Facemask Detection

Moises Figueroa

Problem & Context

In the month of February 2021, the CDC issued an order (LINK) that requires all passengers utilizing public conveyance (airplanes, buses, trains, ships) to wear facemasks. They must wear them before boarding, disembarking, and during the duration of travel. According to the mandate, conveyance operators must use best efforts to ensure that all passengers are wearing a mask. One could imagine that this requires extra effort on the part of the operators to make these checks.

The goal of this project is to design a model that detects if a person's face has a mask or not in order to lessen the load of conveyance operators. Theoretically, this model can be used with a camera in an entryway to determine if individuals are wearing masks.

Data

Data was sourced from Kaggle. The original dataset contains 853 images of individuals and crowds along with an accompanying XML file, formatted in Pascal VOC, for each image that contains coordinates for all the faces in an image along with a label (with, without facemask). This dataset was likely manually labeled using an image labeling program like LabelImg.

No cleaning step was necessary for these raw images.

Data Preprocessing

These were the steps taken to preprocess each image:

1. Read in the XML file that an image was associated with
2. Parse out the section of the XML that contained labels and coordinates for faces
3. Extract those faces using pixel coordinates
4. Resize extracted faces to 50 x 50 if the original face was at least 20 x 20 in order to avoid blurry faces

5. Place the resized face in a folder that corresponds to its label

After these steps, we are left with around 1500 images of faces with masks and around 300 images of faces without masks. In order to balance the classes, more pictures of faces without masks were sourced from a Google Drive repository.

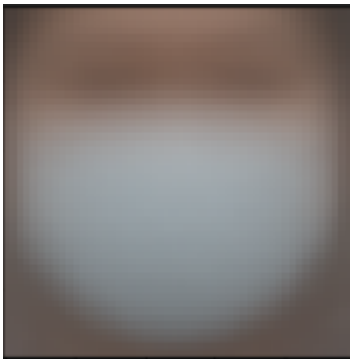
Since these pictures were purely pictures of faces, the only preprocessing step that had to be taken was to resize the images to 50 x 50. Now we have an even number of face samples with masks and without.

Data augmentation usually would also be performed here, but I am using keras. Keras provides a module called ImageDataGenerator that performs data augmentation while training. Some options include rescaling, random rotation, width shifting, height shifting, horizontal flipping, etc.

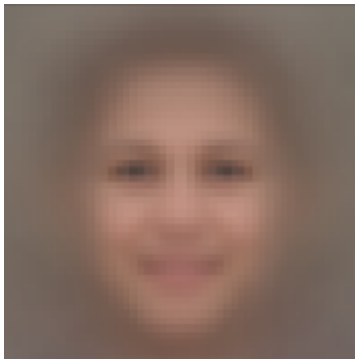
A Train/Test split of 80/20 was performed after classes were balanced.

Exploratory Analysis

For this task, there is little data to explore. The most illuminating detail that we can see from exploring are the average picture of both classes. You can clearly see there are differences between them.



Average Face With Mask



Average Face Without Mask

Something to note is that the picture on the right is less blurry than the picture on the left. This is likely because we had to take samples from another dataset that had clearer/sharper images to balance the classes, so the average of those pictures are also clearer.

Method

Convolutional Neural Networks are a type of neural network that performs excellent in image classification tasks. One of the huge advantages over this type of architecture over traditional methods is that CNNs have the ability to extract features from images that it then uses for learning. Traditional methods involve manually creating these features, which can be tedious.

For this task, we will be designing a small, relativity simple CNN to detect if a face is wearing a mask or not.

These are the components of the CNN:

1. Convolutional Layers + ReLU
2. Batch Normalization Layers
3. Pooling Layers
4. Dropout Layers
5. Fully Connected layer

This CNN takes in RGB images that are 50 x 50 and outputs a single float between 0 and 1. Outputs greater less than .5 will signify no mask and outputs greater than .5 will signify a mask is being worn.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 48, 48, 32)        896
_________________________________________________________________
batch_normalization_3 (Batch (None, 48, 48, 32)        128
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 46, 46, 64)        18496
_________________________________________________________________
batch_normalization_4 (Batch (None, 46, 46, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 23, 23, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 33856)             0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               4333696
_________________________________________________________________
batch_normalization_5 (Batch (None, 128)               512
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 4,354,113
Trainable params: 4,353,665
Non-trainable params: 448
_________________________________________________________________
```

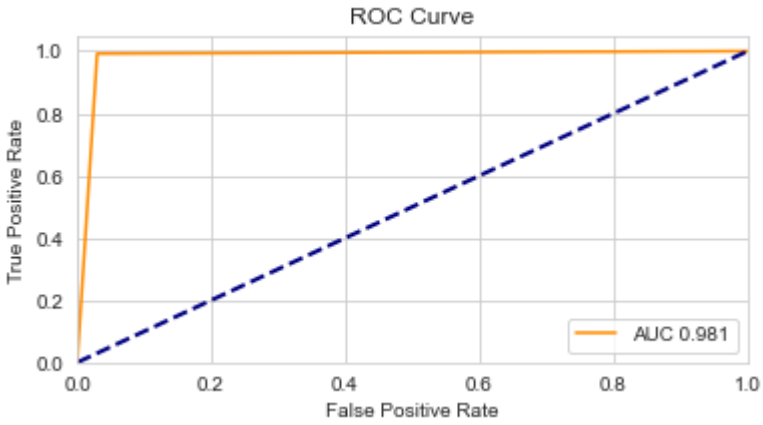
CNN Structure

The model was trained on 100 epochs. During training, three callbacks were used to speed up training and to find the best weights. Early stop with a patience of 20 was used to save time. Reduce Learning rate on Plateau with a patience of 5 was used, in case the model was stuck on not improving. And a checkpoint callback was declared to only save the best weights.

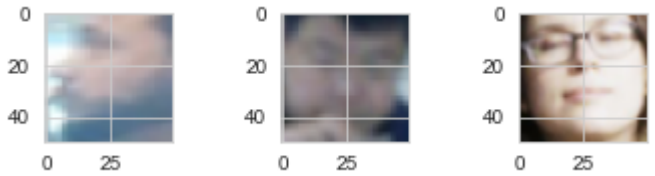
Results

The results on the validation set were as follows:

	precision	recall	f1-score	support
with_mask	0.99	0.97	0.98	371
without_mask	0.97	0.99	0.98	371
accuracy			0.98	742
macro avg	0.98	0.98	0.98	742
weighted avg	0.98	0.98	0.98	742



False Negatives



False Positives

The False negatives images are likely due to two factors: Mask design and color. Unique designs are not the norm, so there is probably not enough training data to account for designs. The model also seems to have a hard time identifying pink masks. This is probably because of skin tone.

The False positives are likely due to bad images. They all seem pretty blurry.

Drawbacks/Limitations

The original goal was to rig the model to a camera for face mask detection, but there is a problem in implementation. In order to determine if a face has a mask or not, something has to first identify the face in question. This is the realm of object detection and I believe it is out of the scope of this project.

The next best thing to do would be to use a pre-existing model (MTCNN) to first identify faces, then feed that extracted face to the Facemask Detection Model in order to determine if a found face is wearing a mask. The only drawback here is that the pre-existing model would not be great at detecting masks with faces, so these would not be fed into the model for classification. Essentially that means that when the pre-existing model has detected a face, there is already a high probability that the person is not wearing a mask, because it was picked up by the pre-existing model.

A viable solution to this problem would be to use Transfer Learning on a pre-existing object detections model. Transfer learning takes a pre-trained neural network and trains it with data that you provide. There are plenty of models to choose from (Tensorflow 2 Detection Model Zoo). The only issue with this solution is that you would need powerful GPU(s) to train the object detection model on your own custom dataset.

Conclusion

The model performs surprisingly well as is, considering the simplicity of the CNN. The issue it seems to have is detecting images with profiles of faces and exotic facemask designs. Further testing should be done with the context of its intended use in mind. For instance, in implementation images would be of high quality and would likely only show faces looking straight ahead.