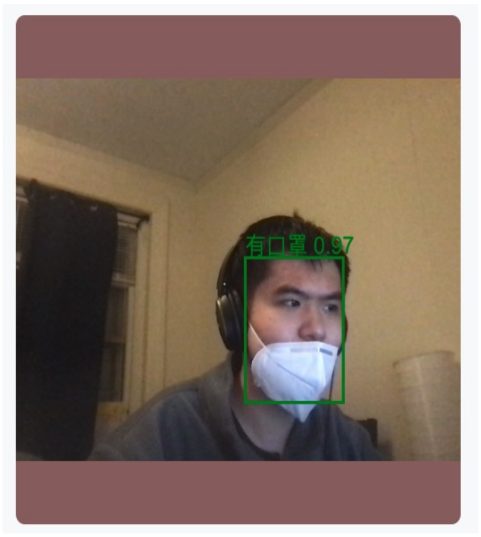


Figure 4: Detecting multi-faces by AIZOO models



(a) Detecting no mask with probability 91%



(b) Detecting mask with probability 97%

However you can also see that this model is not stable enough. For the mult-face image, the woman with white coat is jutified as ”No mask” with probability 85 %. And when we play with the models by ourselves, similar picture might lead to total different mask-wearing results.

Thus we might use another mask-detection model, which is listed as below.

6.1.3 SSD Object Detecting Model

SSD (which is the abbreviation for ”Single Shot Detector”, is a method for detecting objects in images using a single deep neural network. It has heavier weights than the AIZOO model above. The theory is based on this paper and the architecture is listed as below, which is pretrained on the COCO dataset (where there are 91 object categories) from TorchHub.

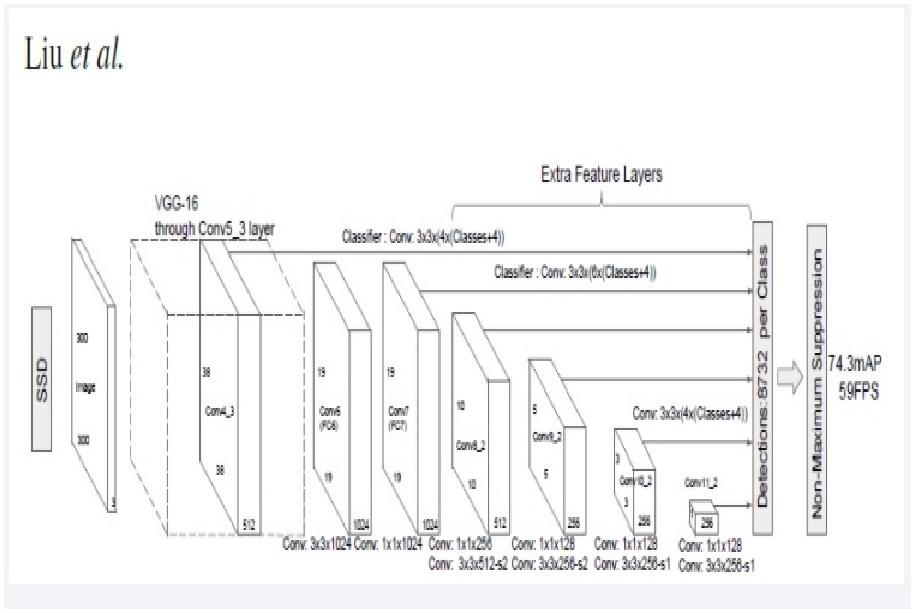
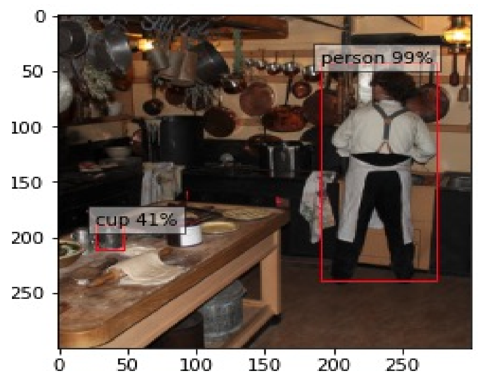


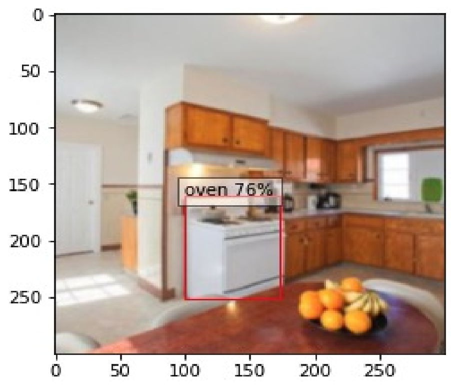
Figure 6: SSD architecture

- The early layers are base network which can be some pretrained network used for image classification (Here we use VGG-16 network)
- Convolutional predictor layers:
  (a) They are added at the end of base network, which decreases in size progressively, providing predictions at different scales.
  (b) For a feature layer of size mxn with p channels, the basic element for predicting parameters of a potential detection is a $3x3xp$ small kernel that produces either a score for a category, or a shape offset relative to the default box coordinates. At each of the m × n locations where the kernel is applied, it produces an output value. The bounding box offset output values are measured relative to a default box position relative to each feature map location.
- The ending layers are non-maximum suppression to reach final detections.

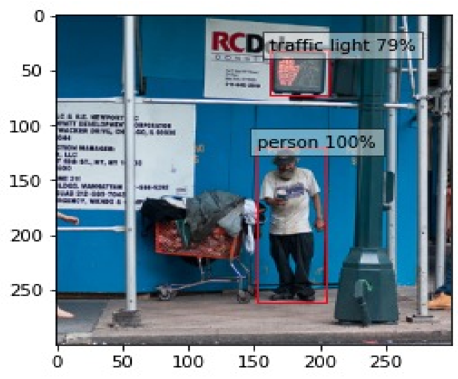
Here are some running results of SSD model.



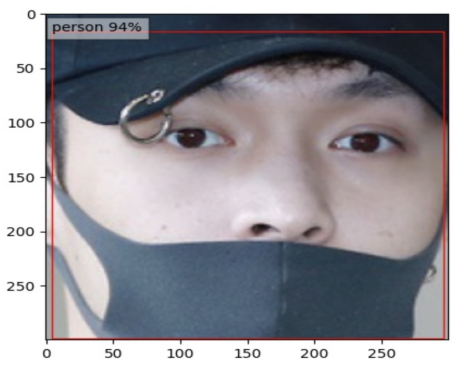
(a) SSD Result 1



(b) SSD Result 2



(a) SSD Result 3



(b) SSD Result 4

6.1.4 FasterRCNN Model

If we chose this option, we would likely build the classifier atop the ResNet backbone, as done in this model (same as linked in section 5).

6.2 Some of our ideas and thoughts

• We also try to create a mask detector by our own. This might be made by adding new blocks in the MobileNet class in our Github repository here, or exploring any of the other models listed above.
• In our model, if we train the mask detector on a small local box instead of the whole image, then it will might reduce the training time and lead to more accurate result. Thus for each input image, we do the facial (or person) detection first listed in subsection 0.1 detected in a small box, and then train the mask detection in this small box.
• With respect to facial recognition process, there are a lot of feature can be found on faces: eyes, noses, mouth, etc. Since we need to proceed nose detection and mouth detection after mask detection, thus we can also substitute the face detection process by three different process: eyes detector, nose detector, and mouth detector. As there are

quite a lot of ”fake noses” detected by HAAR, those three separate detections might be helpful. (For example, we report that noses are detected only when we can also detect eyes several pixels above from it.)

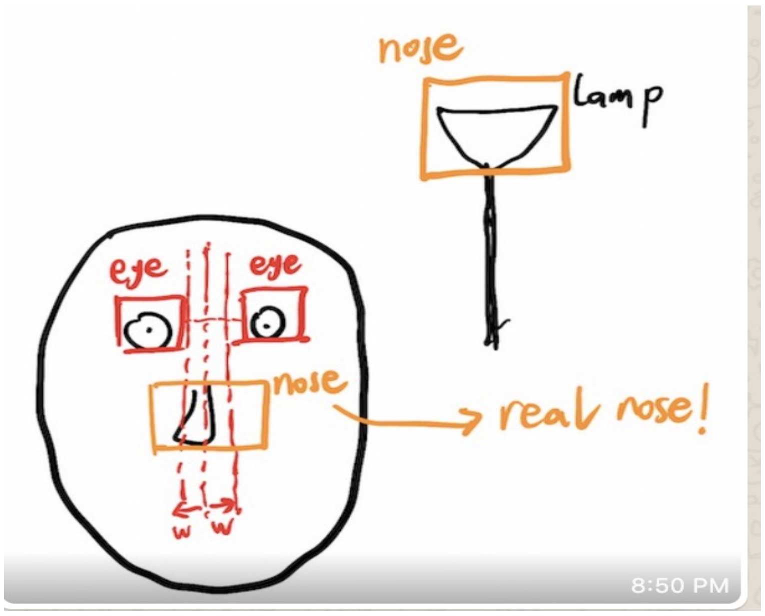


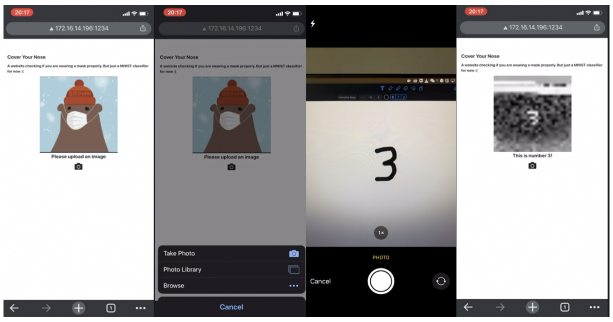
Figure 9: Lamp: fake nose ! Eyes detected above it: real nose !

• One drawback of the SSD model is that the COCO dataset it trains from, has no categories named like ”Mask”,”Covering” or ”noses”. The most related category we might use is ”Person”. So we are wondering how to apply SSD model. Indeed, it seems that we would need to rely on some transfer-learned backbone like ResNet, and then learn our own binary or ternary classifier on top with our own mask-related classes.

6.3 Model deployment

As our ultimate goal is to create a mobile website with our CV models running on the back end, we also need to figure out how to deploy the models to cloud. As a first step, we managed to deploy a tensorflow MNIST classifier to an AWS instance, and made a basic front end.

We built the server with FLask and deployed it with Docker. It can be visited here. Taking a photo (about 3 MB) and upload it to the server may not work smoothly, as we are using a free-tier instance. But it works well on a local server.



Cover Your Nose

A website checking if you are wearing a mask properly. But just a MNIST classifier for now :(

Please upload an image

This is number 3!

Figure 10: MNIST classifier working on the back end

References

[1] Introduction website for opencv haar cascades classifier
`https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970c`

[2] Haar Cascades classifier resources.
`http://alereimondo.no-ip.org/OpenCV/34`

[3] AIZOO Mask model resource.
`https://github.com/AIZOOTech/FaceMaskDetection`
`https://demo.aizoo.com/face-mask-detection.html`

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C.Berg [*SSD: Single Shot MultiBox Detector*] arXiv:1512.02325v5 [cs.CV] 29 Dec 2016
`https://arxiv.org/pdf/1512.02325.pdf`

[5] Mingjie Jiang, Xinqi Fan, Hong Yan [*RetinaFaceMask: A Face Mask Detector*] arXiv:2005.03950v2 [cs.CV] 8 Jun 2020
`https://arxiv.org/pdf/2005.03950.pdf`

[6] Face Mask image (or video) dataset
`https://www.kaggle.com/andrewmvd/face-mask-detection`
`https://www.kaggle.com/omkargurav/face-mask-dataset`
`https://public.roboflow.com/object-detection/mask-wearing/4`
`https://data.mendeley.com/datasets/v3kry8gb59/1`

[7] Wider Face Dataset
`http://shuoyang1213.me/WIDERFACE/`

[8] MAFA dataset
`https://www.kaggle.com/rahulmangalampalli/mafa-data/version/1`

[9] COCO dataset
`http://images.cocodataset.org/zips/train2017.zip`
`http://images.cocodataset.org/zips/val2017.zip`

[10] Our Github repository
https://github.com/player-eric/cs1430-final-project