**Facemask and Proximity detection**

**Problem Statement**: Design a computer vision model to detect facemask along with interpersonal proximity. Send alerts if social distancing is not maintained.

**Approach**:

The project can be broken down into two sections:

1. Detect facemask
2. Determine object proximity

**Section I: Facemask Detection**

The first part of section 1 is data collection. I managed to find a public repository on Kaggle containing ~200 images classified into ‘Mask’ and ‘No Mask’ categories. We use FaceNet embedding to help us with classification in Section 2 which allows us to operate with smaller training data volume. Section 1 can then be further broken down into two phases; Model Training and Deployment. I use transfer learning i.e. build the mask detection model on top of a pre-trained face detection model.

Phase 1:

1. Load facemask data
2. Split the dataset into training and validation data
3. Train convolution model using transfer learning (MobileNetV2)
   1. Base layer 🡪 Avg Pooling 🡪 Flatten 🡪 ReLu Dense layer 🡪 Dropout 🡪 Softmax Dense layer
4. Fit the model
   1. Optimiser: ADAM
   2. Loss function: Binary cross-entropy
5. Validate model performance and accuracy
6. Save .h5 model

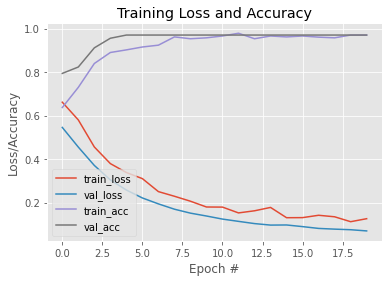


Figure : Training loss and accuracy

Phase 2:

1. Load the trained model along with the pre-trained Caffe model and prototxt model architecture.
2. Setup up webcam video stream using imutils. Convert each frame into an image blob:
   1. Extract dimensions
   2. Resize to 300px
   3. Create blob using OpenCV’s Deep Neural Network (DNN) blob reader
3. Loop through the feed and filter objects based on prediction confidence. I set the confidence boundary at 40% given the quality of the camera.
   1. It initially detects whether the stream contains a face
   2. Calculate probability of a mask on the face
   3. Show ‘Mask Detected’ if at least 40% confident
4. To improve parsing rate, it is advisable to pass a batch of images (32) for prediction instead of processing individual frame.
5. Draw a box and display messages on the webcam video stream.

**Section II: Social Distancing Proximity**

Like Section I, object proximity can also be broken down into transfer-learning based training and deployment. But since there are no visual changes to object detection for body features, I used a Caffe version MobileNet network architecture that has been pre-trained using the COCO dataset, combined with Single Shot Detectors (SSD) to jump straight to proximity measure. There are three frequently used object detection methods: Faster R-CNN, YOLO and SSD. Faster R-CNN is difficult to implement and unlike the name, not especially fast. YOLO has faster processing capabilities but it lacks accuracy in comparison, but in order to deploy the code on an edge device with limited data, I decided to go with SSD. SSD balances the two methods in processing speeds, accuracy and resource utilization.

Once the models have been initialized and video stream loaded, we can perform proximity calculations.

1. Load the trained model along with the pre-trained Caffe model and prototxt model architecture.
2. Setup up webcam video stream using imutils. Convert each frame into an image blob:
   1. Extract dimensions
   2. Resize to 300px
   3. Create blob using OpenCV’s Deep Neural Network (DNN) blob reader
3. Loop through the feed and filter objects based on prediction confidence.
4. If 2 objects identified (40% confident), calculate Euclidean distance between objects. Display warning if objects too close.