Multipurpose IoT Based Camera Using Deep Learning

Urvashi Dube, Sudhish Subramaniam, and G. Sumathi

Abstract The COVID 19 pandemic has given rise to a new normal. This includes wearing masks and maintaining social distance. Nowadays sudents don't focus in offline classes. Also, students with masks in offline proctored exams find ways to roll their eyes at others' work for malpractice. The systems designed to date are not accurate to detect facial features with mask. These problems have motivated us to develop a reliable, robust model to detect mask, eye location, eyeball location, eye status, and head pose of people wearing and not wearing a mask, all at once. We have used 3800 masked, unmasked images to train our model using MobileNetV2, a convolutional neural network, with 99% accuracy. The output of this model is processed using image processing, facial landmark analysis, EAR, and deep learning to detect the facial landmarks accurately. Ultimately, a unique method is used to detect head pose of person.

Keywords Image processing · Deep learning · Face mask · Eye location · Eyeball location · Head pose estimation

1 Introduction

Covid 19 pandemic has changed the style of living in the world. Although the virus is exiting, it has an impact on people. Schools and colleges have opened but people still wear masks to all places. In this masked environment, face image processing has faced a lot of challenges. To overcome all these problems, we have developed a model to detect the face mask accurately and to detect the eye, eye status, eye location, eyeball location and the head pose of the person in front of the camera. Our model is also designed to detect the attentiveness of the person in a classroom. This is one of the major problems faced in the school environment, students are not paying attention in the classroom under the masks, and this decreases the productivity of education in the schools, colleges and classes. Our model can also be used in the

U. Dube · S. Subramaniam · G. Sumathi (✉)
School of Electronics, VIT, Vellore, Tamil Nadu, India
e-mail: sumathi.g@vit.ac.in

J. Hemanth et al. (eds.), *Intelligent Cyber Physical Systems and Internet of Things*,
Engineering Cyber-Physical Systems and Critical Infrastructures 3,
https://doi.org/10.1007/978-3-031-18497-0_7

offline proctored exams where students are not allowed to roll their eyes elsewhere during the duration of the exam. To avoid malpractice, our model can be used in runtime to capture images and process the eye landmarks and head pose of the candidate. The model can be further progressed to end the test of the user if the person has attempted malpractice. We have used mobilenet_v2 to classify the 3800 images of masked and unmasked people. On getting a very high accuracy of 99%, we proceeded to filter out the eyes and other facial features. We ultimately determine mask, eye status, eye location, eyeball location, and head pose irrespective of the person wearing or not wearing a mask.

2 Related Work

The masking of a face can be identified by an edge computing-based deep learning algorithm [1], this method is specifically implemented in busses to identify people wearing masks or not. A dataset of masked faces (MAFA) and two CNN algorithms (LLE-CNN algorithm) are applied in [2]. A model comprising of PIR Sensor, microcontroller and smartphone system is proposed in [3] to detect the motion and store the corresponding output video in the cloud. In [4], a generative adversarial network for masked object detection and image completion of the removed masked region is proposed and implemented. Detection of wearing state of face mask by training a custom dataset: a face without a mask, face with the wrong mask, face with the correct mask is done using Context-Attention R-CNN method in [5]. Head Pose estimation can also be done using the HGL method which is a combination of the H-channel of the HSV colour space with the face portrait and grayscale image, this method is proposed in [6] and achieves an accuracy of 87.17%. The paper [7] includes two novel ideas, a residual context attention module for crucial face mask related regions and an auxiliary task using a synthesized gaussian heat map regression method to discriminate features of the face. The authors of [8] have proposed a model to detect candidates wearing mask regions using the transfer model of Faster_RCNN and InceptionV2 structure, in the second stage real facial masks are verified using a broad learning system. A simple and effective facial landmark detection method comprising of a lightweight U-Net model and a dynamic optical flow is proposed in [9] which exhibits better performance than others without requiring heavy computational loadings. The authors of [10] have developed a model using driving environment datasets and eye aspect ratio (EAR), to detect facial landmarks, eye location and state evaluation, they achieved an average accuracy of 93.9%. The authors of [11] used the method of segmentation of pupil and iris images by pixel to determine the eye status of the driver and his fatigue, they achieved an accuracy of 96.72%. In [12] fatigue detection convolutional network (FDCN) based CNN network was built which has a 1.0% accuracy improvement on the ZJU database on fatigue detection. DriCare, a new face-tracking algorithm to improve the tracking accuracy is developed in [13], and it achieved 92% accuracy. Eye status, PERCLOS of both coloured and infrared, fatigue is detected around the clock in [14]. Using logistic regression,

EAR and analyzing facial landmarks percent eye-closure over a fixed time window (PERCLOS), blink rate, statistics of blink duration, closing speed, reopening speed and number of yawns are extracted in [15]. In [16], eyes for a frontal face are extracted precisely. Histograms of Oriented Gradient (HOG) descriptors are proved to be better than existing models in the case of human detection in [17]. You Only Look Once (YOLO) method for object detection uses regression models instead of repurposing classifiers is developed in [18], this method outperforms DPM, R-CNN methods. İn [19] overall face detection, facial feature localization, and face comparison is carried out all at once. The authors of [20] have built a model to gather environmental parameters to build a smart campus environment. The parameters include air temperature, light intensity, and humidity. The paper also carries out an in-depth study on how to store real-time data in a standard and organized manner. The work in [21] is about principal component analysis (PCA) used for feature extraction that helps in achieving superior performance. The authors of the paper have worked to achieve a high recognition rate for IoT based image recognition. On analyzing all the pre-existing researches, it is evident that a robust model to detect all the facial features and head pose of a person with a mask all at once, does not exist. This has motivated us to develop a plentiful model which extracts all the features mentioned, from an image.

3 Proposed Work

3.1 Hardware Integration

To test the model's performance with different cameras and different lighting, we have used a raspberry pi camera, webcam, CCTV security camera and laptop webcam. The raspberry pi board can be integrated easily with Rpi camera and can also be used to transfer images from one computer to the other, in the same network using file transfer protocol (FTP). The raspberry pi has a very fast processing speed and capability to run long codes as compared to other boards. Using Rpi OS, the Raspberry pi provides an interface to work on and can be programmed using python, C, etc. The raspberry pi camera or a portable webcam, is interfaced with the raspberry pi. Using a python code, the images are sent periodically through file transfer protocol to the Jupyter notebook for image processing. In the case of the raspberry pi camera, picamera module is used to interface the camera with the raspberry pi. Imwrite function of OpenCV module is used to store the images on the RPI OS. File Transfer Protocol is a set of guidelines that controls how computers transfer data across the internet from one system to another. An FTP server is first set up, by connecting the raspberry pi and laptop to the same Wi-Fi network as shown in Fig. 1. ftplib library of python is used to send images through FTP. In the case of CCTV cameras, the TAPO camera serves Real-Time Streaming Protocol (RTSP). The protocol combines intricate programming, transcoding and client server method to send video through a

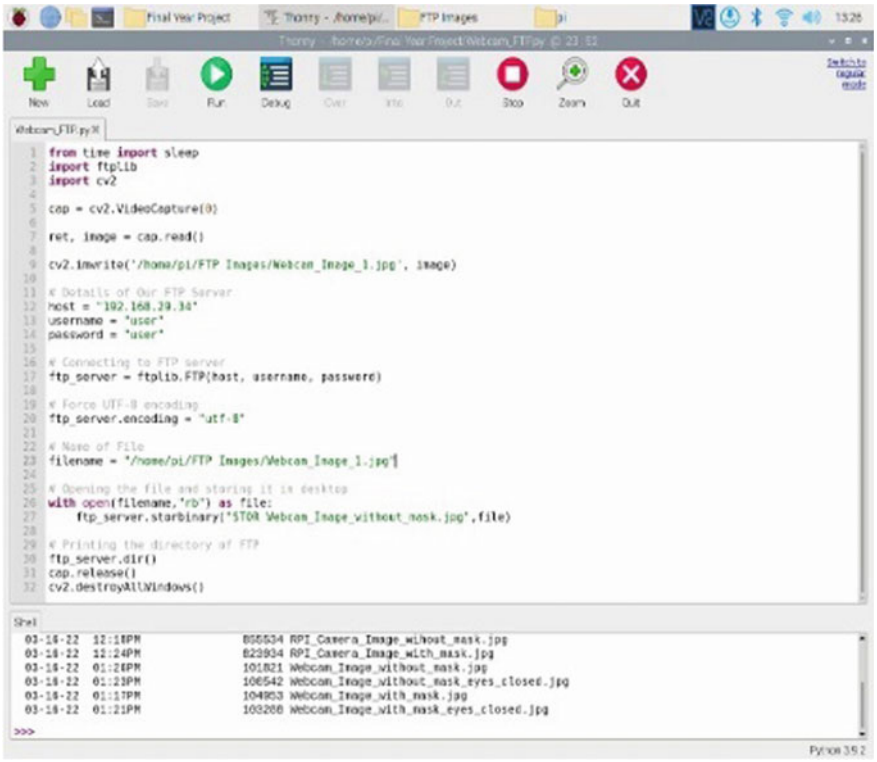

Fig. 1 Using FTP folder with webcam and RPI

network or to the internet using a link. Using this protocol, we have written a python code to display the stream on the Jupyter notebook. By analyzing the frames per second (FPS) we extract the frames periodically and save them in the OS using the OpenCV library. These images are then used for image processing and the results are obtained. In the case of a laptop webcam, images are captured and directly sent to the respective folder where images are extracted one at a time for image processing. The flowchart of working of hardware components of the model is shown in Fig. 2a–c.

3.2 Training the Model

Initially we import ImageDataGenerator, MobileNeyV2, AveragePooling2D, Dropout, Flatten, Dense, Input, Model, Adam, preprocess_input, img_to_array, load_img, to_categorical libraries from their respective Tensorflow.keras libraries. For preprocessing the dataset, we imported libraries from Sklearn. Other imported libraries included utils, matplotlib, NumPy, argparse and OS. Then we initialize

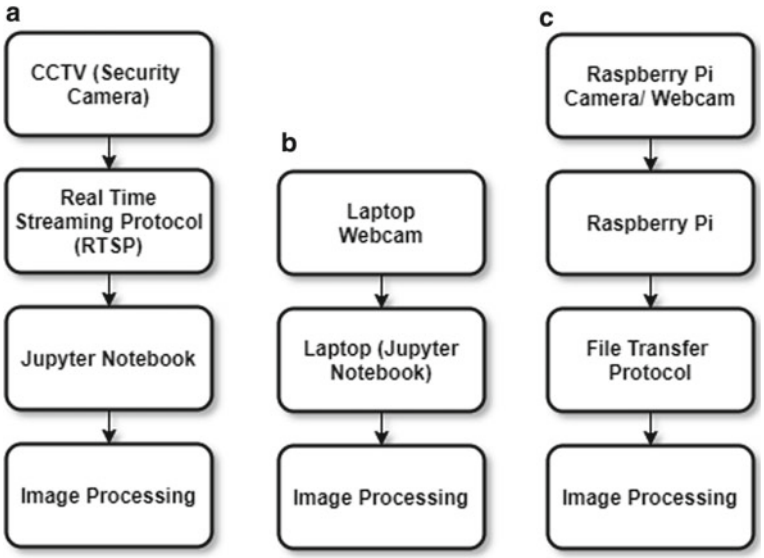

Fig. 2 **a** Flowchart of working of model with CCTV camera and Jupyter notebook, **b** flowchart of working of model with laptop webcam and Jupyter notebook, **c** flowchart of working of model with Rpi and Rpi camera

the initial learning rate, number of epochs to train for, and batch size which are determined through the hit and trial method. Once the dataset (with 3800 images) is imported, the model loops over all the images in the folder by simultaneously labelling them. Data and labels are converted into NumPy arrays and passed to a LabelBinarizer to fit the model. The dataset is split into train and test with 80% for training and 20% for testing. Data augmentation is also applied to increase the accuracy. After creating the base model with the help of MobileNetV2, head Model is made using the layers AveragePooling2D, Flatten, Dense, Dropout and again Dense layer. The head model is then placed on top of the base model. Finally, the model is fit with 20 epochs. The model is tested with the testing dataset and we get the classification report. Using this method, we have achieved an accuracy of 99% for our model.

3.3 Detect and Predict Mask

The extracted frames/images are sent as arguments to this function. FaceNet and MaskNet have defined weights for the convolutional neural network applied. Once the frame/image is obtained, first the height and weight are extracted and saved in shape. A blob image is made using the weights from faceNet. FaceNet contains a module, forward, that detects frontal faces, this is applied to the captured image. If

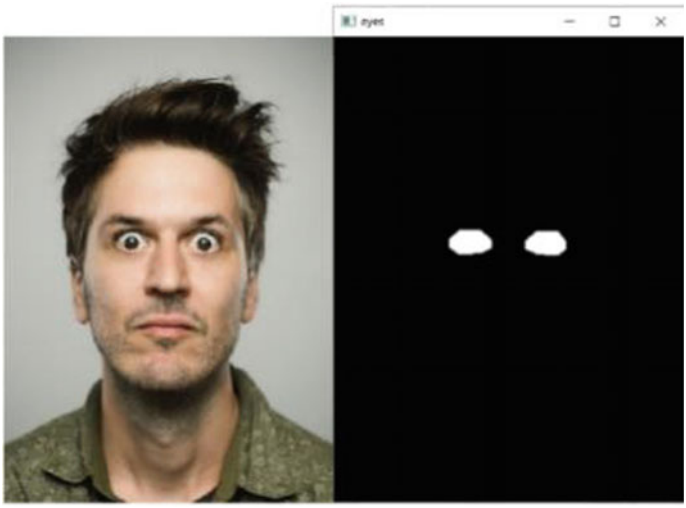

Fig. 3 Input image (left), output image (right), mask applied on the input image to detect eyes (white portion detecting eyes). *Source* Adapted from [21]

the confidence of the detected face is greater than 50%, the coordinates of the face are extracted and saved in locs. If a face has been detected, maskNet is used to detect the face with a mask on it. Once this is predicted, the coordinates are saved in preds. Ultimately a tuple of locs and preds is returned.

3.4 Eye on Mask

To detect the eyes on a face, we have applied a mask on the whole face except on the eyes. This helps to detect the eye location accurately. For this function, the predefined mask and the side of the eyes are sent as arguments. The location of the predefined eyes is saved in points, it is then converted into a NumPy array for processing. Once all the preprocessing is done, the fillConvexPoly () function is used to fill the face with a mask except for the eyes as shown in Fig. 3.

3.5 Eye Open or Closed

We have used the Eye Aspect Ratio (EAR) method to determine the status of the eye. In this method, once the eye location is obtained, the distance between the upper and lower eyelid is calculated. The points used to calculate EAR are shown in Fig. 4 (red points). This is compared with the standard value of eyes open and closed. It helps

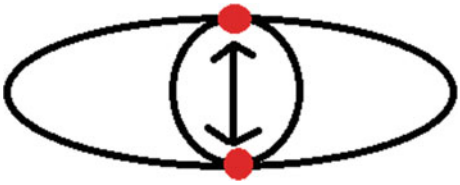

Fig. 4 Pictorial image of an eye marked red points used to calculate EAR

in determining if the eye is closed or opened. The standard values for a masked and unmasked face are different, the functions are defined respectively.

3.6 Eye and Eye Ball Contours

In the case of eye contours, the shape_68.dat is used to extract the coordinates of the eye. Once the coordinates are obtained it is sent as arguments to the eye contour function. The circle() function of OpenCV module takes five arguments, including the source picture, the (x, y) coordinates, the radius, the colour, and the thickness. The circle function draws a circle on the coordinates of the image with the mentioned radius and thickness. Obtained coordinates, radius = 2, respective color and thickness = 2 are passed to the function to draw the contours.

In the case of the eyeball contours, a threshold value is set (different for masked and unmasked faces), the midpoint of the eyeball is calculated and the threshold, midpoint and image are sent with right = False. Now, each eye is taken at a time and a mask is applied to them. Once the mask is applied, contours are detected. These contours are found by adjusting the threshold values sent. The contours detected are eyeballs. Obtained coordinates are marked using the circle function of OpenCV.

3.7 Head Pose of Person

The image obtained from hardware is sent as an argument to the function. The image is first resized to 1000 × 600 pixels and then flipped for processing. Then the image is converted to grayscale, a face mesh is created on the detected image. Using the defined weights, the landmarks of the face are detected. Once detected a loop is iterated on the coordinates. Using the nose coordinates and the other coordinates of the face, rotational and translational vectors are created using the function solvePnP of OpenCV. The rotational vector is then sent to rodrigues function of OpenCV to calculate rmat and jac. Rmat is then sent to RQDecomp 3 × 3 to get the angles (x, y, z) that are roll, pitch and yaw of the face. While testing the algorithm for different images, we found that the images form a pattern, when (x, y, z) is summed and compared. For different images, the summation of roll, pitch and yaw values had

different outputs. Hence the method is generalized and it became a distinguishing factor in the head pose estimation. Finally, the summation of the angles is returned.

3.8 Function Calling and Display of Output

On reading the image, it is converted into grayscale and locs and preds are found. We predict if the person has worn a mask or not. On detecting a mask, we find the locs and preds to create a rectangle in which we detect the eye contours and eyeball contours, eye status is found using Eye Aspect Ratio. Ultimately the head pose of the person is found. The output is displayed on the image. Then we display the final output image with all the labels. If the face is unmasked, we use the detector to find the rectangle surrounding the face. Then we find the eye status, eye contours, eyeball contours and head pose. The final output image is displayed with all the labels. The flowchart in Fig. 5 illustrates the process.

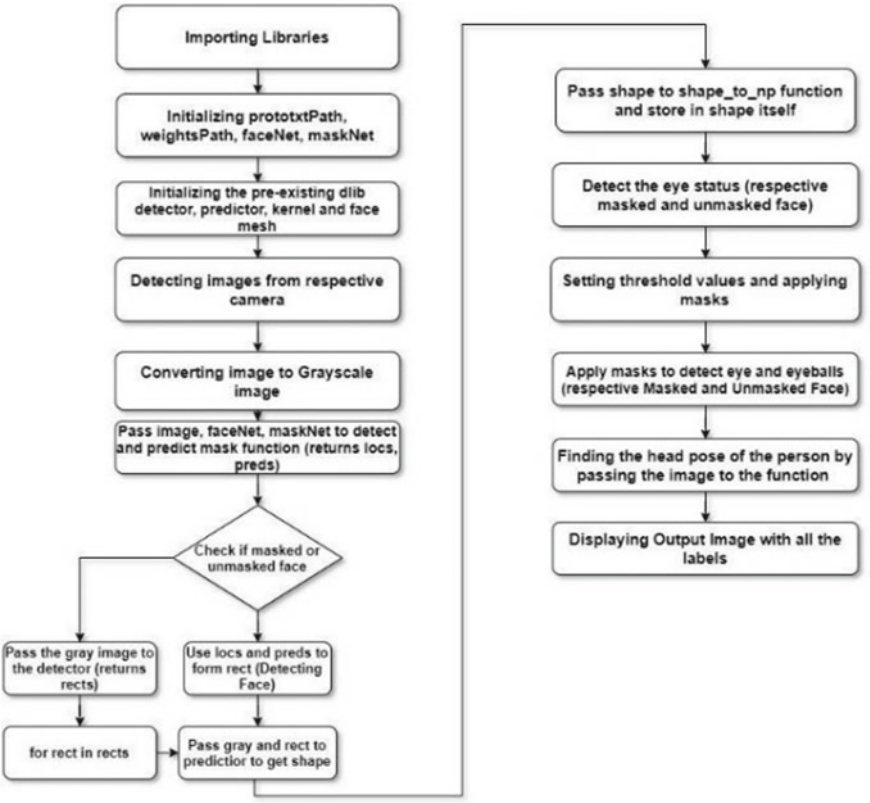

Fig. 5 Flowchart of application of the developed algorithm

4 Result Analysis

We used a laptop webcam for the input image. The input and output images are shown in Fig. 6.

Part 2: Interfacing raspberry pi camera and laptop using File Transfer Protocol (FTP). The results of image processing are shown in Fig. 7.

Part 3: Interfacing webcam, raspberry pi and laptop for the input image we received accurate results. The processed images are shown in Fig. 8.

Part 4: Interfacing Tapo camera directly with Jupyter notebook by extracting the stream using Real-Time Streaming Protocol (RTSP). We saved the frames of the stream at regular intervals. The images and their respective processed output images are shown in Fig. 9.

Part 5: Combining all parts, the head pose of the person is also displayed on the image as shown in Fig. 10.

The training loss and accuracy versus epoch plot are shown in Fig. 13. Loss and accuracy functions from the deep Learning model, training history, in keras is used to find the training loss and accuracy of the model. It is clearly evident that the accuracy of the model is 99% and the training loss of the model is 1%, when the number of epochs reaches 20. The epoch results with 20 epochs are shown in Fig. 14. The classification report with precision, recall, f1-score and support is shown in Fig. 15. The accuracy of the model increases and validation loss of the model decreases with

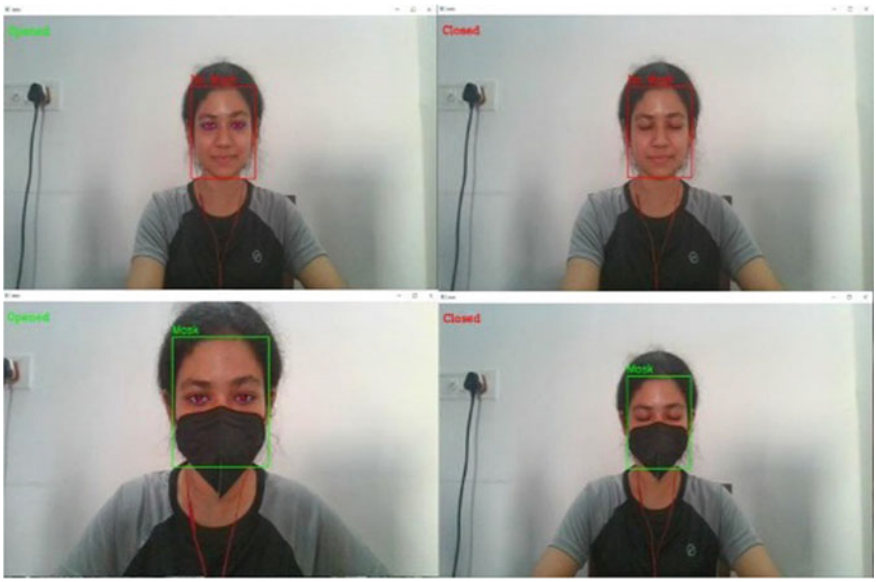

Fig. 6 Input image from a laptop webcam

Fig. 7 Using a raspberry pi camera to get input image for masked and unmasked image

Fig. 8 Using webcam and Rpi to get input image for masked image

Fig. 9 Using Tapo camera to get input image

Fig. 10 Final output images with all the features and head pose

increasing epochs. Once the model is trained, it is fit into the input images to get the output. The model achieves 99% accuracy with both mask images and unmask images. The obtained accuracy is better than other existing models.

The accuracy is calculated using the below formula:

$$\text{Accuracy} = \frac{True\ category1 + True\ category2}{True\ category1 + True\ category2 + False\ category1 + False\ category2}$$

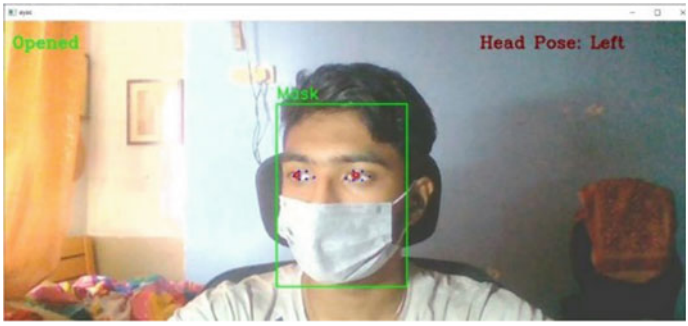

Fig. 11 Final output images with all the features and head pose (left)

Computationally, the training loss is calculated by taking the sum of errors for each example in the training set. The training loss is measured after each batch.

The results obtained are accurate and the model is working perfectly. It can be described as a robust model. The method of detecting each landmark one after the other has increased the performance of the model. The results from the laptop webcam are shown in Figs. 11 and 12. Here the image processing is done with a grayscale image, this increases the performance of the model too. The rotational vector calculated during the head pose estimation differs from one camera to another as the focal length of each camera is different. The model works best when the image processing is done at regular intervals, this reduces the processor requirements. Images are sent at regular intervals to the Jupyter notebook for image processing. Once the image processing is completed, the output image with labels is displayed. The model can be used to alert the alarm system if a person, wearing or not wearing a mask, is not attentive in a class or in a driving system. In a classroom, once the direction of the teacher or the blackboard is set, the model can be modified to detect if the pupil is attentive in the class or not. The major limitation of the model is that it cannot be used in very dim light conditions or at night, where there is minimum or no light. To overcome this problem, night vision cameras can be incorporated, to get the input stream, which can be processed by the model to get the desired results. Another limitation is the hardware interconnections between the camera and the raspberry are prone to wear and tear, and have to be handled with care for accurate results and smooth processing.

5 Conclusion

We have achieved a maximum accuracy of 99% and our model is working better than the pre-existing models. We have also worked on head pose estimation and we achieved better accuracy and precision than the existing solutions. Our model is best

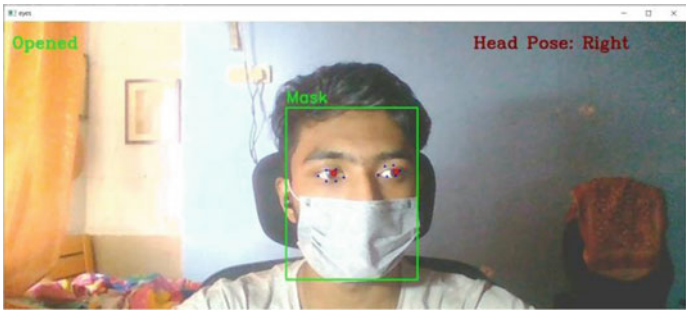

Fig. 12 Final output images with all the features and head pose (right)

Fig. 13 Training loss and accuracy versus epoch plot

```
Epoch 12/20
95/95 [==============================] - 99s 1s/step - loss: 0.0369 - accuracy: 0.9888 - val_loss: 0.0280 - val_accuracy: 0.990
9
Epoch 13/20
95/95 [==============================] - 97s 1s/step - loss: 0.0337 - accuracy: 0.9908 - val_loss: 0.0280 - val_accuracy: 0.990
9
Epoch 14/20
95/95 [==============================] - 100s 1s/step - loss: 0.0414 - accuracy: 0.9878 - val_loss: 0.0259 - val_accuracy: 0.99
22
Epoch 15/20
95/95 [==============================] - 97s 1s/step - loss: 0.0308 - accuracy: 0.9911 - val_loss: 0.0246 - val_accuracy: 0.992
2
Epoch 16/20
95/95 [==============================] - 100s 1s/step - loss: 0.0341 - accuracy: 0.9904 - val_loss: 0.0280 - val_accuracy: 0.99
09
Epoch 17/20
95/95 [==============================] - 98s 1s/step - loss: 0.0317 - accuracy: 0.9911 - val_loss: 0.0243 - val_accuracy: 0.992
2
Epoch 18/20
95/95 [==============================] - 101s 1s/step - loss: 0.0308 - accuracy: 0.9918 - val_loss: 0.0226 - val_accuracy: 0.99
48
Epoch 19/20
95/95 [==============================] - 100s 1s/step - loss: 0.0298 - accuracy: 0.9904 - val_loss: 0.0280 - val_accuracy: 0.99
22
Epoch 20/20
95/95 [==============================] - 98s 1s/step - loss: 0.0277 - accuracy: 0.9911 - val_loss: 0.0260 - val_accuracy: 0.992
2
```

Fig. 14 Epoch results of the trained model

Fig. 15 Classification report of fit model

```
[INFO] evaluating network...
              precision    recall  f1-score   support

   with_mask       0.99      0.99      0.99       383
without_mask       0.99      0.99      0.99       384

    accuracy                           0.99       767
   macro avg       0.99      0.99      0.99       767
weighted avg       0.99      0.99      0.99       767
```

fit in real-time applications and can be used in different scenarios in driving environments, schools, colleges, and offline proctored exams to examine the attentiveness of a person accurately. The model can be trained and modified to get appropriate results in the night too. Night vision cameras can be incorporated in the model to get images in the night. The night vision cameras could be fit inside a car, where the lighting conditions are very less or negligible, to check the status, eye status and head pose, of a masked driver. In case of any abnormality in the driver status or negligence in driving an alarm could be triggered to aware about a possible accident. Our developed model outperforms the pre-existing models, faster_RCNN, inceptionV2 structure, fatigue detection convolutional network (FDCN), Histograms of Oriented Gradient (HOG), of image processing to detect if a person has worn a mask or not or to detect facial features without a mask. The achieved accuracy of the model makes it robust and fit for all lighting conditions and angles.

References

1. Kong X et al (2021) Real-time mask identification for COVID-19: an edge-computing-based deep learning framework. IEEE Internet of Things J 8(21):15929–15938. https://doi.org/10.1109/JIOT.2021.3051844
2. Ge S, Li J, Ye Q, Luo Z (2017) Detecting masked faces in the wild with LLE-CNNs. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 426–434. https://doi.org/10.1109/CVPR.2017.53
3. Sanoob AH, Roselin J, Latha P (2016) Smartphone enabled intelligent surveillance system. IEEE Sens J 16(5):1361–1367. https://doi.org/10.1109/JSEN.2015.2501407
4. Din NU, Javed K, Bae S, Yi J (2020) A novel GAN-based network for unmasking of masked face. IEEE Access 8:44276–44287. https://doi.org/10.1109/ACCESS.2020.2977386
5. Zhang J, Han F, Chun Y, Chen W (2021) A novel detection framework about conditions of wearing face mask for helping control the spread of COVID-19. IEEE Access 9:42975–42984. https://doi.org/10.1109/ACCESS.2021.3066538
6. Li S et al. (2020) Multi-angle head pose classification when wearing the mask for face recognition under the COVID-19 coronavirus epidemic. In: 2020 International conference on high performance big data and intelligent systems (HPBD&IS), pp 1–5. https://doi.org/10.1109/HPBDIS49115.2020.9130585
7. Fan X, Jiang M, Yan H (2021) A deep learning based light-weight face mask detector with residual context attention and Gaussian heatmap to fight against COVID-19. IEEE Access 9:96964–96974. https://doi.org/10.1109/ACCESS.2021.3095191

8. Wang B, Zhao Y, Chen CLP (2021) Hybrid transfer learning and broad learning system for wearing mask detection in the COVID-19 Era. IEEE Trans Instrum Meas 70:1–12, Art no. 5009612. https://doi.org/10.1109/TIM.2021.3069844
9. Wu B-F, Chen B-R, Hsu C-F (2021) Design of a facial landmark detection system using a dynamic optical flow approach. IEEE Access 9:68737–68745. https://doi.org/10.1109/ACCESS.2021.3077479
10. Ling Y, Luo R, Dong X, Weng X (2021) Driver Eye location and state estimation based on a robust model and data augmentation. IEEE Access 9:67219–67231. https://doi.org/10.1109/ACCESS.2021.3076365
11. Zhang F, Su J, Geng L, Xiao Z (2017) Driver fatigue detection based on eye state recognition. In: Proceedings of the international conference on machine vision and information technology (CMVIT), pp 105–110
12. Huang R, Wang Y, Guo L (2018) P-FDCN based eye state analysis for fatigue detection. In: Proceedings of the IEEE 18th international conference on communication technology (ICCT), pp 1174–1178
13. Deng W, Wu R (2019) Real-time driver-drowsiness detection system using facial features. IEEE Access 7:118727–118738
14. Sun Y, Yan P, Li Z, Zou J, Hong D (2020) Driver fatigue detection system based on colored and infrared eye features fusion. Comput Mater Continua 63(3):1563–1574
15. Cheng Q, Wang W, Jiang X, Hou S, Qin Y (2019) Assessment of driver mental fatigue using facial landmarks. IEEE Access 7:150423–150434
16. Colak ME, Varol A (2020) Easymatch-an eye localization method for frontal face images using facial landmarks. Tehnički Vjesnik 27(1):205–212
17. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 886–893
18. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 779–788
19. Colmenarez AJ, Huang TS (1998) Face detection and recognition. In: Face recognition. Springer, Berlin, Heidelberg, pp 174–185
20. Sungheetha A (2021) Assimilation of IoT sensors for data visualization in a smart campus environment. J Ubiquit Comput Commun Technol 3(4):241
21. Jeena JI, Darney PE (2021) Design of deep learning algorithm for IoT application by image based recognition. J ISMAC 3(03):276–290