Project Report: Real-Time Face Mask Detector

Introduction

This project addresses the need for an automated system to verify face mask compliance. The goal was to build an AI-powered application using computer vision to detect face masks in a live webcam feed, providing immediate visual and audio feedback. The system was developed for both standalone and web-based deployment.

Abstract

This report outlines the development of a real-time face mask detection system. A Convolutional Neural Network (CNN) was trained on a public dataset using TensorFlow/Keras. This model was integrated into a Python application using OpenCV for video capture and Haar Cascades for face localization. The system was then deployed as a Flask web application with a professional UI. The project involved significant real-world debugging, including resolving model inaccuracies due to color channel mismatches (BGR vs. RGB) and overcoming software dependency issues, resulting in a robust, interactive application.

Tools and Technologies Used

- **Backend & ML: Python, TensorFlow, Keras, OpenCV, NumPy, Flask, Winsound**
- **Frontend: HTML, CSS, JavaScript**
- **Environment: Kaggle Notebooks, GitHub, Local Machine**

Steps Involved in Building the Project

1. **Setup and Data Analysis: The project began in a Kaggle Notebook using a public COVID Face Mask dataset. An initial Exploratory Data Analysis (EDA) was performed to understand the data structure. A GitHub repository was set up for version control.**
2. **Preprocessing and Model Training: Images were preprocessed using TensorFlow's ImageDataGenerator for resizing, normalization, and data augmentation. A sequential CNN model with three convolutional blocks was built and trained for 20 epochs, with its performance tracked to prevent overfitting.**
3. **Real-Time Implementation and Debugging: The trained model was moved to a local environment. An OpenCV script was developed for live detection. Key debugging challenges were solved:**
    - **Color Correction: Fixed a major prediction bug by converting OpenCV's BGR image format to the RGB format the model expected.**

- **Dependency Conflicts: Resolved model loading and audio library installation errors by using load_weights instead of load_model and leveraging the native winsound library.**

4. **Web Deployment and UI: The detection logic was deployed as a Flask web application. The frontend was built with HTML/CSS/JS to create a professional, dark-themed UI, featuring a "Start Detection" button and a footer with creator details.**

Conclusion

This project successfully demonstrates the complete lifecycle of an AI application, from data analysis to deployment. The final product is a functional web application that accurately performs real-time face mask detection. The key takeaway is the importance of the iterative debugging phase, which proved as crucial as model training. The project serves as a practical example of applying deep learning to solve a relevant, real-world problem.