1. **Problem Statement:** To determine if an individual in the image is wearing a mask or not using CNN. Originally, the idea was to use it in step with a raspberry pi, but I chose to start simpler, so I will be using my laptop's webcam to detect real-time results.

2. **Data Preprocessing**

   - Dataset: https://www.kaggle.com/andrewmvd/face-mask-detection

   - The dataset remains the same and is already split between test, train and validation sets. Each set is further split into images with and without masks. No data was further modified.

   - Train set: 600 with mask, 600 without mask to make 1200 images in total

   - Test set: 100 with mask, 100 without mask to make 200 images in total

   - Validation set: 300 with mask, 300 without mask to make 600 images in total

3. **Model**

Machine Learning Model: CNN (Convolutional Neural Network)

   - Ideal for image and computer vision design. Multi-layer perceptron that doesn't feed outputs back into the model to train it but simply produces an output from an input

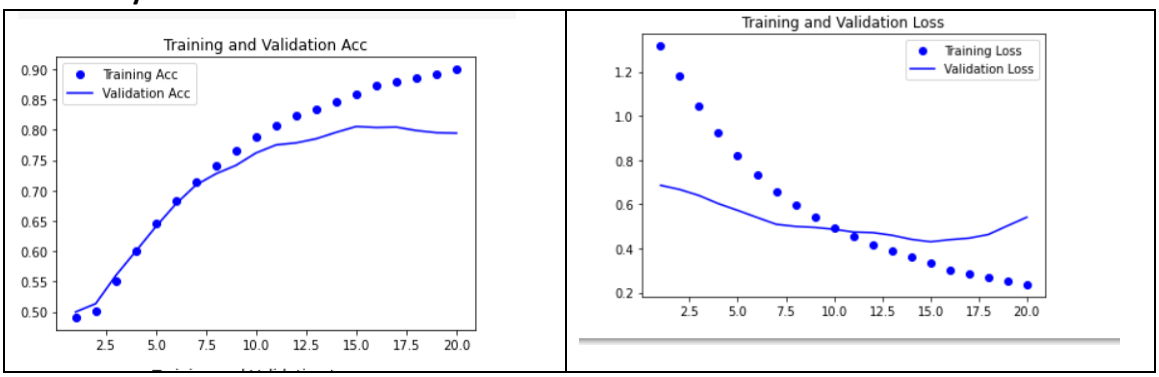
a) Libraries: PyTorch and Keras
   - Layers: 3 Convolution Layers, 3 Max-Pooling Operations (0.5 by 0.5 window), 3 Dropout layers, Flattening Layer, Dense Layer

b) The dense layer reduces dimensionality of inputs by using backpropagation, and tuning the hyperparameters of the weight matrix and the sigmoid function governed by the ReLU activation function
   - The Adam algorithm will be used to perform gradient descent to find the cost function

c) The model was tested over epochs and the graph showing training and validation loss show that training loss increases linearly over time whereas validation accuracy stalls around 50%. The difference in training and validation loss shows that the data is overfitting.

d) It was difficult to determine how many different layers should be implemented in the model to avoid underfitting or overfitting the data but testing different parameters and common practices, I was able to optimize the model's accuracy for the data.

4. **Preliminary Results**



5. **Next Steps:**
   - To prevent the model from overfitting, I could add more training samples so there's no opportunity for the model to generalize on a small number of data samples. Adding dropout layers could help prevent overfitting, but it would decrease the small numbers of outputs already available so data augmentation may be the better thing to implement.