PROJECT REPORT

for

Face Mask Detection Using Computer Vision

Prepared by

Aswin Sreekumar	**B200737CS**
Anuram Anil	**B200713CS**
Rithika Kathirvel	**B200059CS**

Course Faculty: Dr Santosh Kumar Behera

Project Lead: Aswin Sreekumar

Team Number: 12

Course: CS4046D Computer Vision

Date: 01-11-2023

Signatue of Team Members

Contents

Revisions

Version	Project Report by	Date Completed
1.0	Aswin Sreekumar, Anuram Anil, Rithika Kathirvel	30-10-2023

1 Introduction

In our increasingly health-conscious world, the importance of public health and safety has never been more evident. The persistent challenges posed by contagious diseases like COVID-19 and Nipah have underscored the vital role of preventative measures, particularly the widespread use of face masks. With this in mind, our project, "Face Mask Detection using Computer Vision," aims to create a powerful system capable of detecting the presence of face masks on individuals' faces.

Our team's primary objective is to develop a robust and efficient solution employing cutting-edge computer vision techniques to accurately identify and categorize individuals based on their adherence to face mask usage. This system holds vast potential, ranging from monitoring compliance with mask mandates in public spaces to enhancing safety measures in various contexts. To achieve this, we are working towards constructing a machine-learning model trained on a diverse dataset of people both wearing and not wearing masks. Our system operates by detecting faces within images and subsequently running the model to determine whether the person in the image is wearing a mask or not. Additionally, we are committed to providing a user-friendly experience through the implementation of a graphical user interface (UI).

This document is structured to cover key aspects of our project, including objectives, methodology, implementation details, results, conclusions, references, and acknowledgments. By embarking on this initiative, we aim to contribute to the broader mission of safeguarding public health and safety through innovative technological solutions.

2 Objective

Our team's central objective is to create a robust machine-learning model capable of confidently discerning whether a person is wearing a mask or not. This endeavor involves training a face mask detection model, leveraging MobileNetV2 as our base model. To accomplish this mission, we have chosen to harness the capabilities of the Keras deep learning library, ensuring efficiency and effectiveness in model development and training. In pursuit of a comprehensive solution, we are designing a driver code that will analyze images and make crucial determinations. This code seamlessly integrates with the OpenCV library, utilizing a pre-trained Haar Cascade classifier to perform face detection. If a face is detected within the image, our custom model steps in to classify whether the individual is wearing a mask or not. This two-step process ensures the system's accuracy and reliability in recognizing mask-wearing behaviors. Moreover, we aim to provide a seamless user experience by implementing a user-friendly graphical user interface (UI) that delivers the model's results. In summary, our team's mission revolves around building a face mask detection system that combines machine learning, and intuitive user interaction.

4 Methodology

Step 1: Develop a machine-learning model capable of confidently discerning whether a person is wearing a mask or not.

We coded a python script for building and training a face mask detection model using transfer learning with MobileNetV2 as the base model taking advantage of the Keras deep learning library. MobileNetV2 is a deep neural network architecture that is specifically designed to address the need for lightweight and efficient neural networks for tasks like image classification, object detection, and more, which can be deployed on resource-constrained devices. MobileNetV2 has already been pre-trained on a large dataset, typically for image classification tasks. By using MobileNetV2 as the base model, we are taking the pre-trained MobileNetV2 model and building additional layers on top of it to adapt it to the specific task, which, in this case, is face mask detection. Keras simplifies the process of building, training, and evaluating deep learning models by providing a user-friendly and modular approach. The team used the Keras deep learning library to build and train a new model that extends and customizes MobileNetV2 by adding layers specifically designed for the face mask detection task. This approach is efficient because it takes advantage of the knowledge already captured by MobileNetV2, saving a lot of time and computational resources that would be required to train a model from scratch. It allows you to adapt a pre-existing model for a different but related task. The model was developed as follows:

1. Import the necessary libraries and modules:
    a. os for operating system-related functions.
    b. numpy for numerical operations.
    c. matplotlib for data visualization.
    d. keras.utils for loading and preprocessing images.
    e. keras.applications.MobileNetV2 for using the MobileNetV2 architecture.
    f. imutils.paths for handling image paths.
    g. sklearn for machine learning tools.
    h. ImageDataGenerator for data augmentation.
    i. Various Keras layers, models, optimizers, and metrics.
2. Set up hyperparameters:
    a. INIT_LR for the initial learning rate.
    b. EPOCHS for the number of training epochs.
    c. BS for the batch size.
3. Load and preprocess image data:
    a. Load images from a dataset directory and store them in the data list.
    b. Extract labels from the image file paths and store them in the labels list.
    c. Preprocess the images (resize to 224x224 pixels and apply preprocessing function from MobileNetV2).
    d. Convert data and labels to NumPy arrays.
4. Perform label binarization and one-hot encoding for classification.
5. Split the data into training and testing sets.
6. Create an image data generator for data augmentation, which helps prevent overfitting.
7. Build the MobileNetV2 base model with pre-trained weights, and add custom layers for classification.
8. Set the MobileNetV2 layers as non-trainable, so only the custom layers will be trained.
9. Compile the model with binary cross-entropy loss and the Adam optimizer. The Adam optimizer is a widely used optimization and is known for its fast convergence and good performance on a variety of deep learning tasks. Both binary cross-entropy loss and the Adam optimizer are computationally efficient. This is crucial when dealing with large datasets. Efficiency in training reduces the time required to train a model and minimizes the computational resources needed.
10. Train the model using the training data with data augmentation and validate it using the testing data.
11. Evaluate the model's performance and print a classification report.
12. Save the trained model to a file.

13. Plot and save a graph of training loss and accuracy over epochs. The code will save the trained model as "mask.model" and the training plot as "plot.png" in your current working directory.

Step 2: Create a face detection script to detect faces on an image.

We implemented a Python script using the OpenCV library for face detection. This script uses a pre-trained Haar Cascade classifier to detect faces in images. Haar Cascade classifier is a machine learning object detection method used to identify objects in images or video. It was developed by Viola and Jones and is based on Haar-like features, which are simple rectangular filters that can be applied to an image. If a face is detected, the classifier returns the coordinates of the detected face, allowing you to draw a bounding box around it or perform additional tasks, such as analyzing whether the detected face is wearing a mask. Face detection script was coded as follows:

1. Import the OpenCV library using import cv2
2. Load the Haar Cascade Classifier for face detection.
   haar_model=cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml').
   This line loads the pre-trained Haar Cascade classifier for detecting frontal faces. The cv2.data.haarcascades provides the path to the directory containing the pre-trained Haar Cascade XML files.
3. Define a function called detect_face that takes an input frame (an image) as its parameter.
   a. Convert the input frame to grayscale. Converting the image to grayscale is often done to simplify the image and improve the efficiency of face detection.
   b. Use the Haar Cascade classifier to detect faces in the grayscale image: The detectMultiScale function detects faces in the grayscale image. It returns a list of rectangles (x, y, width, height) where faces are detected.
   c. check the number of detected faces: If only one face is detected, it draws a blue rectangle around the face, extracts the cropped face, and displays it in a separate window called 'Cropped Face'. It returns 1. If no faces are detected, it returns 0. If more than one face is detected, it returns -1.

Step 3: Develop a test code to use the previously built pre-trained face mask detection model and predict whether a mask is detected in a given image. It is done as follows:

1. Import necessary libraries and scripts.
   a. keras.models for loading the pre-trained model.
   b. keras.applications.mobilenet_v2 for preprocessing input images.
   c. numpy for numerical operations.
   d. cv2 for image processing.
   e. face detection script.
2. Load the pre-trained face mask detection model using Keras and specify a confidence threshold for making predictions.
3. Define a function predict_mask(image_file) that takes an image file as input and performs the following steps:
   a. Read the image from the specified file.
   b. Using the previously developed face.detect_face function to detect faces in the image. The function returns output based on whether no face, one face, or multiple faces are found.
   c. If no face is found, it returns "No face found." If multiple faces are found, it returns "Multiple faces found."

d. if one face is found, resize and preprocess the image to match the input size and format expected by the model (224x224 pixels, RGB format).
e. Make a prediction using the loaded model. The model predicts whether a mask is detected in the image.
f. Check the prediction confidence, and based on the confidence threshold, return "Mask detected" or "Mask not detected."

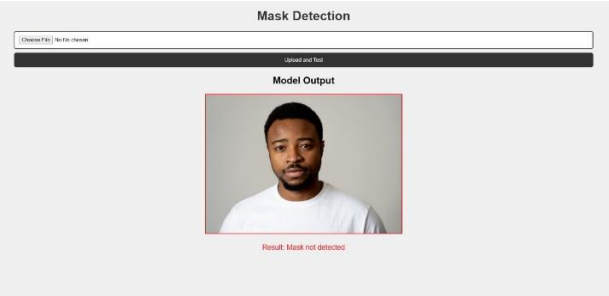
Step 4: Create a Flask web application that allows users to upload an image and then uses the predict_mask function in the driver code to predict whether a mask is detected in the uploaded image.

This Flask web application allows users to interact with the face mask detection system built, providing an easy and user-friendly way to upload images and view predictions about whether masks are detected or not detected in the images.

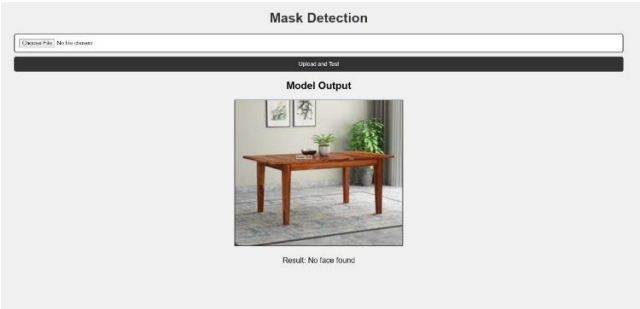
5 Result

The output as predicted by the model for different images is presented below:

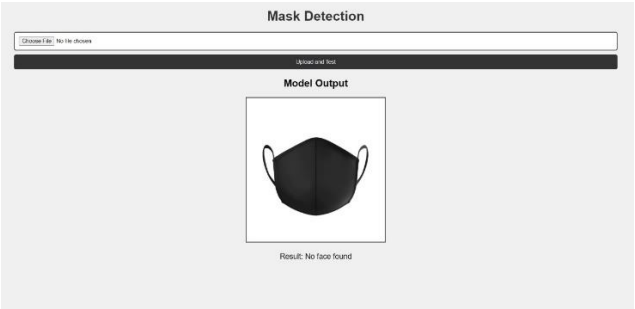
Case 1: Face mask detected



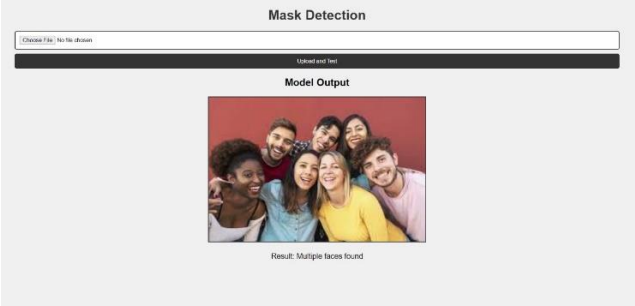
Case 2: Face mask not detected



Case 3: No face found



Case 4: No face found



Case 5: Multiple faces found

Case 6: Slightly titled face with mask

6 Conclusion

The purpose of this project is to implement a face mask detection system that predicts if a face is wearing a mask or not with a good confidence level. By machine learning concepts and a well-balanced blend of the Keras deep learning library and OpenCV, the project has successfully constructed a highly capable model for face mask detection. The incorporation of the MobileNetV2 architecture as the base model and the use of binary cross-entropy loss with the Adam optimizer have elevated the project's effectiveness and efficiency. The results obtained from the trained model have demonstrated its ability to make confident predictions, thus serving as a valuable tool for enhancing public health and safety. The development of a user-friendly graphical user interface (UI) further accentuates the project's significance. This UI not only streamlines the process of image analysis but also makes the system accessible to a broader audience. The team demonstrated an understanding of machine learning concepts by using computer vision and neural network tools to complete this project successfully.

8 References and Acknowledgements

1. Viola, P., & Jones, M. (2004). Robust real-time face detection. International Journal of Computer Vision, 57(2), 137-154.
2. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (pp. 4510-4520).
3. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
4. Official MobileNetV2 GitHub Repository: You can find the official code and more information about MobileNetV2 in the repository maintained by the authors: https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet
5. Blog Posts and youtube Tutorials: Various blog posts and tutorials are available online that explain MobileNetV2 and how to use it for different applications. You can find these resources on platforms like Medium, youtube and Towards Data Science.
6. Keras Official Documentation: The official documentation for Keras provides a wealth of information, including tutorials, guides, and examples for deep learning tasks, including face mask detection. https://keras.io/
7. Kaggle: Kaggle is a popular platform for data science and machine learning. It offers a wide range of datasets, notebooks, and competitions related to face mask detection and deep learning using Keras. https://www.kaggle.com/c/ml-fmi-23-2020/overview

Appendix A - Activity Log

Serial Number	Meeting Date	Meeting time	Description
1	17-10-2023	6 pm - 7 pm	Discussed the requirements over a google meet through an hour-long session.
2	25-10-2023	8 pm - 10 pm	The group met offline, and the progress of the project was discussed.
3	27-10-2023	6 pm - 9 pm	In the online coding and brainstorming session, the team coded together, and the model along with the driver code was completed.
4	29-10-2023	3 pm – 5 pm	The UI was completed and the face detection system was tested successfully
5	31-10-2023	3 pm – 6pm	The team met offline to finalize the project report and presentation

Serial Number	Team Member Name	Contribution
1	Aswin Sreekumar	Design, building and training of the mask detection model and the presentation.
2	Anuram Anil	The User interface for the face detection system and the presentation.
3	Rithika Kathirvel	Face detection system along with driver code and project report.