JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

FACEMASK AND BODY TEMPERATURE DETECTION SIMULATION BY USING PROTEUS SOFTWARE

Mrs Swetha B[1], Harshitha H S[2], Nikitha P[3]

Professor, Department of E&C, B.I.E.T, Davangere, Karnataka, India[1]

Students, U.G., Department of E&C, B.I.E.T, Davangere, Karnataka, India[2,3]

Abstract

The COVID-19 pandemic underscored the necessity for intelligent monitoring systems capable of ensuring public safety through automated screening. This work presents a contactless, integrated approach for face mask detection and body temperature monitoring using a Raspberry Pi-based embedded platform. A deep learning model implemented through a Convolutional Neural Network (CNN) combined with OpenCV facilitates accurate mask recognition, while a temperature sensor records real-time body temperature. The system provides an efficient, cost-effective, and scalable solution suitable for deployment in public environments such as hospitals, institutions, and transportation hubs.

Key words: Face Mask Detection, CNN, Raspberry Pi, Temperature Monitoring, IoT

1. INTRODUCTION

The rapid spread of the COVID-19 pandemic emphasized the urgent need for intelligent and automated safety systems to monitor individuals in public environments. To address this challenge, this research proposes a Face Mask and Body Temperature Monitoring System that utilizes a Raspberry Pi platform integrated with deep learning and computer vision technologies. The system employs a camera module and temperature sensor to simultaneously detect the presence of a face mask and measure body temperature in real time.

The implementation leverages Convolutional Neural Networks (CNN) and OpenCV for image processing, enabling accurate face detection and mask classification. A pre-trained MobileNetV2 model is adapted to the Raspberry Pi for efficient edge-based performance, offering a low-cost yet effective surveillance solution. By combining image recognition with temperature sensing, the proposed design provides a contactless, automated, and scalable system suitable for public places such as hospitals, educational institutions, and transportation hubs.

This approach aims to enhance community safety by integrating IoT-driven automation with machine learning-based decision making, ensuring continuous real-time monitoring without the need for human supervision.

2. RELATED WORK

Several research studies have explored automated monitoring systems integrating Internet of Things (IoT) technologies with deep learning for enforcing safety measures during the COVID-19 pandemic. In [1], the authors combined IoT-based camera networks with Faster R-CNN to identify individuals wearing or not wearing face masks while estimating interpersonal distance to ensure social compliance. Although effective, their approach required high computational resources and demonstrated sensitivity to lighting and camera orientation.

The study in [2] introduced a Raspberry Pi–based face mask detection model employing a Convolutional Neural Network (CNN) to classify masked and unmasked faces in real time. This design achieved portability and low cost but suffered from reduced accuracy in low-light environments and

limited processing capacity inherent to the Raspberry Pi platform.

In [3], the authors developed a deep learning–based mask detection system trained on datasets containing both masked and unmasked facial images. The CNN model provided reliable classification accuracy; however, the performance degraded when dealing with occluded faces, poor illumination, or limited training data.

A complementary study [4] emphasized edge computing by performing face mask detection directly on the Raspberry Pi device to minimize cloud dependency. Although this improved response time and system autonomy, the method remained constrained by the device's limited hardware capabilities.

Finally, [5] proposed a smart door-lock mechanism integrating face mask detection with a Raspberry Pi and transfer learning model. The system granted or denied access based on mask status, demonstrating practical applications of computer vision for safety automation. Nonetheless, issues such as inconsistent lighting, partial mask coverage, and hardware latency affected its overall robustness.

3. METHODOLOGY

The proposed face mask detection framework is systematically designed by integrating computer vision, deep learning, and real-time image processing techniques. The system processes live video input acquired from a Raspberry Pi camera to determine the presence or absence of a face mask. A Convolutional Neural Network (CNN) model is employed for classification, trained on a dataset containing both masked and unmasked facial images to ensure high accuracy and robustness.

Upon capturing real-time frames, the system performs face detection and subsequently classifies each detected region into either "mask" or "no mask" categories. The CNN-based feature extraction and classification are executed on the Raspberry Pi module, optimizing the model for embedded edge-device performance. In mask detection, the system integrates a temperature monitoring module, which measures the body temperature of the individual using a connected sensor. The detected temperature and mask status are displayed on an LED screen, providing immediate visual feedback.

The overall workflow involves continuous data acquisition, preprocessing, feature extraction, and classification, enabling the system to operate autonomously in real time. The combined use of deep learning algorithms and image processing enhances detection accuracy while maintaining efficient computation suitable for Raspberry Pi's hardware limitations.

SYSTEM ARCHITECTURE

The architecture of the proposed system is illustrated in Fig.1 which outlines the overall workflow from data acquisition to result generation. The system integrates a Raspberry Pi camera for capturing input, a Raspberry Pi module for processing and feature extraction, and a MySQL database for data storage and retrieval. The design ensures smooth interaction between hardware and software components to achieve efficient face mask detection and monitoring.

The process initiates with user registration, wherein facial images are captured through the Sony IMX219 camera sensor. The camera is configured via the I2C protocol to ensure proper calibration and optimized data acquisition. The captured image is subsequently pre-processed by converting it from the RGB colour domain to grayscale, thereby reducing computational complexity and facilitating the extraction of relevant features.

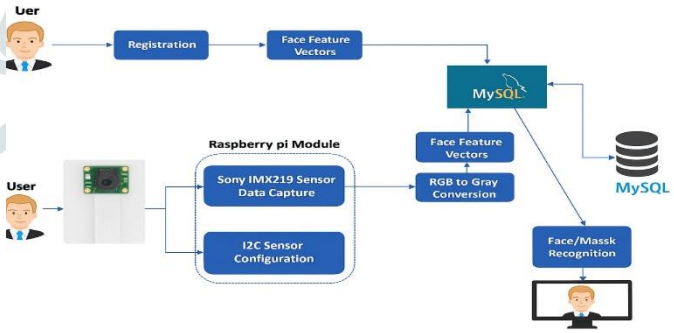

Fig.1: System architecture

From the grayscale images, the system generates face feature vectors, which are unique numerical representations of the user's facial characteristics. These feature vectors are then stored in a MySQL database to serve as reference data for future recognition tasks. During the recognition phase, the Raspberry Pi again captures the facial image of the user, performs the same preprocessing, and extracts the corresponding feature vectors. The extracted vectors are

compared against the stored database entries to authenticate the user.

This architecture ensures secure and real-time operation by combining the lightweight processing capability of the Raspberry Pi with the robust storage and retrieval functions of the MySQL database. The system design makes it suitable for various applications such as biometric authentication, automated attendance monitoring, access control systems, and real-time surveillance.

4. CNN ALGORITHM

Fig.2 shows architecture of a Convolutional Neural Network (CNN), a type of machine learning model particularly effective for image classification tasks. The CNN's workflow is depicted in a blue box labelled "ML - Convolutional Neural Network." It starts with an input image that undergoes feature extraction through convolution and pooling operations.

The convolution step involves applying filters to the input image to detect features like edges or patterns, resulting in feature maps. Pooling reduces the spatial dimensions of these feature maps, helping to decrease computational complexity and enhance robustness. After feature extraction, the data is fed into a fully connected layer for classification. Here, the model makes predictions based on the extracted features, leading to an output. the output layer, as noted in the text above the diagram, typically uses SoftMax for multi-class classification or Sigmoid for binary classification to produce probabilities for each class. CNNs are widely used in various applications, including object detection, facial recognition, and image segmentation, due to their ability to automatically and adaptively learn spatial hierarchies of features from images.

The architecture of a CNN can be customized and fine-tuned for specific tasks by adjusting the number of layers, the type of activation functions used, and the optimization algorithms employed during training. This flexibility, combined with their powerful feature extraction capabilities, makes CNNs a cornerstone of modern computer vision and deep learning research. By leveraging large datasets and computational power, CNNs can achieve state-of-the-art performance in many image-related tasks, driving advancements in fields such as healthcare, security, and autonomous systems.

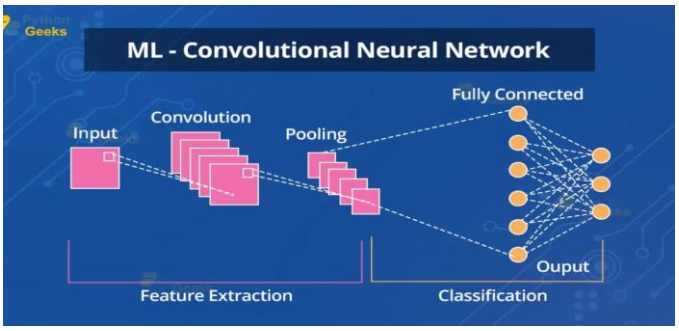

Fig.2: CNN Algorithm

The Pre-processing process includes: -

- Loading image dataset.
- Resizing dataset images to 224 x 224 pixels.
- Conversion to array format using NumPy array.
- Scaling the pixel intensities in the input image.
- Ensuring training data is in NumPy array format.
- Segmentation image data for training and testing.
- Data augmentation.

The training includes the following process: -

- Loading Image Data
- Pre-processing
- Loading the MobileNetV2 classifier
- Building a new fully connected (FC) head

5. Algorithm for Face Mask and Body Temperature Detection System.

1. Start
2. Initialize all hardware components including the camera module, temperature sensor, LCD display, and gate control mechanism.
3. Start mask detection program
4. Check mask status

   If mask detected, go to step 5

Else,

   Display message “Mask not detected” on LCD

   Do not open gate

   Go back to step 3

5. Display “Mask detected” on LCD
6. Read body temperature from the temperature sensor
7. Check temperature range

   If 90 ≤ Temperature ≤ 100°F,

   Open gate

   Display “Access Granted” (optional)

Else,

Display “Body temperature high” on LCD

Do not open gate

8. Stop

6. HARDWARE AND SOFTWARE REQUIREMENT

Hardware Description

1. Raspberry Pi (likely Pi 3 or 4) acts as the central controller

2. COMPIM (COM Port Interface Module): Acts as a serial communication interface between Proteus simulation and external devices via the PC’s COM port. Use in IoT Project

3. L293D: Motor driver IC that allows control of DC motors in both directions. Use in IoT Project: Useful if the mask detection system is integrated with gates or automated doors for entry control.

4. LM016L (16x2 LCD Module): Displays alphanumeric data such as messages, status, or results. Use in IoT Project: Can display "Mask Detected" or "No Mask" messages in real-time.

5. LM35: Analog temperature sensor that outputs voltage proportional to temperature in °C Use in IoT Project

6. MCP3208: 8-channel, 12-bit ADC (Analog to Digital Converter) using SPI protocol. Use in IoT Project. Converts analog sensor outputs (like LM35) to digital data for Raspberry Pi, which does not have built-in ADC.

7. MOTOR: Mechanical actuator that converts electrical energy to motion. Use in IoT Project. Can be used for opening/closing gates, barriers, or automated entry systems based on detection results.

Software Description

Proteus Software:

Key Features:

- Circuit Simulation – Allows real-time simulation of analog, digital, and mixed-signal circuits.
- Microcontroller Simulation – Supports popular controllers like Arduino, PIC, ARM, AVR, and 8051.
- PCB Design – Includes tools for schematic capture and PCB layout design.
- Error Detection – Helps identify wiring mistakes and faulty logic before implementation.

Python IDLE

Key Features:

- Interactive Shell – Runs Python commands line-by-line for quick testing.
- Code Editor – Provides syntax highlighting, auto-indentation, and code completion.
- Debugger – Supports breakpoints and step-by-step code execution. Cross-Platform – Works on Windows, macOS.

OpenCV library:

In a face mask detection system using Raspberry Pi, OpenCV plays a vital role by enabling the device to process and analyse images or video streams in real time. It is primarily used for capturing video frames from a camera connected to the Raspberry Pi and for detecting faces within these frames using pre-trained classifiers such as Haar Cascades or deep learning-based detectors.

VSPE (Virtual Serial Port Emulator)

Key Features:

- Virtual COM Ports – Simulates hardware serial ports for software testing.
- Data Transfer Testing – Helps test serial communication between applications.
- Port Splitting – Allows multiple programs to share the same serial device.
- Debugging Tool – Useful for IoT, embedded systems, and serial-based applications.

7. RESULTS:

7.1 Simulation by using Proteus Software

Fig.3 shows Hardware interface in Proteus Software.

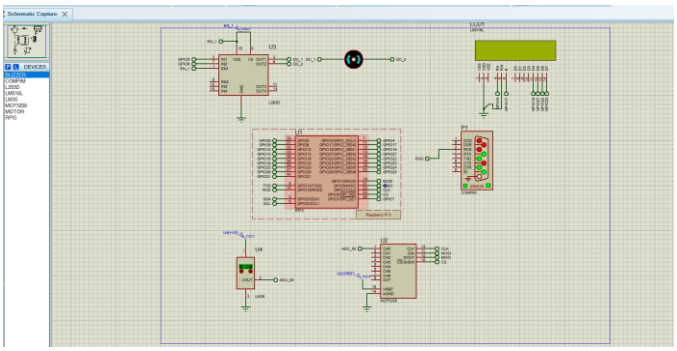

Fig.3: Hardware interface in Proteus Software

The simulation process including the following steps:

1. Initialization

- Camera starts capturing live video.
- Pre-trained ML/DL model (CNN, Mobile Net etc.) is loaded on Raspberry Pi.

2. Face Detection

Using OpenCV Haar Cascade or DNN face detector, the system detects faces in the video frame.

3. Mask Classification

The detected face is cropped and passed into the trained model.

Model predicts two classes:

- With Mask
- Without Mask

7.2 virtual port creation

Figure 4 shows the Virtual Serial Ports Emulator (VSPE) interface, which is used to create and manage virtual COM ports for testing and data communication between application without physical serial hardware.

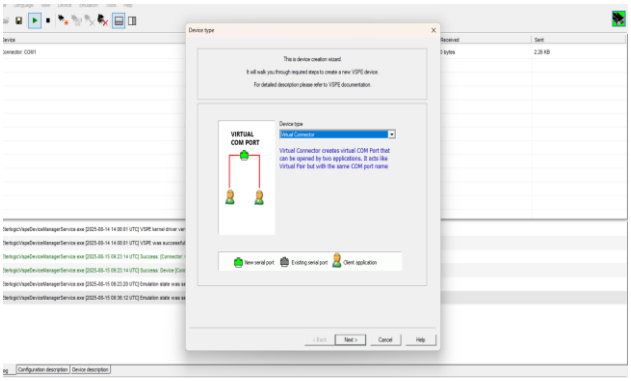

Fig.4: Virtual Port Creation

Virtual COM Port Creation include following steps: -

- VSPE allows you to create pairs of virtual COM ports (ex: COM1).
- The Applications Communication (e.g., Proteus simulation or Raspberry Pi emulator) sends data to COM1.
- This simulates real UART communication without physical hardware.

7.3 Run the Simulation

Figure 5 shows the simulation process as run in the proteus software.

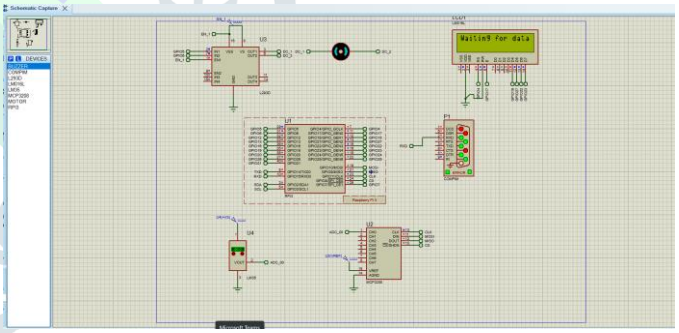

Fig.5: Start the Simulation Process

On starting the simulation, Proteus emulates the Raspberry Pi processing input from the virtual camera. Based on the detection results, the corresponding output is displayed.

7.4 Start the Video Streaming

Figure 6 shows the capturing the image, once the simulation process is start on the proteus software and video streaming is start on the python window.

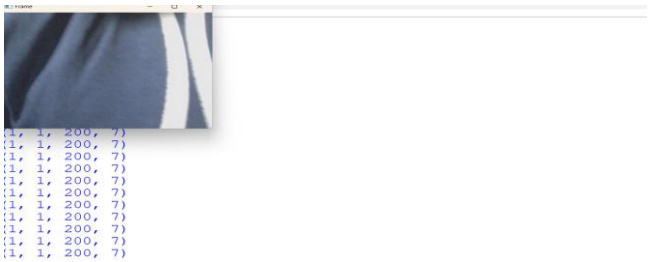

Fig.6: Capturing the image

Video streaming including the following steps:

1. Capture Real-Time Input

The camera continuously streams live video frames, which act as input for the face detection and mask detection algorithms.

2. Frame-by-Frame Processing

Each frame of the video is analysed in real time using machine learning or deep learning models (for example, CNN models with OpenCV, TensorFlow). The system checks if a person is wearing a mask or not.

3. Instant Feedback

Based on the detection result, the system provides immediate feedback such as displaying messages ("Mask Detected" / "No Mask"), triggering alerts, or controlling hardware (like allowing/denying entry through a gate).

7.5 Detecting the without Mask

Figure 7 shows capture without mask, it refers two individual persons are visible, and the system has drawn red detection boxes around their faces. The red colour visually indicates a negative classification(unmasked).

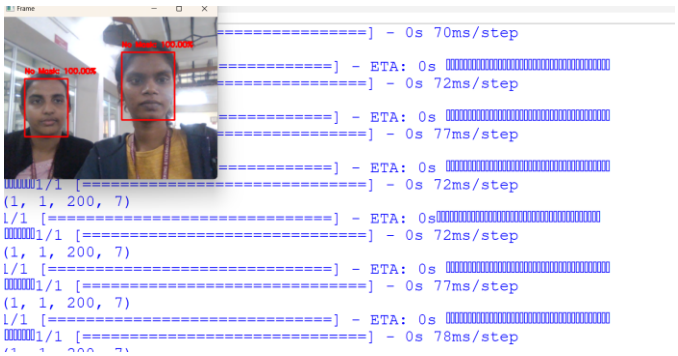

Fig.7: Capture without mask

7.6 Detecting with Mask

Figure 8 shows capture with mask. The system accurately identifies two individuals wearing face mask. The compliant state is confirmed by green bounding boxes over their masked faces.

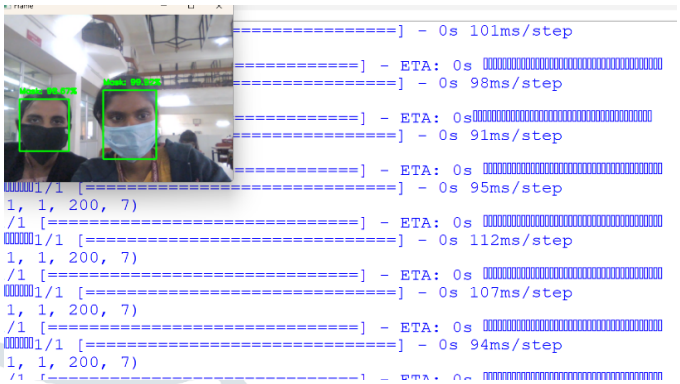

Fig.8: Capture with mask

8. LIMITATION

1. Limited Processing Power

- Struggles with large AI models and high-resolution video streams.

2. Lighting Sensitivity

- Performance decreases in low-light or overexposed environments.

3. Limited Multi-Person Tracking

- Accuracy drops when detecting multiple people in a crowded scene.

4. Storage Constraints

- Raspberry Pi has limited onboard storage for large datasets or video logs.

5. Camera Quality Dependency

- Detection performance depends on resolution and clarity of the camera feed.

6. Internet Dependency for Cloud-Based Processing

- Cloud-based AI requires stable internet for operation.

9. CONCLUSION

Smart Monitoring System that integrates face mask and body temperature detection is built using Raspberry pi platform and deep learning R-CNN algorithm. The system employs a camera module with deep learning algorithms to automatically detect whether individuals are wearing a face mask. Simultaneously, a temperature sensor is interfaced with the Raspberry Pi to measure body temperature in real time. This automated Solution offers a contactless, reliable, and cost-effective approach for screening individuals in public spaces such as school, offices, hospitals and transportation hubs

10. REFERENCES

[1] A. Kumar, A. Kaur, M. Kumar," Face detection techniques: A review," Artificial intelligence review, volume.52 no.2pp.927928,2019. D.H. Lee, K.L. CHEN, K. Liou, C. Liu, and J. Liu," Deep learning and control algorithms" of direct perception for autonomous driving,2019.

[2] F. Rustam, A. A. Reshi, A. Mehmood et al., "COVID-19 Future Forecasting Using Supervised Machine Learning Models," IEEE Access, vol. 8, Article ID 101489, 2020

[3] N. Sindhwani, V. P. Maurya, A. Patel, R. K. Yadav, S. Krishna, and R. Anand, "Implementation of intelligent plantation system using virtual IoT," in IOT and its Applications, pp. 305–322, Springer, Cham, Switzerland, 2022.

[4] D. Singh, V. Kumar, M. Kaur, M. Y. Jabar Ulla, and H.-N. Lee, "Screening of COVID-19 suspected subjects using multicores over genetic algorithm based dense convolutional neural network," IEEE Access, vol. 9, Article ID 142566, 2021.

[5] Guang Cheng wang, Yumiko, "Masked face recognition data sets and application", National natural science foundation of China, 2020

[6] Amit Chavda, Jason Dsouza, SumeetBadgujar Multi-Stage CNN Architecture for Face Mask Detection September 2020

[7] Amrit Kumar, Bhadani, Anurag Sinha A Facemask detector using machine learning and image processing techniques November 2020 Engineering science And technology and international conference.

[8] Sammy v. militant, Nanette. Dionisio *Real time face mask*

recognition with alarm system using deep learning 2020 11th IEEE control and system graduate research colloquium

[9] Mohammad macroform. Motalebhossenmd. Milon Islam Sai fuddinmahmud A automated system to limit covid 19 using facial mask detection in smart city network 2020 IEEE international IOT

[10] Toshanlalmeenpal, Ashutosh Balakrishnan, Amit Verma, Face mask detection using semantic segmentation 2019,4th international conference on 4th International computing, communications and security (ICCCS)

[11] Wenyu sun, Yulong, Changsheng, Face spoofing detection based on local ternary label supervision in fully convolutional networks IEE transactions on information forensic and security 2020