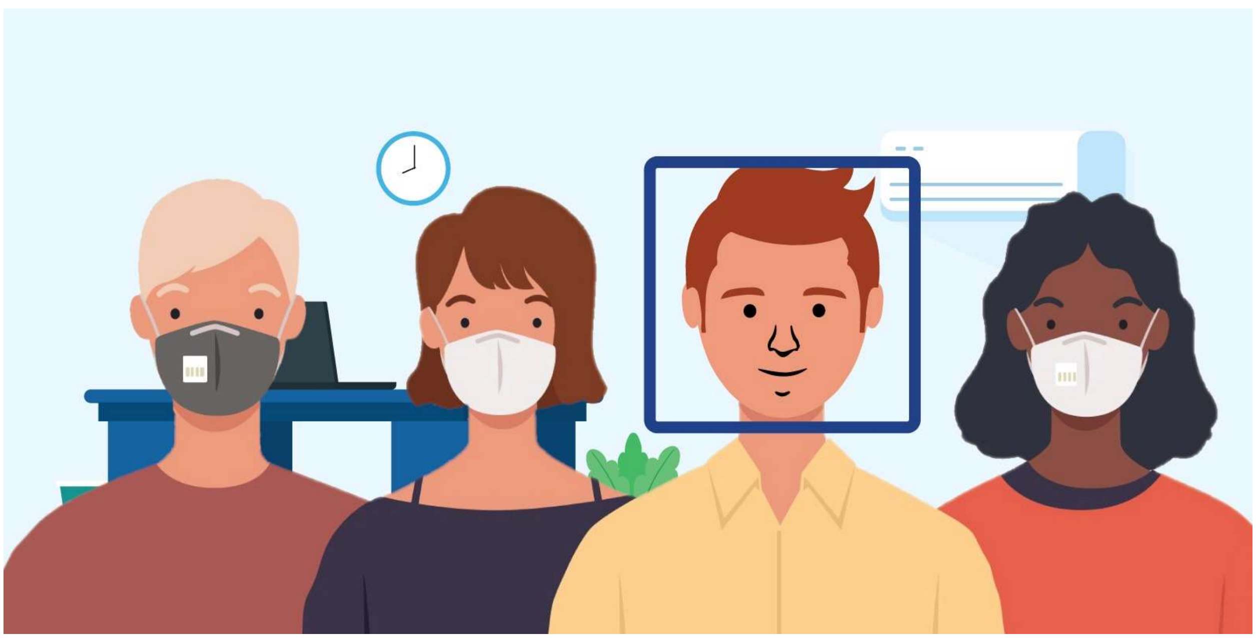
22I-1994 Muhammad Qasim

22I-2066 Ayaan Khan

22I-2003 Abubakar Nadeem

22I-1915 Ahmed Mehmood



Face Mask Detection System

1. Introduction

This report documents the development of a real-time face mask detection system using computer vision and deep learning techniques. The system can identify whether individuals in images or video streams are wearing face masks, which has significant applications in public health monitoring and safety compliance during respiratory disease outbreaks.

2. Dataset

2.1 Dataset Overview

- **Source**: Kaggle dataset "Face Mask Dataset" (https://www.kaggle.com/datasets/omkargurav/face-mask-dataset)
- **Structure**: Binary classification with two folders:
    - with_mask: Images of people wearing face masks (~3725 each)
    - without_mask: Images of people without face masks (~3725 each)
- **Sample Size**: The dataset contains thousands of facial images across both classes
- **Division**: 80% training, 20% testing (stratified split to maintain class distribution)

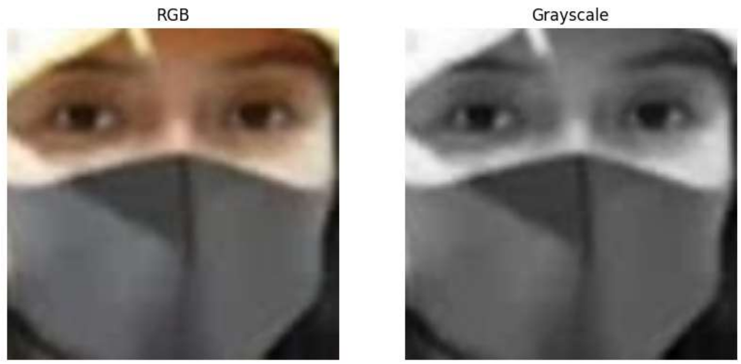
3. Image Processing Techniques

3.1 Preprocessing Pipeline

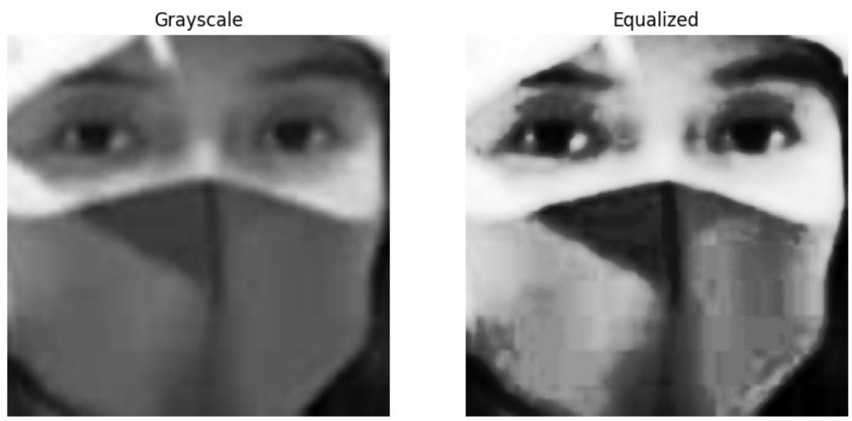
Our preprocessing pipeline includes several digital image processing techniques:

The pipeline includes:

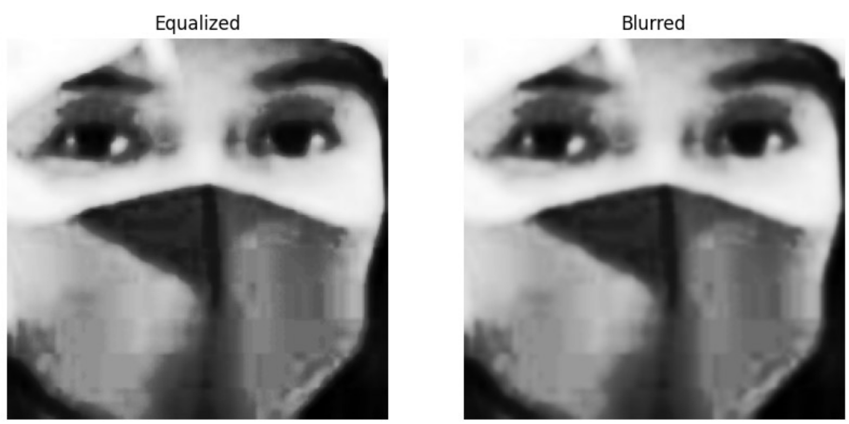
1. **Resizing**: Standardizing all images to 224×224 pixels
2. **Color Space Conversion**: Converting BGR to RGB for model input and to grayscale for further processing:



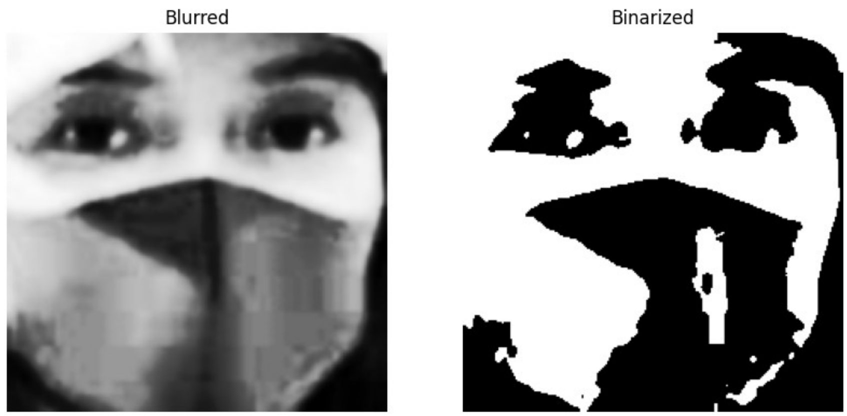
3. **Histogram Equalization**: Enhancing contrast in grayscale images



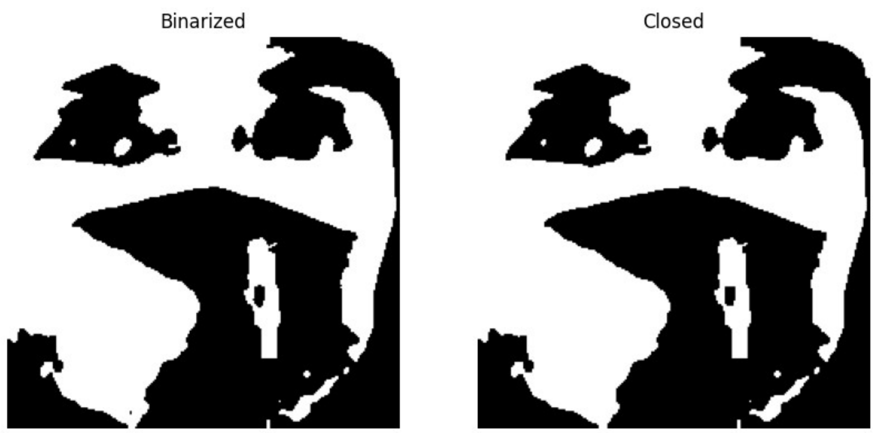
4. **Gaussian Blur**: Reducing noise and detail



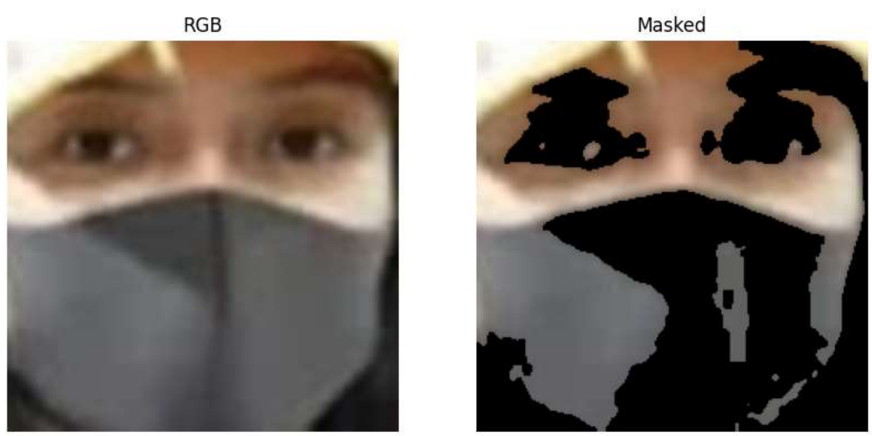
5. **Otsu Thresholding**: Adaptive binarization to separate foreground from background



6. **Morphological Operations**: Closing operation to fill small holes and connect nearby components



7. **Masking**: Applying the binary mask to the RGB image to focus on relevant facial features



8. **MobileNetV2 Preprocessing**: Normalizing pixel values as required by the model

4. Model Architecture

4.1 MobileNetV2 Architecture

After initial attempts with a custom CNN model that proved computationally intensive, taking a long time for the model to train.

```
INFO:tensorflow:Using MirroredStrategy with devices ('/job:localhost/replica:0/task:0/device:CPU:0',)
GPUs detected: 1
Training on 6042 images with 94 steps per epoch
Validating on 1511 images with 23 validation steps
Epoch 1/15
 3/94 ━━━━━━━━━━━━━━━━━━━━ 5:30:42 218s/step - accuracy: 0.5095 - loss: 0.7068
```

We then adopted MobileNetV2 for transfer learning:

MobileNetV2 is an efficient CNN architecture designed for mobile and edge devices with the following key features:

- **Inverted Residual Blocks**: Unlike traditional residual connections, MobileNetV2 uses inverted residuals where the shortcut connections are between thin bottleneck layers
- **Linear Bottlenecks**: Removes non-linearities in narrow layers to prevent information loss
- **Depth wise Separable Convolutions**: Factorizes standard convolutions into depth wise and pointwise convolutions, significantly reducing computation
- **Expansion and Projection Layers**: Expands channels before depth wise convolution and projects back to a smaller dimension

4.2 Fine Tuning

We implemented the following hyperparameter optimization strategies:

- **Learning Rate Scheduling**: Reducing learning rate when validation loss plateaus
- **Early Stopping**: Preventing overfitting by monitoring validation loss
- **Fine-tuning**: Unfreezing and training the last 30% of the base model layers
- **Dropout**: Adding 20% dropout for regularization
- **Data Augmentation**: Implementing random flips, rotations, zooms, and contrast adjustments

5. Face Detection

For real-time face detection, we utilized a pre-trained model:

- **Architecture**: Single Shot Multibox Detector (SSD) with ResNet-10 backbone
- **Model Files**:
    - deploy.prototxt: Network architecture definition
    - res10_300x300_ssd_iter_140000_fp16.caffemodel: Pre-trained weights
- **Implementation**: OpenCV's DNN module for efficient inference

6. Web Application

We developed a Flask-based web application with the following features:

- **Real-time Video Processing**: Webcam integration for live mask detection
- **Image Upload**: Ability to process uploaded images
- **Statistics**: Tracking of detected faces, mask status, and confidence levels
- **Visualization**: Bounding boxes and labels for detected faces

7. Results

7.1 Performance Metrics

The MobileNetV2-based model achieved:

- High accuracy in mask detection

```
    accuracy                           0.88      1511
   macro avg       0.88      0.88      0.88      1511
weighted avg       0.88      0.88      0.88      1511
```

- Real-time performance on standard hardware
- Robust face detection across various lighting conditions

7.3 Visual Results:

Web Application Interface:

Figure 1: Screenshot of the Flask web application interface showing real-time detection, statistics, and user controls.

Image of a Single Person with Mask:

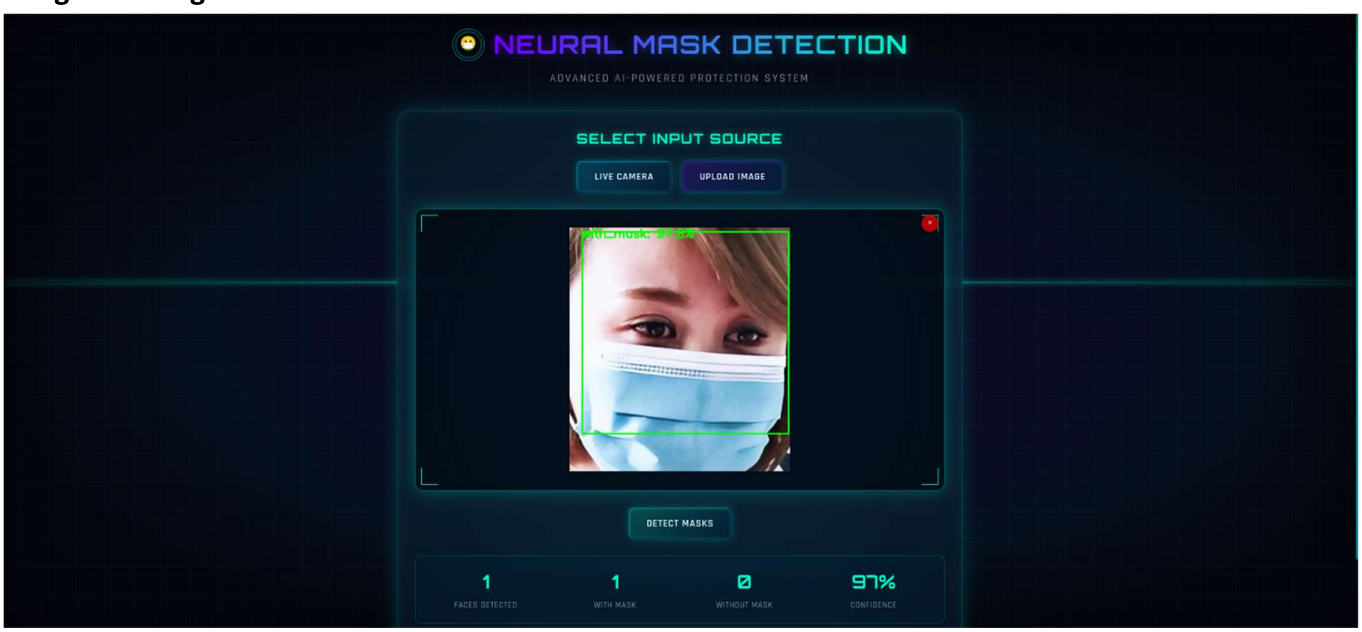


Figure 2: Detection of a single person wearing a face mask in an image. The system correctly identifies the mask with high confidence (green bounding box).

Image of Multiple People with Masks:

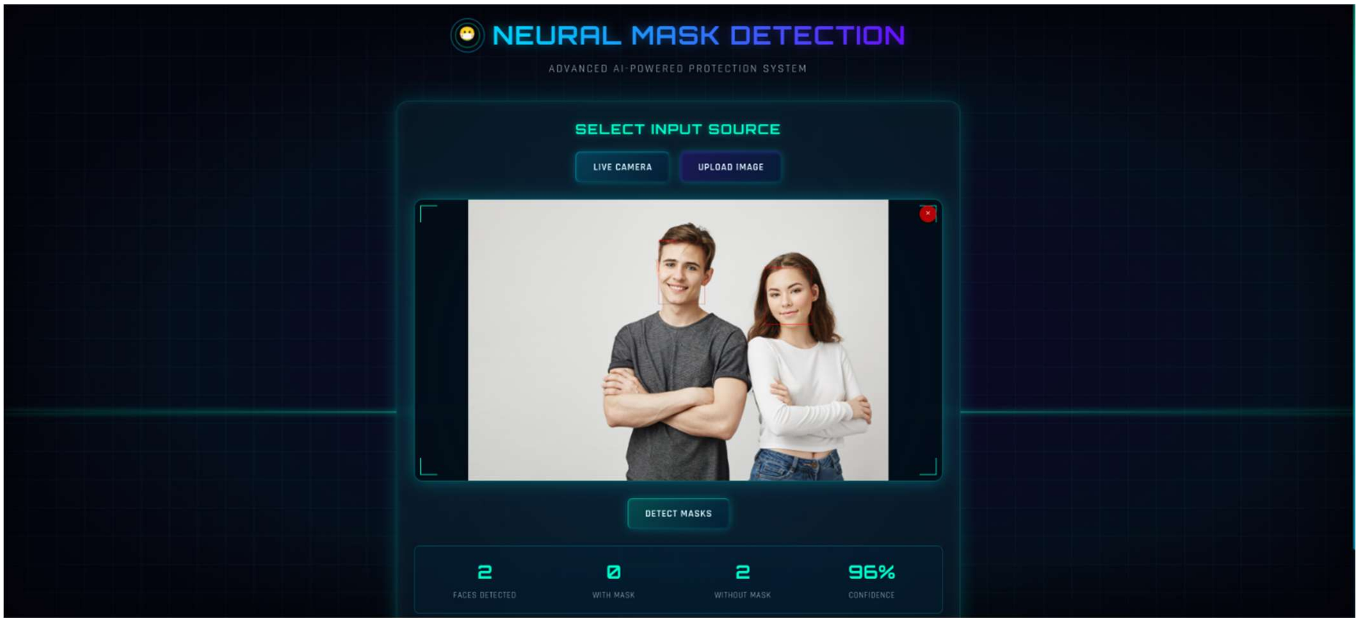


Figure 2: Detection of a multiple people not wearing a face mask in an image. The system correctly identifies the absence of a mask (red bounding box).

Real Time Video With Mask:

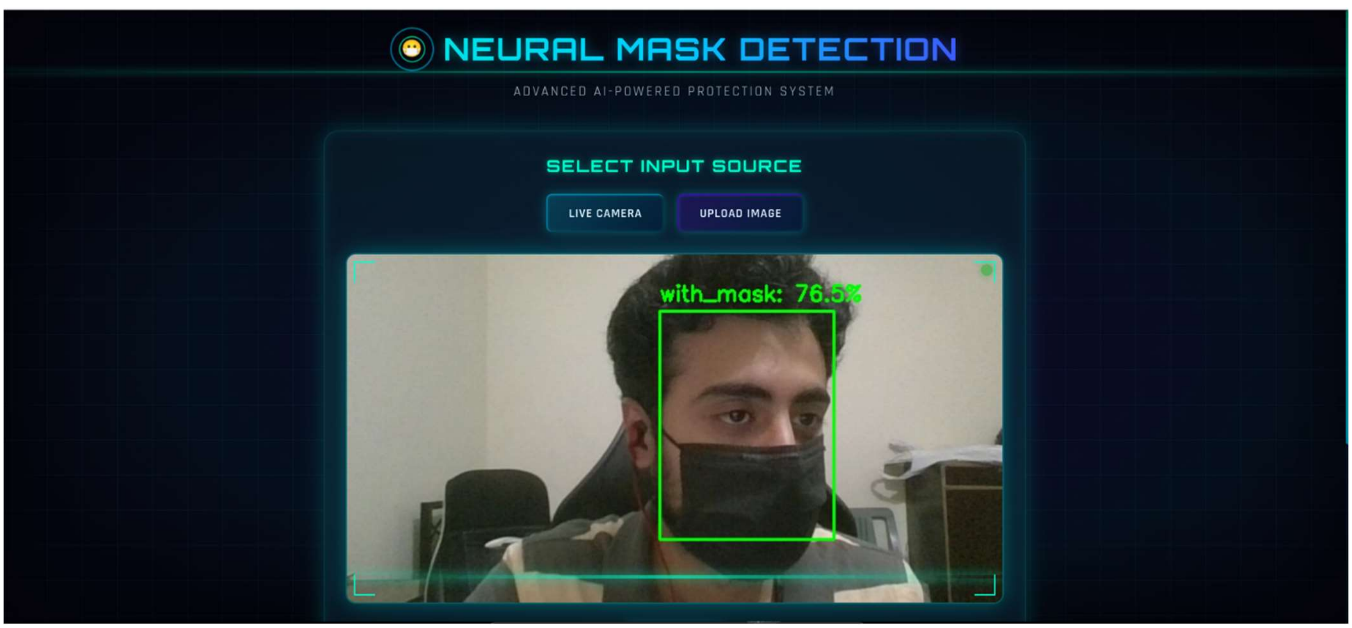


Figure 3: Detection of single person wearing a mask in real time video. The system successfully identifies all faces and their mask status.

7.3 Limitations

- Performance degradation with occlusions other than masks
- Reduced accuracy with extreme head poses
- Computational constraints for high-resolution video streams

8. Future Work

8.1 Potential Improvements

- Multi-class classification (different types of masks)
- Integration with crowd monitoring systems
- Edge deployment optimization
- Attention mechanisms to focus on relevant facial regions

8.2 Additional Features

- Mask-wearing compliance statistics over time
- Integration with access control systems
- Mobile application development

9. Conclusion

This project successfully demonstrates the application of computer vision and deep learning techniques for face mask detection. By leveraging transfer learning with MobileNetV2 and implementing a comprehensive image processing pipeline, we achieved an efficient and accurate system capable of real-time mask detection. The Flask-based web application provides an accessible interface for utilizing this technology in practical settings.