B. Tech Third Year
Course: Foundation of Data Science (FDS)
Course Code: IT 3101

"FACE MASK DETECTION ALERT USING PYTHON"

by

Name of the Student
Gautam Vhavle

(219302198)

IT 5C

Under the guidance

of

Name of the mentor
Mr. Dhananjay Singh Sir
Manipal University Jaipur, Jaipur (Raj.).

Designation of mentor
Assistant Professor
Department of Information Technology

Manipal University Jaipur, India

NOV,2023

Certificate

This is to certify that the project titled **"Face Mask Detection for Safety in Public Areas "** is a record of the bona fide work done by **GAUTAM VHAVLE (219302225**) submitted for the fulfilment of the requirements for the completion of the Foundation of Data Science (IT 3101) course in the Department of Computer and Computer Engineering of Manipal University Jaipur, during the academic session JAN TO MAY 2023.

Signature of the mentor

Name of the Mentor
Designation of the mentor
Department of _____

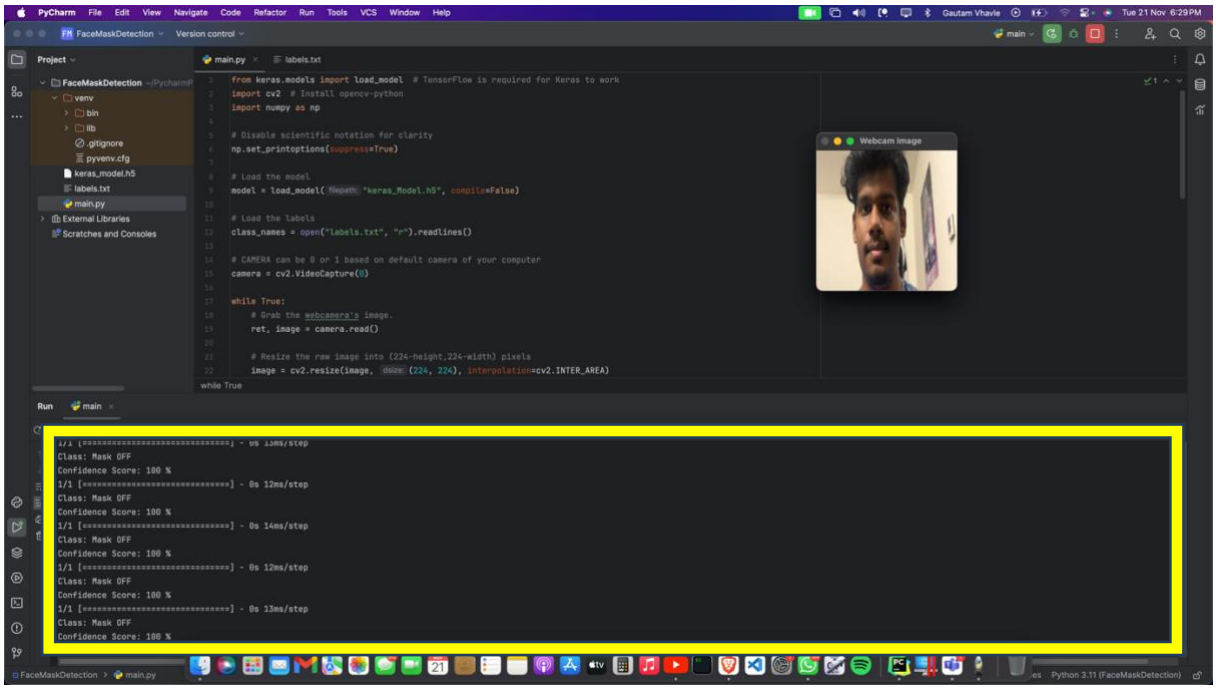
Signature of the HoD

Name of the HoD
Head of the Department
Department of _____

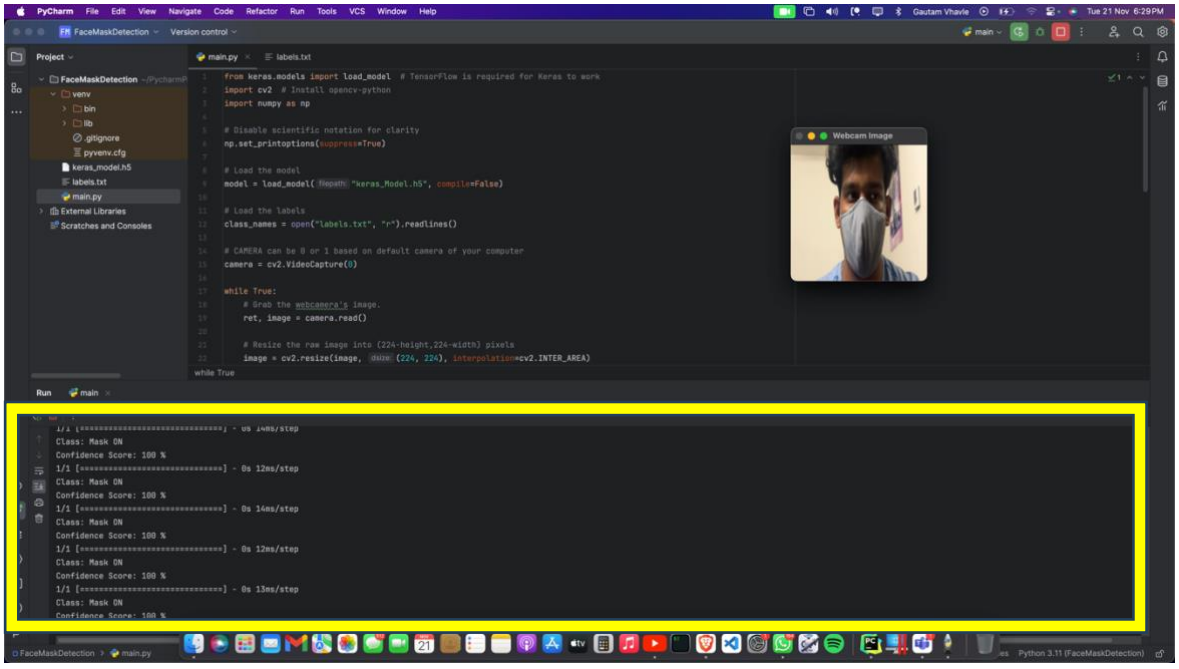
Abstract

The "Face Mask Detection in Public Places" project addresses the critical need for automated monitoring of face mask compliance in public spaces, contributing to the broader effort of controlling the spread of infectious diseases. Employing advanced computer vision and machine learning techniques, our system is designed to detect the presence or absence of face masks on individuals in real-time. The methodology involves the utilization of a deep learning model trained on a diverse dataset of facial images with and without masks. The system, implemented on a robust hardware platform, analyses live video streams from surveillance cameras in public areas. When a lack of face mask compliance is identified, the system triggers an alert mechanism, notifying relevant authorities or individuals for prompt action.

Working Model:



Working Model Unmasked.



Working Model masked.

Masked Dataset used for training.

Unmasked humans Dataset used for training.

Algorithm

1. **libraries and Modules:**
    - The algorithm utilizes several Python libraries and modules, including Keras for deep learning, OpenCV (**cv2**) for computer vision tasks, and NumPy for numerical operations.
2. **Loading the Pre-trained Model:**
    - The Keras model for face mask detection is loaded using the **load_model** function. The model is stored in a file named "keras_Model.h5". The **compile=False** argument is used to load the model without compiling, as the model is already trained.
3. **Loading Labels:**
    - The algorithm reads class labels from a file named "labels.txt". These labels likely correspond to the classes output by the model, indicating whether a face mask is present or absent.
4. **Webcam Initialization:**
    - The algorithm initializes the webcam using OpenCV's **VideoCapture** class. The **camera.read()** method captures each frame from the webcam.
5. **Image Preprocessing:**
    - Each captured frame is resized to 224x224 pixels, which is a common input size for many deep learning models. The interpolation method used is **cv2.INTER_AREA**.
6. **Image Normalization:**
    - The image is converted to a NumPy array and normalized to be suitable for input to the deep learning model. Normalization is performed by dividing each pixel value by 127.5 and subtracting 1.
7. **Model Prediction:**
    - The pre-trained model predicts whether a face mask is present in the processed image using the **model.predict** method. The class with the highest probability is selected as the predicted class.
8. **Displaying the Webcam Feed:**

- The original webcam feed is displayed in a window using **cv2.imshow**.

9. **Outputting Predictions:**
   - The predicted class and confidence score are printed to the console. The class name is obtained from the loaded labels, and the confidence score is calculated as a percentage.
10. **Keyboard Input Handling:**
    - The algorithm listens for keyboard input using **cv2.waitKey(1)**. If the ESC key (ASCII code 27) is pressed, the loop breaks, and the webcam is released.
11. **Cleanup:**
    - After the loop, the webcam is released using **camera.release()**, and all OpenCV windows are closed with **cv2.destroyAllWindows()**.

Methodology / CODE

```
from keras.models import load_model  # TensorFlow is required for Keras
to work
import cv2  # Install opencv-python
import numpy as np

# Disable scientific notation for clarity
np.set_printoptions(suppress=True)

# Load the model
model = load_model("keras_Model.h5", compile=False)

# Load the labels
class_names = open("labels.txt", "r").readlines()

# CAMERA can be 0 or 1 based on default camera of your computer
camera = cv2.VideoCapture(0)

while True:
    # Grab the webcam's image.
    ret, image = camera.read()

    # Resize the raw image into (224-height,224-width) pixels
    image = cv2.resize(image, (224, 224), interpolation=cv2.INTER_AREA)
```

```
    # Show the image in a window
    cv2.imshow("Webcam Image", image)

    # Make the image a numpy array and reshape it to the models input
shape.
    image = np.asarray(image, dtype=np.float32).reshape(1, 224, 224, 3)

    # Normalize the image array
    image = (image / 127.5) - 1

    # Predicts the model
    prediction = model.predict(image)
    index = np.argmax(prediction)
    class_name = class_names[index]
    confidence_score = prediction[0][index]

    # Print prediction and confidence score
    print("Class:", class_name[2:], end="")
    print("Confidence Score:", str(np.round(confidence_score * 100))[:-
2], "%")

    # Listen to the keyboard for presses.
    keyboard_input = cv2.waitKey(1)

    # 27 is the ASCII for the esc key on your keyboard.
    if keyboard_input == 27:
        break

camera.release()
cv2.destroyAllWindows()
```

Sample Output:

Check out the sample output of the code.

```
Class: Mask OFF
Confidence Score: 93 %
1/1 [==============================] - 0s 15ms/step
Class: Mask OFF
Confidence Score: 90 %
1/1 [==============================] - 0s 12ms/step
Class: Mask ON
Confidence Score: 99 %
1/1 [==============================] - 0s 22ms/step
Class: Mask ON
Confidence Score: 98 %
1/1 [==============================] - 0s 16ms/step
```

Learnings

Deep Learning Model Loading:
Understanding how to load a pre-trained deep learning model using a library like Keras (load_model function).

Real-time Image Processing:
Working with real-time image data from a webcam using OpenCV (cv2.VideoCapture and camera.read()).

Image Preprocessing:
Resizing images to a specific input size suitable for the model and understanding the importance of preprocessing steps for neural networks.

Normalization of Image Data:
Normalizing pixel values to a range suitable for neural network input (in this case, scaling pixel values to be between -1 and 1).

Model Prediction:
Using a pre-trained model to make predictions on new data (model.predict).

Interpreting Model Output:
Understanding how to interpret the output of a classification model, including extracting class names and confidence scores.

Real-time User Interaction:
Listening for keyboard input to control the execution of the program (breaking the loop with the ESC key).

Webcam Handling:
Initialization and release of the webcam using OpenCV.

Application of Deep Learning to Real-world Problems:
Applying deep learning techniques to address a specific problem, in this case, detecting face masks in real-time.

Understanding Computer Vision Basics:
Basic computer vision concepts, such as capturing and processing video frames from a webcam.

Conclusions

In summary, the face mask detection algorithm demonstrates practical applications of deep learning and computer vision for real-time monitoring. By leveraging a pre-trained model, the system effectively detects face masks from webcam input. The project's simplicity and relevance to public health underscore its potential for broader applications in creating safer public spaces**.**

Prospects

Enhanced Accuracy and Robustness:

Fine-tuning the model for improved accuracy and robustness across diverse demographics and environmental conditions.

Integration with Smart Surveillance Systems:

Collaborating with smart surveillance systems for real-time monitoring in various public spaces.

Mobile Application Integration:

Developing a mobile application for individual alerts and reminders, fostering personal responsibility in public health.

Privacy-aware Solutions:

Implementing privacy-aware features to address concerns related to surveillance and data collection.

Global Deployment and Collaboration:

Scaling the algorithm for global deployment and collaborating with health authorities for broader public health initiatives.

References

- OpenCV
- Numpy
- Keras
- Python

Acknowledgements

We would like to express our special thanks of gratitude to our teacher Mr. **Dhananjay Singh Sir** who gave us the golden opportunity to do this wonderful project on the topic Palm Recognition using python language which also helped us in doing a lot of research and we came to know about so many new things I am thankful to them.