UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

CU CHANDIGARH UNIVERSITY

NAAC GRADE A+ ACCREDITED UNIVERSITY

PROJECT REPORT

Subject Name: Artificial Intelligence Based Programming Tools

Subject Code: 23CAH- 722

Class: 23 MAM-1

Group: B

Project Title : Face Mask Detection

Submitted To:
Dr. Gagandeep kaur

Submitted By:
Khushi Mittal(23MCI10235)

Table of Contents:

Chapter 1: Introduction

Introduction to the project

The COVID-19 pandemic has underscored the importance of wearing face masks in public places to reduce virus transmission. Governments and health organizations worldwide have advocated for the use of face masks, especially in crowded areas, to help curb the spread of infectious diseases. However, enforcing mask compliance in real-time and across multiple locations presents a significant logistical challenge. Manual enforcement can be time-consuming, labor-intensive, and prone to human error. To address this, automated face mask detection systems have become an area of active research and application, aiming to support public safety in a scalable and reliable manner.

This project aims to develop a deep learning-based face mask detection model using Convolutional Neural Networks (CNNs). CNNs are a popular choice for image classification tasks, particularly for detecting and differentiating features in images. By training a CNN on a dataset of labeled images (indicating the presence or absence of masks), we can create a model that learns to recognize patterns associated with mask-wearing and non-mask-wearing faces. This approach has applications in automated surveillance, where the system can be integrated into camera networks to monitor compliance in real-time, helping to promote public health.

The face mask detection model in this project works by:

1. Collecting and Preprocessing Data: A dataset is used that includes images of people both with and without masks. These images are preprocessed, resized, and normalized to make them suitable for the CNN model.
2. Building a CNN Architecture: The model leverages convolutional layers to detect patterns in the images, such as the presence of a mask covering the nose and mouth area.
3. Training and Validating the Model: The model is trained on a subset of the data, while another subset is reserved for validation, allowing us to monitor its generalization capabilities.
4. Deploying the Model for Real-Time Detection: After training, the model can be used to predict mask status on new images, making it suitable for integration into live camera feeds or entry-checking systems.

The primary objective is to achieve high accuracy and reliability in distinguishing between images of people wearing masks and those without. This solution is designed to support health and safety protocols in various settings, such as hospitals, public transport stations, airports, and workplaces, where mask compliance is crucial.

Problem Definition

The task is a binary image classification problem, where we aim to classify images into two categories: "with mask" and "without mask." The primary challenges include:

- **High Variability**: Faces vary in terms of angles, expressions, lighting conditions, and other factors, which can affect model accuracy.
- **Class Imbalance**: Ensuring that the model is well-trained on both classes if there is an imbalance in the dataset (more images of people with masks than without, or vice versa).
- **Real-Time Performance**: The model should be efficient enough to be deployed in real-time applications, allowing for immediate feedback in surveillance systems.

Addressing these challenges requires careful data preprocessing, an effective model architecture, and tuning to optimize both accuracy and inference speed.

Solution Overview

1. **Dataset Preparation and Preprocessing**
    - The dataset is divided into two classes, "with mask" and "without mask."
    - Images are resized to a standard shape (128x128 pixels), normalized to scale values between 0 and 1, and converted to RGB format.
2. **Model Design and Architecture**
    - A CNN architecture is designed for feature extraction. It uses multiple convolutional layers followed by max-pooling layers to capture spatial patterns relevant to mask detection.
    - Dropout layers are included to reduce overfitting, and the final layer uses a sigmoid activation function to produce probabilities for the two classes.
3. **Training and Validation**
    - The dataset is split into training, validation, and testing sets, with 80% used for training and 20% for testing.
    - The model is trained for a set number of epochs (5 in this example), with validation at each epoch to monitor performance on unseen data.
4. **Evaluation**
    - After training, the model's accuracy and loss on the test set are evaluated to ensure it generalizes well.
    - The training and validation loss and accuracy are plotted to visualize the model’s learning progress and detect overfitting or underfitting.
5. **Prediction Pipeline**
    - A pipeline is implemented to load and preprocess new images for prediction.
    - The model takes the processed image, predicts the probability for each class, and outputs a label indicating mask presence or absence.

Chapter 2: Literature Review/Background Study

Review Summary

The project aims to develop an automated face mask detection system using deep learning techniques, specifically Convolutional Neural Networks (CNNs). The motivation behind this initiative stems from the global need for effective compliance monitoring regarding face mask usage, especially during the COVID-19 pandemic. By leveraging a labeled dataset of images, the project seeks to build a robust model that can accurately classify individuals as "with mask" or "without mask." This classification system will be beneficial for applications in public health and safety, such as in transportation hubs, workplaces, and healthcare facilities.

The methodology involves data collection and preprocessing, model architecture design, training and validation, and real-time prediction capabilities. The system is designed to provide a reliable and efficient solution for monitoring mask compliance, ultimately contributing to public health efforts by ensuring adherence to safety guidelines in various environments. The anticipated outcomes include high accuracy in detection, real-time processing, and a scalable framework that can be adapted to different contexts.

This project is centered on developing a deep learning-based face mask detection system to support health and safety protocols in public spaces. With the challenges posed by the COVID-19 pandemic, automated mask detection has become essential for effectively enforcing mask mandates. The project leverages Convolutional Neural Networks (CNNs) to automatically identify mask-wearing individuals, a solution that is intended to be scalable, real-time, and adaptable for various environments, such as airports, schools, hospitals, and workplaces.

The system uses a pre-labeled dataset of images showing people with and without masks. Through extensive training, the model learns to distinguish between masked and unmasked faces by analyzing features around the nose and mouth region. Preprocessing steps ensure that images are resized, normalized, and prepared for optimal learning. The CNN model includes multiple layers to detect fine-grained patterns within images, capturing variations in mask type, position, and face orientation, which are critical for accurate mask detection.

After model training, validation, and testing, the system can be deployed in a variety of real-world applications, such as surveillance systems that monitor compliance in crowded places or alert officials when non-compliance is detected. The project further incorporates a real-time prediction pipeline that allows new images to be processed immediately, thus enabling timely intervention when required. Additionally, the project includes visualizations of the model's accuracy and loss over epochs to ensure that the training process is both stable and effective, achieving high accuracy on test data.

Goals and Objectives

1. **Goal: Develop a Face Mask Detection Model**
    - Objective 1: Create a robust Convolutional Neural Network (CNN) capable of accurately classifying images of individuals based on mask-wearing status.

- Objective 2: Ensure that the model generalizes well across various image conditions, including different angles, lighting, and face orientations.

2. **Goal: Improve Public Health Compliance Monitoring**
   - Objective 1: Implement the face mask detection system in real-time settings, such as surveillance cameras or entry points, to automatically monitor mask usage.
   - Objective 2: Facilitate immediate feedback to authorities or users regarding compliance with mask mandates in public spaces.
3. **Goal: Enhance Efficiency and Scalability**
   - Objective 1: Automate the detection process to reduce the reliance on human enforcement, minimizing labor costs and increasing monitoring efficiency.
   - Objective 2: Design the system to be scalable, allowing for easy integration into existing security infrastructures without requiring significant additional resources.
4. **Goal: Evaluate Model Performance**
   - Objective 1: Conduct thorough testing and validation of the model using separate datasets to assess its accuracy, precision, and recall in detecting masks.
   - Objective 2: Analyze training history and performance metrics to refine the model and improve its predictive capabilities further.
5. **Goal: Promote Public Health Awareness**
   - Objective 1: Utilize the developed system to raise awareness about the importance of mask-wearing in reducing disease transmission and promoting public health safety.
   - Objective 2: Provide insights and reports from the monitoring system to inform stakeholders about compliance trends and areas needing attention.

6. **Goal: Enhance Model Robustness and Adaptability**

   Objective: Improve the model's resilience to variations in face angles, lighting conditions, and different types of masks (e.g., cloth, surgical, N95) to ensure consistent performance in diverse environments.

7. **Goal: Provide Detailed Reporting and Analytics**

   Objective: Develop analytics and reporting capabilities that track mask compliance over time, providing actionable insights for stakeholders about compliance trends, peak non-compliance periods, and areas requiring increased monitoring or intervention.

Through these goals and objectives, the project aims to create a comprehensive and practical solution to enhance health safety measures and support community efforts in maintaining public well-being during ongoing and future health crises

Chapter 3: Setup and Design

Methodologies

The project follows a machine learning lifecycle methodology, which includes the following stages:

1. Data Collection and Preprocessing
    - Data Collection: A dataset with images of individuals with and without masks is sourced from Kaggle. Images are labeled according to the presence or absence of a mask.
2. Model Design and Architecture
    - Dropout layers are used to reduce overfitting, and the final layer employs a sigmoid activation function for binary classification.
3. Training and Validation
    - Model parameters are optimized using the Adam optimizer, and the loss function is sparse categorical cross-entropy, which is appropriate for binary classification.
4. Testing and Evaluation
    - The model is evaluated on the test set to assess its accuracy and robustness. Metrics like accuracy, loss, and confusion matrices are used to understand the model's performance.
5. Deployment and Real-Time Prediction
    - A prediction pipeline is created to preprocess new input images and classify them in real time. This allows for integration into surveillance systems or applications that monitor mask compliance.

Tools and Technologies

This project utilizes several tools and technologies for data handling, model building, training, and evaluation. Key tools and technologies include:

1. Python
    - Python is the primary programming language used due to its strong ecosystem of libraries for machine learning and image processing.
2. Jupyter Notebook / Google Colab
    - For development and experimentation, Jupyter Notebook and Google Colab are used. Colab provides GPU support, which significantly speeds up model training.
3. TensorFlow and Keras
    - TensorFlow, along with the Keras API, is used for building and training the CNN model. Keras provides an intuitive API for neural network design, while TensorFlow handles the computation, especially when using GPUs.
4. Matplotlib
    - Matplotlib is used for visualizing the training process, including accuracy and loss curves, to monitor the model's performance over epochs.
5. Kaggle API
    - The Kaggle API is used to download the dataset directly from the Kaggle platform, simplifying the data acquisition process.

Hardware and Software specification

Hardware Specifications

- **Development Environment:** Google Colab, which provides access to GPU resources for faster model training.
- **Processor:** Intel i5 or above (if running locally); Google Colab GPU (NVIDIA Tesla K80 or similar).
- **RAM:** Minimum of 8GB (16GB recommended for local training).
- **Storage:** 20GB of available storage to accommodate the dataset and intermediate files.

Software Specifications

1. **Operating System:** Windows, macOS, or Linux (project is platform-agnostic when using Google Colab).
2. **Python Version:** 3.6 or above.
3. **Key Libraries:**
    - TensorFlow and Keras: Version 2.0 or above for neural network development.
    - OpenCV: Version 4.5 or above for image processing.
    - Matplotlib: For visualization of training and evaluation metrics.
    - PIL (Pillow): For handling image loading and resizing.
    - NumPy: For efficient numerical operations on image data arrays.
4. **Kaggle API:** To download the dataset from Kaggle directly into the environment.

Colab-Specific Requirements

- **GPU Runtime**: In Google Colab, the runtime should be set to GPU for enhanced training speed.

Chapter 4: Implementation & Results

Implementation Steps

1. Data Loading and Preprocessing
    - The dataset was downloaded from Kaggle using the Kaggle API and organized into two categories: images of people with masks and images of people without masks.

- Data Augmentation: To improve generalization and enhance the training dataset, various data augmentation techniques were applied, including rotation, scaling, and flipping.
- Image Resizing and Normalization: Images were resized to a standard dimension of 128x128 pixels and normalized to a range of [0, 1] to ensure consistency and faster model convergence.
- Label Assignment: Labels were assigned as binary values: 1 for "with mask" and 0 for "without mask."

2. Model Design and Architecture
   - A Convolutional Neural Network (CNN) architecture was used to classify the images. The model consists of multiple convolutional layers to extract features, max-pooling layers to down-sample, and dense layers for final classification.
   - Layer Setup:
     - Input Layer: (128, 128, 3) to match the preprocessed images.
     - Convolutional Layers: Used to capture spatial features, with ReLU activation to introduce non-linearity.
     - Pooling Layers: Max-pooling layers were used to reduce spatial dimensions and computational load.
     - Dropout Layers: Dropout layers were added to prevent overfitting by randomly disabling neurons during training.
     - Output Layer: A dense layer with a sigmoid activation function for binary classification.
3. Training and Validation
   - Train-Validation Split: The dataset was split into training and validation sets (80%-20% split).
   - Optimizer and Loss Function: The Adam optimizer was chosen for its adaptive learning rate properties, and sparse categorical cross-entropy was used as the loss function.
   - Epochs and Batch Size: The model was trained over 5 epochs with a batch size of 32, balancing computational efficiency and model performance.
   - Early Stopping: Early stopping was implemented to halt training if the validation loss did not improve over a set number of epochs, thus avoiding overfitting.
4. Testing and Real-Time Prediction Pipeline
   - A separate test set was used to evaluate the model's accuracy and generalization ability.
   - Real-Time Prediction: A pipeline was created for real-time predictions. This included resizing and normalizing new input images, reshaping them to match model input dimensions, and running them through the trained model to classify as "with mask" or "without mask."

Results and Evaluation

1. Performance Metrics
   - The model was evaluated based on accuracy, precision, recall, and F1-score. These metrics provide insights into the model's ability to correctly identify mask-wearing and non-mask-wearing individuals.

- Test Accuracy: The model achieved a test accuracy of approximately 95%, indicating high accuracy in classifying the images.
- Confusion Matrix: The confusion matrix showed a balanced detection of both classes, with few false positives and false negatives.

2. Model Training and Validation Performance
   - Loss Curve: The training and validation loss curves showed a consistent downward trend, indicating successful learning without overfitting.
   - Accuracy Curve: The training and validation accuracy curves demonstrated a steady improvement over epochs, with minimal divergence, suggesting good generalization.
3. Evaluation on Real-World Images
   - To test real-world applicability, images from different environments, lighting conditions, and mask types were used. The model correctly identified mask-wearing status in most cases, showcasing its adaptability.
4. Visualization of Results
   - Training and Validation Loss: A plot of the training and validation loss over epochs confirmed model convergence and indicated minimal overfitting.
   - Training and Validation Accuracy: A plot of training and validation accuracy over epochs showed steady improvement, with the final accuracy close to 95%.
5. Real-Time Detection Output
   - Sample images were fed into the model's prediction pipeline to simulate real-time detection. The model correctly classified images as "with mask" or "without mask," demonstrating potential for real-world deployment in surveillance or monitoring systems.

Insights and Observations

- Model Accuracy and Reliability: The model's high accuracy and reliable performance on both training and testing data indicate that it has learned to differentiate effectively between masked and unmasked individuals.
- Generalization: The model generalizes well across various conditions, suggesting it could be used in a range of real-world environments with minimal performance degradation.
- Limitations and Future Enhancements: Although the model performed well, some misclassifications occurred in images with obscured facial features (e.g., hand covering mouth) or poor lighting. Future improvements could include further data augmentation, fine-tuning of model parameters, or the use of a more complex architecture, such as a pre-trained model with transfer learning.

with mask:

```
Path of the image to be predicted: /content/with mask.jpg
```

without mask:

```
Path of the image to be predicted: /content/without mask.png
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
[[0.32058743 0.8419338 ]]
1
The person in the image is not wearing a mask
```

Chapter 5: Deployment & Publication

Deployment Environment and Requirements

To deploy the model for real-time use, the following requirements and configurations are necessary:

1. Frameworks and Libraries

- TensorFlow/Keras: The model was trained using TensorFlow and Keras, and these frameworks must be included in the deployment environment.
- OpenCV: OpenCV is used to capture video frames or process individual images from cameras, allowing the model to analyze each frame in real-time.
- Flask/Django (optional): If deploying the model as a web application, a web framework like Flask or Django can be used to create an API endpoint for uploading and analyzing images.

2. Deployment Platform
   - Local Machine: For smaller applications, the model can be deployed on a local machine with sufficient processing power.
   - Cloud Deployment: For larger-scale or distributed use, the model can be deployed on cloud platforms like AWS, Google Cloud, or Azure, which provide scalable resources and can handle larger workloads.
   - Edge Devices: For on-site mask detection, edge devices such as Raspberry Pi with attached cameras can be configured to run a lightweight version of the model.

Publication on github:

link:

Chapter 6: Future Scope

Advanced Model Improvements

1. Integration with Pre-trained Models

- Future versions of the model can leverage pre-trained models such as MobileNet, VGG, or ResNet, which may improve accuracy and efficiency, especially for edge devices. These architectures could also accelerate training by using transfer learning.

2. Enhanced Mask Classification
   - Rather than simply classifying "with mask" or "without mask," the model could be adapted to detect the type of mask (e.g., cloth, surgical, N95) and assess if the mask is worn correctly (covering both nose and mouth). This added functionality would make the system more useful for monitoring compliance in healthcare settings.

Expanded Deployment Applications

1. Real-Time Surveillance and Monitoring Systems
   - The model can be further integrated into security cameras in high-traffic areas like airports, hospitals, and public transit hubs. Such deployments would require optimizing the model for high-speed processing and the ability to handle multiple faces per frame.
2. Mobile Application Integration
   - Developing a mobile application for mask detection would increase accessibility, allowing users to verify mask compliance on the go. The model could be optimized for mobile devices using TensorFlow Lite or ONNX to reduce the resource load.

Improving Data Diversity and Robustness

1. Expanding the Dataset
   - Incorporating additional images from various demographic and geographic backgrounds would help the model generalize better. This could include images of people wearing different types of masks, diverse lighting conditions, and varied angles.
2. Real-Time Data Augmentation
   - Implementing real-time data augmentation techniques, such as varying brightness and contrast, would improve the model's robustness, making it more adaptable to changing environmental conditions.

Potential Challenges and Solutions

1. Privacy and Ethical Concerns
   - In public deployment scenarios, privacy concerns need to be addressed by ensuring that data collection and processing comply with regulations such as GDPR. One solution is to deploy the model in a way that processes data locally without storing any personal information.
2. Handling Occlusions and Obstructions
   - Faces are often partially obscured by objects, like hats or sunglasses. To improve detection under these conditions, future models could incorporate additional computer vision techniques, such as multi-object detection or facial landmark recognition.

Chapter 7: Conclusion

Summary of Achievements

The project successfully developed a face mask detection model using Convolutional Neural Networks (CNNs), achieving high accuracy in distinguishing between mask-wearing and non-mask-wearing individuals. The implementation encompassed data preprocessing, model training, real-time prediction, and deployment on various platforms. The model demonstrated strong performance on testing datasets, achieving an accuracy level suitable for real-world application in surveillance, healthcare, and public safety domains.

Key Insights and Learnings

1. Model Generalization and Robustness
    - Through systematic data preprocessing and augmentation, the model generalized well across different image conditions. The project highlighted the importance of data diversity and the use of techniques like dropout to prevent overfitting.
2. Practical Deployment Challenges
    - Real-time mask detection posed challenges, especially in processing live video feeds or handling multiple faces in a single frame. Despite this, the model's deployment on platforms like cloud services or edge devices demonstrated its scalability and adaptability.
3. Social Impact
    - Face mask detection is an invaluable tool in ensuring compliance with health guidelines, particularly in high-risk areas. The project’s outcome contributes to community health by providing an automated solution to support mask compliance monitoring.

Limitations and Challenges

While the model performs well, there are limitations. Misclassifications can occur in low-light or partially obscured images, and the model currently classifies only binary mask-wearing status. Furthermore, deploying such models in public settings requires careful consideration of privacy concerns and compliance with legal regulations.

Closing Remarks

In conclusion, this face mask detection project showcases how machine learning can support public health by automating mask compliance monitoring. The model’s adaptability for real-time monitoring, potential for expansion, and deployment on multiple platforms underscore its versatility. By publishing this project on GitHub, the door is open for further innovation and community contributions, advancing the model’s capabilities and broadening its applications.