PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100 Ft. Road, BSK III Stage, Bengaluru – 560 085

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Title: Image Processing and Data Visualization Using MATLAB		
Course code: -UE19CS257B		
Semester : 4th sem	**Branch:CSE**	**Team Id:48**
SRN:PES1UG19CS323	**Name:PARIMALA S**	
SRN:PES1UG19CS324	**Name:PARIMALA V**	
SRN:PES1UG19CS463	**Name:SHREYA VISHWAS**	

PROJECT REPORT

Problem Statement: CovidSafe – Face Mask Detector

Built based on Image Processing and Computer Vision

Objectives:

This Project aims to detect whether the faces detected in the frame are properly wearing mask or not. There are 2 models which are a part of this.
The first model is implemented by blending Deep Learning Algorithmic models into Image processing done using MATLAB.
The second one is built using Viola Jones Algorithm.

Description:

- This project is implemented with a basic interface supported by Matlab which either takes live input from the camera or from any picture fed as input.
- The face/s present in the image are detected. This is further used to detect whether a mask is present in the face or not.
- The Dataset has been adapted from a Google drive source which is partitioned for training and testing in the ratio of 9:1 to achieve a good efficiency.
- The dataset has around 2800 images which are classifies into 2 folders as – mask with 1281 images and no_mask with 1486 images.

MODEL 1:

- This is implemented using Deep learning model – neural network consisting of 15 layers including layers such as the Input layer, Convolution layer – Max Pooling layer – reLu layer – batch normalization layer and SoftMax- (sigmoid activation function) layer for classification and error minimization.
- Various parameters like Training Epochs, Learning rate, iterations per epoch, Mini-Batch size and Validation data etc are tuned.
- The designed deep learning model is then trained on the training image dataset and then tested using the rest of the images.
- The training dataset is obtained from the augmented partition of Image Datastore.
- The trained network is saved for importing it to detect the face masks.
- **This model has an efficiency of 94.5%**

DEMO :

- The model is invoked which starts the device camera if it exists.
- The entire demo works based on live input from the webcam.
- It takes each frame of the webcam input and detects face mask.
- First, it detects if there are faces in the camera. If yes, it proceeds to detect all of them and draws a boundary around each one of it.
- If faces are present, it invokes the trained neural network and detects the presence or absence of mask.
- This detects the mask on multiple faces also, if present in the camera and marks it as the label along with the face boundary.
- Press Ctrl+C to exit.

MODEL 2:

- This is implemented using Viola Jones model.
- This uses the existing approach of the Viola Jones algorithm which uses Haar basis feature filters.
- This uses a simple logical approach of detecting face masks by detecting the presence/absence of the facial features.
- It uses the basic idea that, if the mask is being worn properly, then the features- nose and mouth will completely be not seen.
- Hence if the model detects nose/ face or both, then the conclusion is that the mask is not worn /properly.
- This model works on the images directly by importing them from the datastore.

DEMO:

- The model is invoked which picks an image from testing dataset's datastore.
- It pre-processes the image by resizing it to the required resolution.
- It takes this input and detects the face mask.
- First, it detects if there are faces in the camera. If yes, it proceeds to detect all of them and draws a boundary around each one of it.
- It shows 3 cases in total –
    - No faces found
    - Face found without a mask
    - Face with mask
- It uses the basic idea that, if the mask is being worn properly, then the features-nose and mouth will completely be not seen.
- It outputs 4 images – Input image, Face Detected Image, Nose detection in the Image and Mouth Detected Image and the corresponding final message.

New Concept Learnt(Explanation):

- Deep Learning in Matlab:
  We learnt a lot of things under this domain.

    1. Building the convolutional neural network with multiple hidden layers and classification layer with appropriate loss minimization layer:

       CNNs eliminate the need for manual feature extraction—the features are learned directly by the CNN. They produce highly accurate recognition results. They can be retrained for new recognition tasks, enabling you to build on pre-existing networks.

       The primary objective of the Convolutional Layer is to extract features such as edges, lines, and basic shapes, from the input image. The early convolutional layers extract very low-level features (colour, lines, gradients) with subsequent layers extracting higher-level features (lines, shapes, objects).

       There is a Pooling layer which extracts the dominant features which are both rotational and positional invariant within the input image, thus helping to maintain strong prediction capability. There are two common types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the filter.

Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

2. Training the neural network on data.

   For image classification and image regression, train a single network in parallel using multiple GPUs or a local or remote parallel pool. Training on a GPU or in parallel requires Parallel Computing Toolbox. When training a neural network, specify the predictors and responses as a single input or in two separate inputs.
   Syntax: **net = trainNetwork(images,layers,options)** trains the neural network specified by layers for image classification and regression tasks using the images and responses specified by images and the training options defined by options – hyperparameters of the model.

```
miniBatchSize = 10;
valFrequency = floor(numel(augimdsTrain.Files)/miniBatchSize);
options = trainingOptions('sgdm', ...
    'MiniBatchSize',miniBatchSize, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',3e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',valFrequency, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

3. Creating Augmented Image datastore.

   An augmented image datastore transforms batches of training, validation, test, and prediction data, with optional preprocessing such as resizing, rotation, and reflection. Resize images to make them compatible with the input size of your deep learning network. Augmenting training image data with randomized preprocessing operations helps prevent the network from overfitting and memorizing the exact details of the training images.

   Our model trains the network using augmented images, by supplying the augmentedImageDatastore to trainNetwork.
   Syntax: **auimds = augmentedImageDatastore(outputSize,imds)** creates an augmented image datastore for classification problems using images from image datastore imds, and sets the OutputSize property.

4. Testing the saved network on live input.

   The trained neural network can be saved for later use by using
   save networkName;
   This network can be used later using

load (networkName.mat);
All the saved networks have the extension .mat.

5. Labeling images for neural networks:

   Labels here, refer to the classes that images get categorised into, after Image classification by the neural network. The labels have to be assigned to all images in the dataset so that images can be trained.
   In our model, labels are taken from the name of the folder it exists in – 'with mask' or 'without mask'.

6. Locating the face/mask and drawing the bounding box with appropriate label.

   This is used to identify the face if present on the image and then crop the same and send that as input for mask classification.
   Syntax:
   rectangle('Position', BB(faces_iter,:), 'Linewidth',2,'LineStyle',' ','EdgeColor',line_color);
   draws rectangular bounding boxes around the faces with specified line attributes.

- Viola Jones Algorithm based Model:
  1. Viola Jones Algorithm

     Viola-Jones was designed to detect frontal faces, rather than faces looking sideways, upwards or downwards. Before detecting a face, the image is converted into grayscale, since it is easier to work with and there's lesser data to process. The Viola-Jones algorithm first detects the face on the grayscale image and then finds the location on the coloured image.
     The algorithm has four stages:
     - Haar Feature Selection
     - Creating an Integral Image
     - Adaboost Training
     - Cascading Classifiers

  2. Accessing Images from Image datastore.

     ImageDatastore object is used to manage a collection of image files, where each individual image fits in memory, but the entire collection of images does not necessarily fit. You can create an ImageDatastore object using the imageDatastore function, specify its properties, and then import and process the data using object functions.

Syntax: **imds = imageDatastore(location,Name,Value)** specifies additional parameters and properties for imds using one or more name-value pair arguments

3. Processing images
   Enhancing the image, resizing and cropping to the required size using imresize, imcrop etc.

4. Vision.CascadeObjectDetector

   The cascade object detector uses the Viola-Jones algorithm to detect people's faces, noses, eyes, mouth, or upper body.
   Syntax:
   detector = vision.CascadeObjectDetector
   creates a detector to detect objects using the Viola-Jones algorithm.

5. Merge Thresholding while detecting facial features.

   Merge thresholding or Detection threshold is an integer. The threshold defines the criteria needed to declare a final detection in an area where there are multiple detections around an object. Groups of colocated detections that meet the threshold are merged to produce one bounding box around the target object. Increasing this threshold may help suppress false detections by requiring that the target object be detected multiple times during the multiscale detection phase.

6. Detecting Facial Features:

   The cascade object detector is further used to detect various facial features such as nose , mouth etc .
   Syntax:
   NoseDetect =
   vision.CascadeObjectDetector('Nose','MergeThreshold',16);

Learning Outcome:

- Sound knowledge of Deep learning – Convolutional Neural networks with MATLAB
- Training CNN for an image classification problem and testing on sample images.
- Running the model on live webcam input and detecting face mask.
- Face mask detection for multiple faces in the same frame.

- Learning and Implementing Loss minimization layers and achieving good efficient model.
- Learning and Understanding the basics of Viola Jones model.
- Applying Viola Jones algorithm for Face mask detection.
- Detecting the face and facial features using cascading detection
- Obtained conclusive results and efficiency of 94.5% for the model
- Most Importantly , learning the art of debugging errors and warning messages.

Code:

There are 3 Code files written.

1. Training.m – It has the code pertaining to building the deep learning network and training the model.

```
clc;

layers = [
    imageInputLayer([227 227 3],"Name","imageinput")
    convolution2dLayer([3 3],3,"Name","conv_1","Padding","same")
    batchNormalizationLayer("Name","batchnorm_1")
    reluLayer("Name","relu_1")
    maxPooling2dLayer([2 2],"Name","maxpool_1","Padding","same")
    convolution2dLayer([3 3],3,"Name","conv_2","Padding","same")
    batchNormalizationLayer("Name","batchnorm_2")
    reluLayer("Name","relu_2")
    maxPooling2dLayer([2 2],"Name","maxpool_2","Padding","same")
    convolution2dLayer([3 3],3,"Name","conv_3","Padding","same")
    batchNormalizationLayer("Name","batchnorm_3")
    reluLayer("Name","relu_3")
    fullyConnectedLayer(2,"Name","fc")
    softmaxLayer("Name","softmax")
    classificationLayer("Name","classoutput")];

ins=layers(1).InputSize;

    folders = fullfile('dataset');

    imds = imageDatastore(folders,'IncludeSubfolders',true,'LabelSource','foldernames');
    tbl = countEachLabel(imds)
    [trainImgs,testImgs] = splitEachLabel(imds,0.9);

    audsTrain=augmentedImageDatastore([227 227 3],trainImgs);
    audsTest=augmentedImageDatastore([227 227 3],testImgs);



 opts = trainingOptions('sgdm', ...
          'Plots', 'training-progress', ...
          'ValidationData',audsTest, ...
          'MaxEpochs',10,...
          'MiniBatchSize', 32,...
          'InitialLearnRate',0.001);


net = trainNetwork(audsTrain,layers,opts);
gen_net = net;
save gen_net;
```

2. Facedetecttest.m – This code file is a client file which takes live camera input runs the deep learning model on it and displays required output.

```
close all;
clear all;
clc;


% Loading the trained network
load('gen_net.mat');
%load(gen_net);

% Connect to the camera
if ~exist('camera')
    camera = webcam;
end
cr_label = ' Face Mask Detection ';
img_size = [227,227];
frameCount = 0;


FDetect = vision.CascadeObjectDetector;

while true && frameCount<400
    picture = camera.snapshot;
    orig_picture = picture;
    BB = step(FDetect, picture);
    frameCount=frameCount+1;

    if size(BB,1) >= 1 % if a face is found
        for faces_iter = 1:size(BB,1) % for total number of faces found in the camera
            picture_cropped = imcrop(orig_picture,BB(faces_iter,:));
            %imshow(picture_cropped)
            picture_resized = imresize(picture_cropped,img_size);
            %imshow(picture_cropped)
            label = classify(net, picture_resized); % find out if it has mask or not
            label_text = char(label);

            text_color = 'green';
            line_color = 'g';
            if strcmp(label_text, 'without mask')
                text_color = 'red';
                line_color = 'r';
            end
        picture = insertText(picture,BB(faces_iter,1:2),label_text,'FontSize',16,'BoxColor',...
            'white','BoxOpacity',0,'TextColor',text_color); % text label around each face
        image(picture); % show the picture
        axis off;
    end
    for faces_iter = 1:size(BB,1)  % bounding boxes around each face
        rectangle('Position', BB(faces_iter,:), 'Linewidth',2,'LineStyle','-','EdgeColor',line_color);
    end
    label = strcat(cr_label,'| Detected face(s) = ',num2str(size(BB,1)));
else % if no face found
    image(picture); % show the picture
    picture = imresize(picture,img_size);
    axis off
    label = strcat(cr_label,'| Detected face(s) = 0');
end
```

```
    label_cell{1} = label;
    title(label_cell, 'FontSize', 24); % show the label
    drawnow;

    fprintf('Detected face(s) = %d. Press ''Ctrl+c'' to end...\n', size(BB,1))
end

clear camera;
```

3. Facedettest.m – This file contains code pertaining to Viola-Jones Model. This is the second model created. It has complete model code.

```
close all;
clear all;
clc;
flag = 0;
img_size = [227,227];

folders = fullfile('dataset');
%Detect objects using Viola-Jones Algorithm

%To detect Face
FDetect = vision.CascadeObjectDetector;
imds = imageDatastore(folders,'IncludeSubfolders',true,'LabelSource','foldernames');
tbl = countEachLabel(imds)
[trainImgs,testImgs] = splitEachLabel(imds,0.9);
%Img = readimage(testImgs,2)

[fileName,pathName] = uigetfile('Profile.jpg')
Img = imread(fullfile(pathName, fileName));


I = imresize(Img,img_size);
imshow(I);
hold on
title('Input Image');
hold off;

BB = step(FDetect,I);
if(isempty(BB))
    msgbox("No Face Detected",'Conclusion');
else
    figure,
    imshow(I); hold on
    for i = 1:size(BB,1)
        rectangle('Position',BB(i,:),'LineWidth',5,'LineStyle','-','EdgeColor','r');
    end
    title('Face Detection');
    hold off;

    NoseDetect = vision.CascadeObjectDetector('Nose','MergeThreshold',16);
    BB=step(NoseDetect,I);
    flag = isempty(BB)
```

```
    figure,
    imshow(I); hold on
    for i = 1:size(BB,1)
        rectangle('Position',BB(i,:),'LineWidth',4,'LineStyle','-','EdgeColor','b');
    end
    title('Nose Detection');
    hold off;

    %To detect Mouth
    MouthDetect = vision.CascadeObjectDetector('Mouth','MergeThreshold',16);

    BB=step(MouthDetect,I);
    flag = flag | isempty(BB)

    figure,
    imshow(I); hold on
    for i = 1:size(BB,1)
        rectangle('Position',BB(i,:),'LineWidth',4,'LineStyle','-','EdgeColor','r');
    end
    title('Mouth Detection');
    hold off;

    if(flag == 1)
        msgbox("Wearing Mask Properly",'Conclusion');
    else
        msgbox("No mask",'Conclusion');
    end
end
```

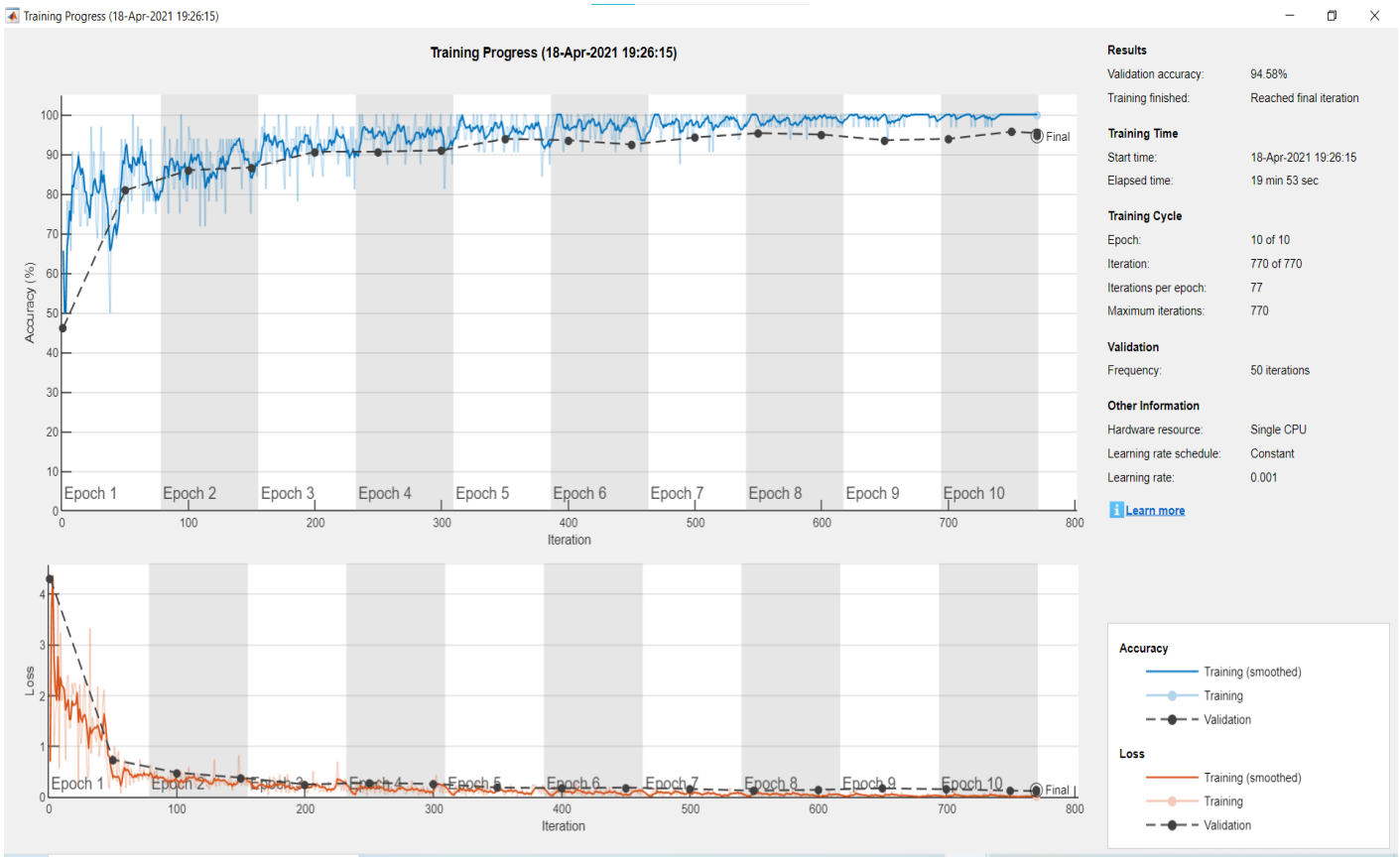
Output Screenshots:

Complete Summary of The Model Training and Various Hyperparameters:

Command Window

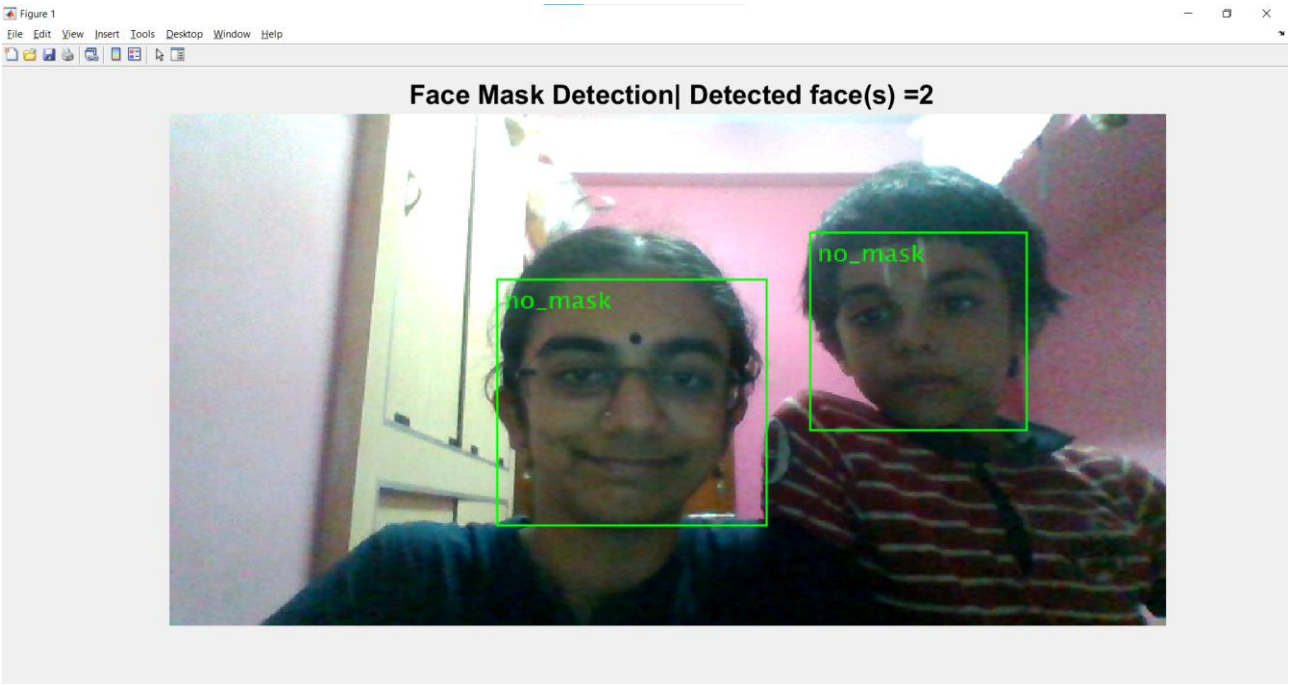
```
     Label      Count
    _______    _____

    mask       1281
    no_mask    1486

Training on single CPU.
Initializing input data normalization.
|=========================================================================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate       |
|=========================================================================================================================================|
|       1 |           1 |       00:00:19 |       65.63% |       46.21% |       0.6969 |       4.3040 |          0.0010 |
|       1 |          50 |       00:01:38 |       93.75% |       80.87% |       0.3416 |       0.7338 |          0.0010 |
|       2 |         100 |       00:02:51 |       96.88% |       85.92% |       0.1676 |       0.4765 |          0.0010 |
|       2 |         150 |       00:04:08 |       87.50% |       86.64% |       0.2310 |       0.3781 |          0.0010 |
|       3 |         200 |       00:05:22 |       93.75% |       90.61% |       0.1242 |       0.2458 |          0.0010 |
|       4 |         250 |       00:06:36 |       96.88% |       90.61% |       0.1543 |       0.2771 |          0.0010 |
|       4 |         300 |       00:07:50 |       90.63% |       90.97% |       0.2731 |       0.2562 |          0.0010 |
|       5 |         350 |       00:09:07 |       93.75% |       93.86% |       0.1265 |       0.1841 |          0.0010 |
|       6 |         400 |       00:10:18 |       96.88% |       93.50% |       0.0972 |       0.1733 |          0.0010 |
|       6 |         450 |       00:11:34 |       96.88% |       92.42% |       0.0744 |       0.1836 |          0.0010 |
|       7 |         500 |       00:12:51 |       96.88% |       94.22% |       0.0680 |       0.1581 |          0.0010 |
|       8 |         550 |       00:14:05 |       96.88% |       95.31% |       0.0736 |       0.1226 |          0.0010 |
|       8 |         600 |       00:15:20 |       96.88% |       94.95% |       0.0346 |       0.1417 |          0.0010 |
|       9 |         650 |       00:16:40 |      100.00% |       93.50% |       0.0140 |       0.1689 |          0.0010 |
|      10 |         700 |       00:17:56 |      100.00% |       93.86% |       0.0262 |       0.1524 |          0.0010 |
|      10 |         750 |       00:19:13 |      100.00% |       95.67% |       0.0101 |       0.1234 |          0.0010 |
|      10 |         770 |       00:19:49 |      100.00% |       95.31% |       0.0148 |       0.1168 |          0.0010 |
|=========================================================================================================================================|
```

Training Progress Display after training the Deep learning model and validating on test data:



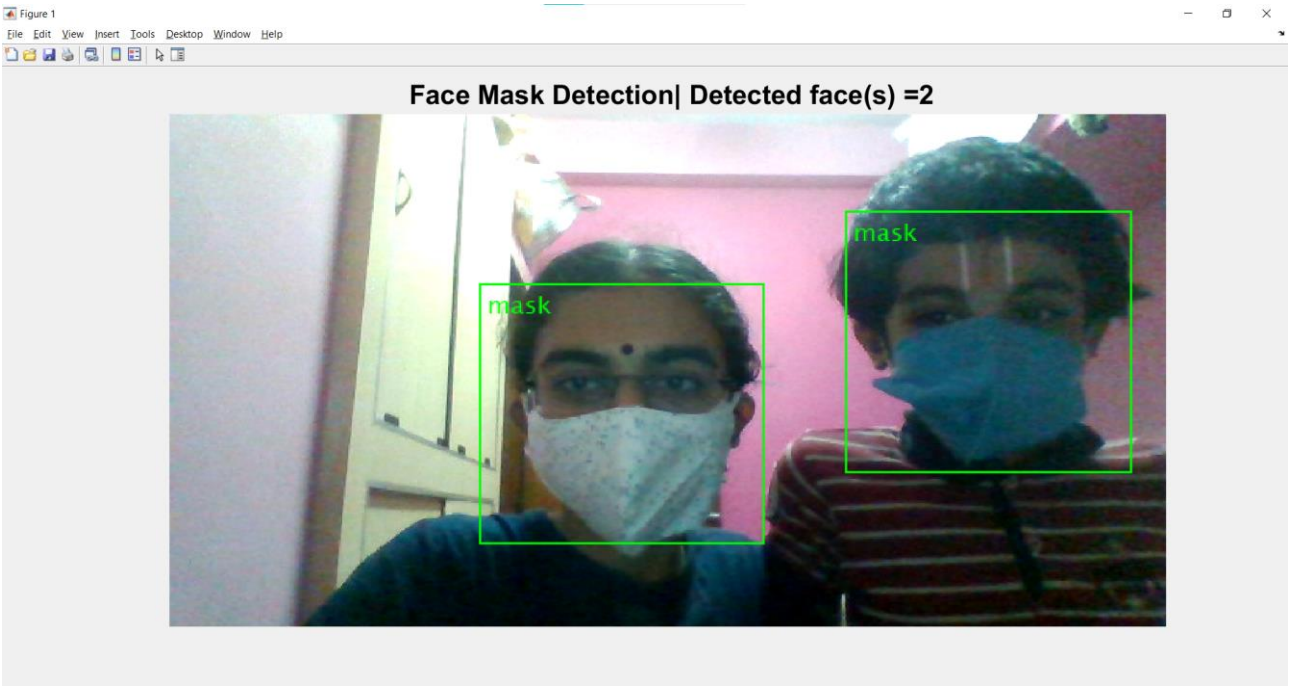
OUTPUTS FROM LIVE WEBCAM:

Output with multiple faces without mask:



Output with either of the faces with mask:

Output with all the faces wearing masks:

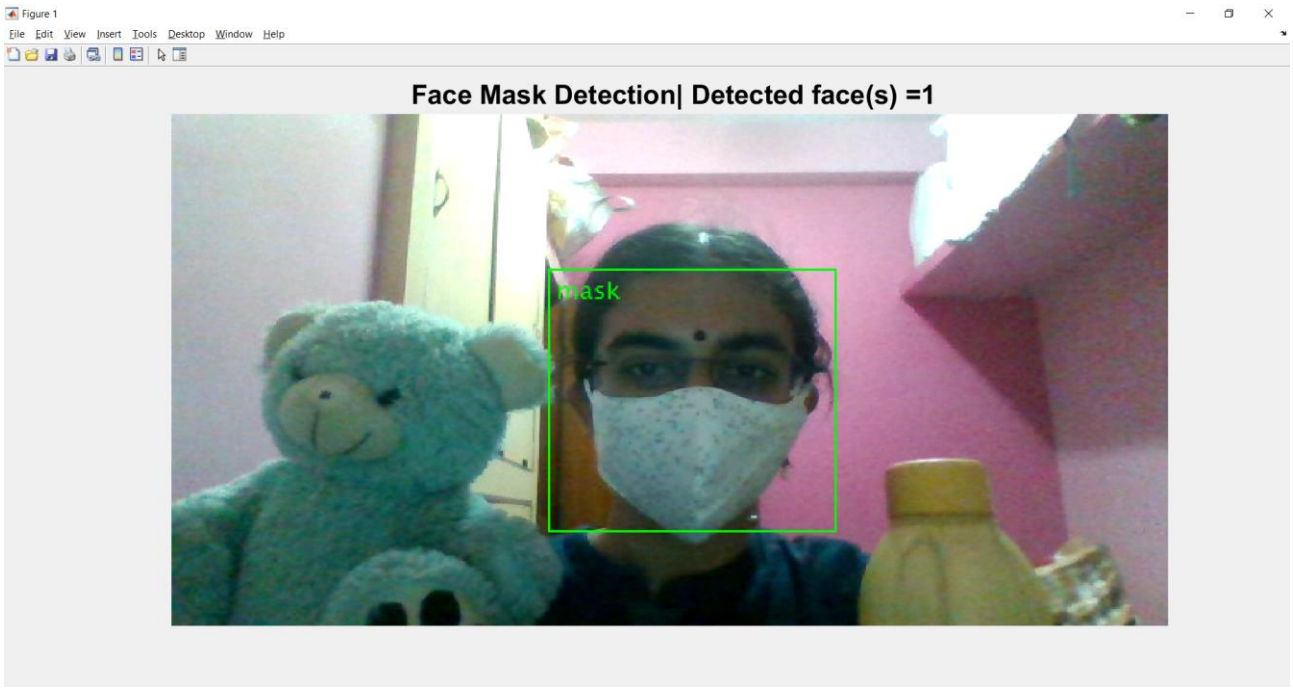


Command Window

```
Detected face(s) = 2. Press 'Ctrl+c' to end...
Detected face(s) = 1. Press 'Ctrl+c' to end...
Detected face(s) = 2. Press 'Ctrl+c' to end...
Detected face(s) = 2. Press 'Ctrl+c' to end...
Detected face(s) = 2. Press 'Ctrl+c' to end...
Detected face(s) = 2. Press 'Ctrl+c' to end...
Detected face(s) = 2. Press 'Ctrl+c' to end...
Detected face(s) = 2. Press 'Ctrl+c' to end...
Detected face(s) = 2. Press 'Ctrl+c' to end...
Detected face(s) = 1. Press 'Ctrl+c' to end...
Detected face(s) = 1. Press 'Ctrl+c' to end...
```

Command Window Output:

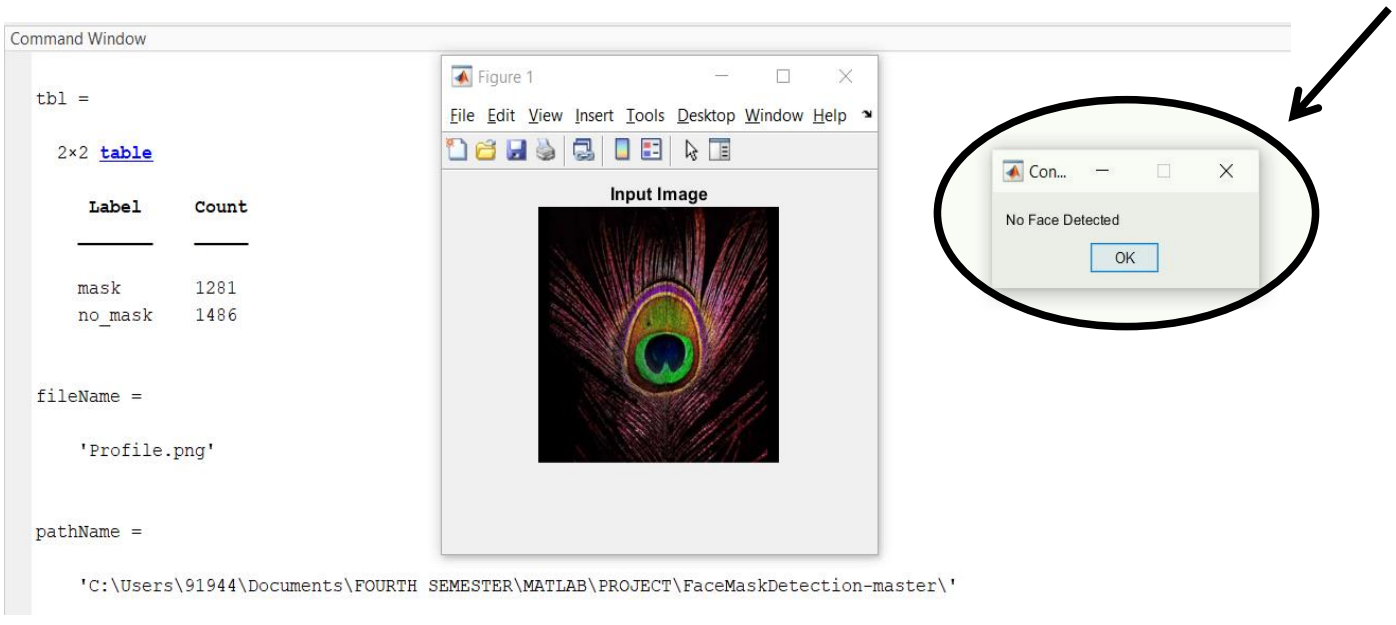
Does not mis detect with other objects in the image:



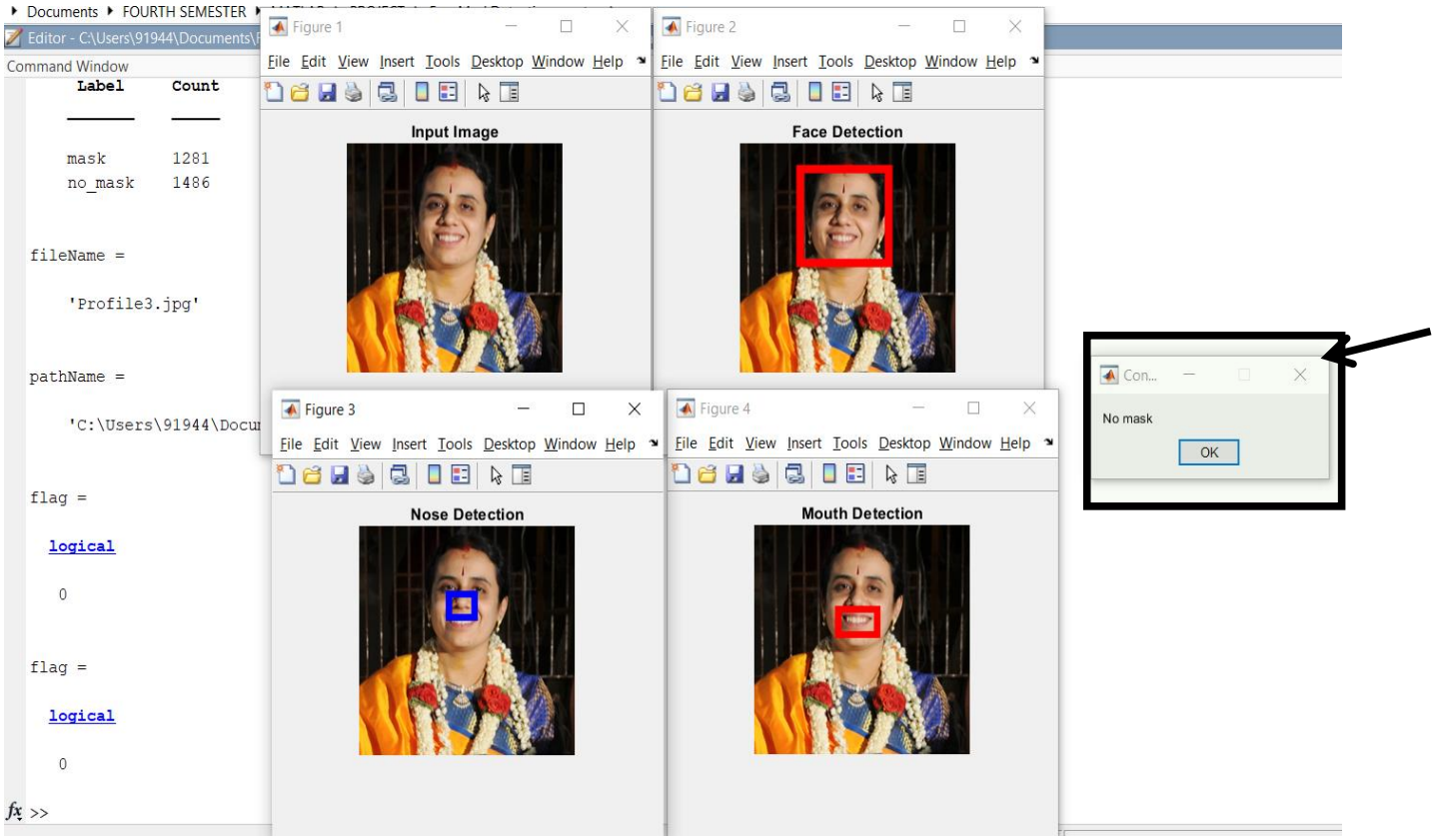
**** In this case face when found in the periphery is not detected with a great accuracy**

VIOLA JONES MODEL:

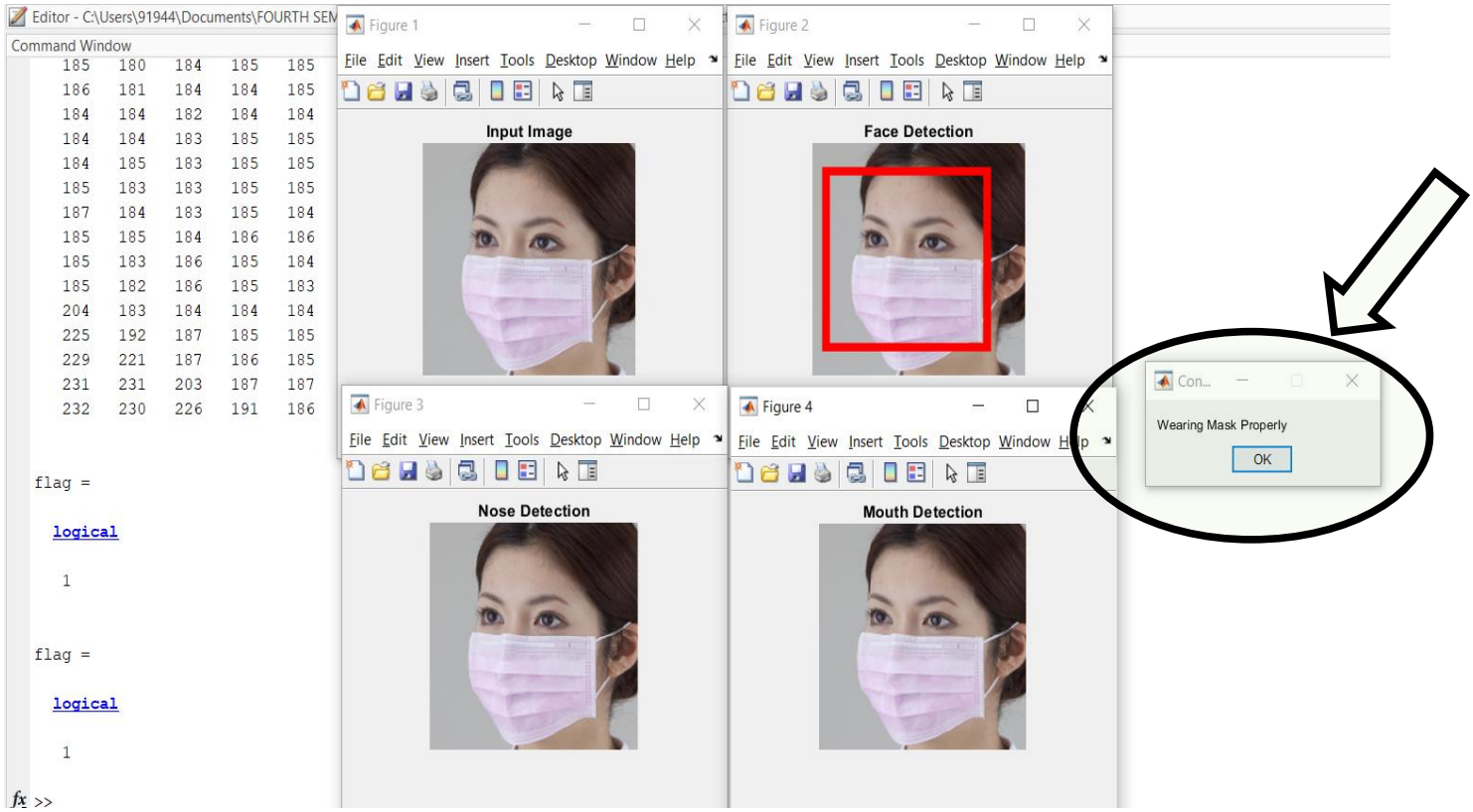
Output when Image has no face:



Output when Face has no mask:



Output when face has mask:



References:

1. **Mathworks.com**
2. **https://towardsdatascience.com/the-intuition-behind-facial-detection-the-viola-jones-algorithm-29d9106b6999#:~:text=The%20Viola-Jones%20algorithm%20first%20detects%20the%20face%20on,these%20haar-like%20features%2C%20which%20will%20be%20explained%20later.**

GitHub Link to this Project Repo:

https://github.com/ParimalaS27/Face-Mask-Detection

Name and Signature of the Faculty: Revathi G. P.