Face Mask Detection System: Project Report

Page 1

1. Introduction

The global COVID-19 pandemic highlighted the critical importance of public health measures, with wearing face masks being one of the most effective ways to curb the spread of the virus. Enforcing mask-wearing policies in public spaces, however, presents a significant challenge. Manual monitoring is often impractical and resource-intensive. To address this, we have developed an automated Face Mask Detection system. This project leverages computer vision and deep learning techniques to create a system capable of identifying whether individuals in a video feed are wearing a face mask. The system processes video in real-time, detects faces, and classifies them as "Mask" or "No Mask," providing an efficient and scalable solution for monitoring compliance with public health guidelines. An auditory alert feature is also integrated to notify supervisors when a person without a mask is detected.

2. Abstract

This report outlines the development of a real-time Face Mask Detection system built using Python, OpenCV, and the TensorFlow/Keras deep learning framework. The project is divided into three primary stages: dataset preparation, model training, and real-time deployment. Initially, a dataset of images with and without masks was preprocessed by extracting facial regions of interest (ROIs) based on XML annotations. Subsequently, a Convolutional Neural Network (CNN) was designed and trained on this prepared dataset to learn the features distinguishing masked faces from unmasked ones. The trained model was then saved and integrated into a real-time detection application. This final application utilizes a Haar Cascade classifier to locate faces in a live video stream and employs the trained CNN to classify each detected face. The system visually indicates the status with colored bounding boxes and text labels ("Mask" or "No Mask") and triggers an audible alert when a mask is not detected, offering a practical tool for automated public health monitoring.

3. Tools Used

The project was developed using a combination of programming languages, libraries, and frameworks. The core components include:

- **Programming Language:** Python
- **Deep Learning Framework:** TensorFlow with Keras API
- **Computer Vision Library:** OpenCV
- **Numerical Operations:** NumPy
- **File/Path Management:** os, imutils
- **Data Preprocessing:** xml.etree.ElementTree
- **Machine Learning Utilities:** Scikit-learn (for data splitting and label binarization)
- **Audio Handling:** Pygame (for the alert system)

4. Steps Involved in Building the Project

The construction of the Face Mask Detection system followed a systematic, multi-stage process, as detailed below.

Step 1: Dataset Preparation (`prepare_dataset.py`)

The first crucial step was to prepare a clean and structured dataset for training the model.

- **Data Sourcing:** The initial dataset consisted of images and corresponding XML annotation files. Each XML file contained the coordinates for bounding boxes around faces and a label indicating if the person was `with_mask` or `without_mask`.
- **Face Extraction:** The `prepare_dataset.py` script was executed to automate the preprocessing. It parsed each XML annotation file to retrieve the bounding box coordinates for every face.
- **Region of Interest (ROI) Cropping:** Using these coordinates, the script cropped the face region from the corresponding image. This isolates the most relevant features for the model to learn from.
- **Categorization:** The extracted face ROIs were then saved into two separate directories: `preprocesssed_dataset/with_mask` and `preprocesssed_dataset/without_mask`. This created a well-organized image dataset ready for the training phase.

Step 2: Model Training (`train_model.py`)

With the preprocessed data, the next step was to build and train a deep learning model.

- **Data Loading:** The `train_model.py` script loaded the images from the two categories, resizing them to a uniform size (128x128 pixels) and converting them to NumPy arrays. Labels were assigned based on the folder name.
- **CNN Architecture:** A Convolutional Neural Network (CNN) was constructed using the Keras Sequential API. The architecture consisted of:
  - Two `Conv2D` layers with `ReLU` activation for feature extraction.
  - Two `MaxPooling2D` layers to reduce spatial dimensions.
  - A `Flatten` layer to convert the 2D feature maps into a 1D vector.
  - A `Dense` (fully connected) layer with `ReLU` activation.
  - A final `Dense` output layer with `softmax` activation for binary classification (Mask/No Mask).
- **Training Process:** The dataset was split into training (80%) and testing (20%) sets. The model was compiled with the `Adam` optimizer and `categorical_crossentropy` loss function. It was then trained on the training data for a set number of epochs.
- **Model Saving:** After training, the model's architecture and learned weights were saved to a single file, `face_mask_detector_model.h5`, for later use in the real-time detection application.

Step 3: Real-time Detection & Deployment (`detect_mask_video.py`)

The final stage involved using the trained model to perform predictions on a live video stream.

- **Initialization:** The script `detect_mask_video.py` begins by loading the pre-trained `face_mask_detector_model.h5` model and a Haar Cascade classifier for frontal face detection. It also initializes the video stream from the webcam and sets up the Pygame mixer for the alert sound.
- **Real-time Loop:** The script captures frames from the video stream one by one.
- **Face Detection:** In each frame, the Haar Cascade algorithm is used to detect the locations of faces, returning a set of bounding box coordinates (x, y, w, h) for each face.
- **Mask Prediction:** For each detected face, the ROI is extracted, preprocessed (resized, converted to an array, and normalized), and then passed to the trained CNN model for prediction.
- **Visualization and Alert:** The model outputs the probability for "Mask" and "No Mask." Based on which probability is higher, a label and a colored bounding box (green for "Mask", red for "No Mask") are drawn on the frame. If any face is classified as "No Mask," the script triggers an audible alert.
- **Display:** The processed frame is displayed in a window, providing a live visual feed of the detection results.

5. Conclusion

This project successfully demonstrates the creation of an end-to-end Face Mask Detection system using modern computer vision and deep learning techniques. The system is capable of accurately identifying mask-wearing compliance in real-time, offering a valuable tool for public health and safety enforcement in various settings like airports, hospitals, and public transport. The modular design—separating data preparation, training, and deployment—allows for easy maintenance and future enhancements. Potential future work could include improving model accuracy with a larger and more diverse dataset, optimizing performance for deployment on low-power edge devices, and integrating the system with existing security camera networks.