Face Mask Detection using YOLOv8

Abstract

The COVID-19 pandemic has underscored the critical need for the effective public health monitoring systems, particularly in the high-density environments. This project presents an automated **real-time face mask detection system** using the **YOLOv8**, designed to identify the individuals wearing the masks with the high accuracy and the efficiency. The system addresses the key challenges in the computer vision, including the varying lighting conditions, the occlusions, and the real-time processing requirements. Leveraging a carefully curated dataset of the 1,683 annotated images, we developed a model achieving the **90.3% mAP@0.5** with the inference speeds of **62 FPS** on an NVIDIA T4 GPU. The solution is deployed as a **scalable FastAPI web service** with the ngrok integration, enabling the seamless adoption in the surveillance systems. Comprehensive benchmarking against the alternative architectures (SSD, Faster R-CNN) demonstrates the superior performance in the both accuracy and the speed. This report details the end-to-end development process, from the dataset analysis to the development, including the ethical considerations and the future enhancement pathways.

1. Introduction

1.1 Problem Statement

The persistence of the respiratory pathogens post-pandemic necessitates continued the vigilance in the public health measures. Face mask compliance the monitoring in the high-traffic areas (airports, hospitals, public transit) remains a critical need, with the manual surveillance proving:

- **Labor-intensive**: Requires the constant human oversight
- **Inconsistent**: Subjective judgment leads to the variable enforcement
- **Non-scalable**: Impossible to monitor the large crowds effectively

Automated the computer vision solutions offer:

✓ 24/7 operational capability

✓ Objective, consistent detection

✓ Real-time alerts for the non-compliance

✓ Integration with the existing security infrastructure

1.2 Project Objectives

This project establishes the measurable technical goals:

- Implement the **dynamic confidence thresholding** for varying the lighting conditions
- Develop the **multi-face tracking** in the video streams (SORT algorithm integration)
- Optimize for the **low-power edge devices** (Jetson Nano, Coral TPU)
- Ensure the **GDPR compliance** with the anonymization features

1.3 Technology Stack

Core Components:

- **YOLOv8n Architecture**:
    - Backbone: Modified CSPDarknet53
    - Neck: PANet + SPPF
    - Head: Decoupled classification/regression

Hardware Configuration:

GPU: NVIDIA T4 (16GB VRAM)

CPU: Intel Xeon @ 2.2GHz (4 vCPUs)

RAM: 25GB DDR4

Storage: 100GB NVMe (Google Colab Pro)

Software Ecosystem:

Layer	Components	Version
Framework	PyTorch, Ultralytics	2.0.0, 8.0.0
Image Processing	OpenCV, PIL	4.7.0, 9.5.0
Data Handling	Pandas, NumPy	1.5.3, 1.23.5

Layer	Components	Version
Visualization	Matplotlib, Seaborn	3.7.1, 0.12.2
Deployment	FastAPI, Uvicorn	0.95.2, 0.22.0
Monitoring	Prometheus, Grafana	2.41.0, 9.5.1

Development Tools:

- **Data Versioning**: DVC (Data Version Control)
- **Experiment Tracking**: Weights & Biases
- **CI/CD**: GitHub Actions (Automated testing)
- **Containerization**: Docker + Kubernetes