

FACE MASK DETECTION USING CONVOLUTIONAL NEURAL NETWORKS

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

**ROHAN KUMAR K (Reg.No - 39110852)
RITHIK JOSEPH G (Reg.No – 39110850)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119**

APRIL - 2023



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **ROHAN KUMAR K (REG NO: 39110852)** and **RITHIK JOSEPH G (REG NO: 39110850)** who carried out the Project entitled "**FACE MASK DETECTION USING CONVOLUTIONAL NEURAL NETWORKS**" under my supervision from January 2023 to April 2023.

Internal Guide

Ms. NITHYA S , M.E.

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on

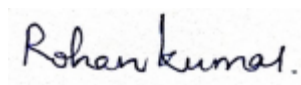
Internal Examiner

External Examiner

DECLARATION

I, **ROHAN KUMAR K(Reg.No- 39110852)**, hereby declare that the Project Report entitled “**FACE MASK DETECTION USING CONVOLUTIONAL NEURAL NETWORKS**” done by me under the guidance of **NITHYA S , M.E.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 25-04-2023

A handwritten signature in blue ink that reads "Rohan Kumar." The signature is written in a cursive style with a period at the end.

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **NITHYA S , M.E.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TABLE OF CONTENTS

ABSTRACT	viii
LIST OF FIGURES	vii

CHAPTER NO	TITLE	PAGE NO
1	INTRODUCTION	1
2	AIM AND SCOPE OF THE PROJECT	3
	2.1 AIM OF THE PROJECT	3
	2.2 SCOPE OF THE PROJECT	3
3	LITERATURE SURVEY OF PROJECT	4
	3.1 LITERATURE SURVEY	4
	3.2 INFERENCES FROM LITERATURE.	11
4	PROPOSED METHODOLOGY	12
	4.1 SOFTWARE REQUIREMENTS	12
	4.2 HARDWARE REQUIREMENTS	12
	4.3 PYTHON	12
	4.4 APPLICATIONS OF PYTHON	12
	4.5 FEATURES OF PYTHON	13
	4.6 LIBRARIES	16
	4.6.1 NUMPY	16
	4.6.2 OPEN CV	16
	4.6.3 TENSOR FLOW	16
	4.6.4 KERAS	17
	4.6.5 PANDAS	17

5	PROJECT DESIGN AND IMPLEMENTATION	18
	5.1 EXISTING SYSTEM	18
	5.2 PROPOSED SYSTEM	18
	5.2.2 DATASET	19
	5.2.3 DATA PRE-PROCESSING	19
	5.2.4 TRAINING OF CNN MODEL	19
	5.2.5 DEPLOYMENT	20
	5.2.6 CNN ALGORITHM FOR PERSON DETECTION	20
6	RESULTS AND DISCUSSIONS	21
	6.1 RESULTS	21
	6.2 Future Scope	22
7	CONCLUSIONS REFERENCE AND APPENDIX	25
	7.1 CONCLUSION	25
	7.2 REFERENCES	26
	7.3 APPENDIX	28
	A. SOURCE CODE	28
	B. SCREENSHOTS	34
	C. PUBLICATION	36

LIST OF FIGURES

FIGURE NO	NAME	PAGE NO
1.1	System Architecture of face mask detection	2
5.1.1	Flowchart	18
6.1.1	Testing with mask	21
6.1.2	Testing without mask	22
6.2.1	Detecting Masks in public Places	23
6.2.2	Detecting Masks	23
6.2.3	Description of masks properly	24
6.2.4	Detection of masks	24
7.3.1	Detection of masks	34
7.3.2	Detection of masks without mask	34
7.3.3	Detection of masks with mask	35

ABSTRACT

Corona virus sickness has become a big public health issue in 2019. Because of its contact-transparent characteristics, it is rapidly spreading. The use of a face mask is among the most efficient methods for preventing the transmission of the Covid-19 virus. Wearing the face mask alone can cut the chance of catching the virus by over 70%. Consequently, World Health Organization (WHO) advised wearing masks in crowded places as precautionary measures. Because of the incorrect use of facial masks, illnesses have spread rapidly in some locations. To solve this challenge, we needed a reliable mask monitoring system. Numerous government entities are attempting to make wearing a face mask mandatory; this process can be facilitated by using face mask detection software based on AI and image processing techniques. For face detection, helmet detection, and mask detection, the approaches mentioned in the article utilize Machine learning, Deep learning, and many other approaches. It will be simple to distinguish between persons having masks and those who are not having masks using all of these ways. The effectiveness of mask detectors must be improved immediately. In this article, we will explain the techniques for face mask detection with a literature review and drawbacks for each technique.

Keywords: Corona virus disease 2019, Face mask detection, CNN, Object Detection, Keras, TensorFlow.

CHAPTER 1

INTRODUCTION

As the COVID-19 (Coronavirus) pandemic continues to spread, most of the world's population has suffered as a result. COVID-19 is a respiratory disorder that results in severe cases of pneumonia in affected individuals. The disease is acquired via direct contact with an infected person, as well as through salivation beads, respiratory droplets, or nasal droplets released when the infected individual coughs, sneezes, or breathes out the virus into an airspace. Globally, thousands of individuals die from the COVID-19 virus daily.

A Coronavirus (COVID-19) report by the World Health Organization (WHO) reveals that, as of 22 November 2021, there were 258 million confirmed cases of COVID-19 cases and 5,148,221 deaths worldwide. Therefore, people should wear face masks and keep a social distance to avoid viral spread of disease. An effective and efficient computer vision strategy intends to develop a real-time application that monitors individuals publicly, whether they are wearing face masks or not. The first stage to identify the existence of a mask on the face is identifying the face. This divides the whole procedure into two parts: face detection and mask detection on the face.

Computer vision and image processing have an extraordinary impact on the detection of the face mask. Face detection has a range of case applications, from face recognition to facial movements, where the latter is required to show the face with extremely high accuracy. As machine learning algorithms progress rapidly, the threats posed by face mask detection technology still seem effectively handled. This innovation is becoming increasingly important as it is used to recognize faces in images and in real-time video feeds.

However, for the currently proposed models of face mask detection, face detection alone is a very tough task. In creating more improved facial detectors, following the remarkable results of current face detectors, the analysis of events and video surveillance is always challenging.

Recent years have seen the rise of big data as well as a dramatic rise in the capabilities of computers that use parallel computing. Target detection has become an important research area in computer vision and is also extensively utilized in the real world. For instance, traditional techniques for targeting, like face recognition, autonomous driving, and even target tracking, employ artificially extracted features; however, there are some issues that include incomplete feature extraction, and a weak recognition effect. With the introduction of convolutional neural networks, significant advances have already been made in the field of image classification.

In recent years, authors have used predefined standard models like VGG-16, Resnet, MobileNet, which require large memory and computational time. In this paper, an effort was made to customise the model in order to reduce memory size, computing time, and boost the accuracy of the model's findings. This paper presents a face mask detection system based on deep learning. The presented approach can be used with surveillance cameras to detect persons who do not wear face masks and hence restrict COVID-19 transmission.

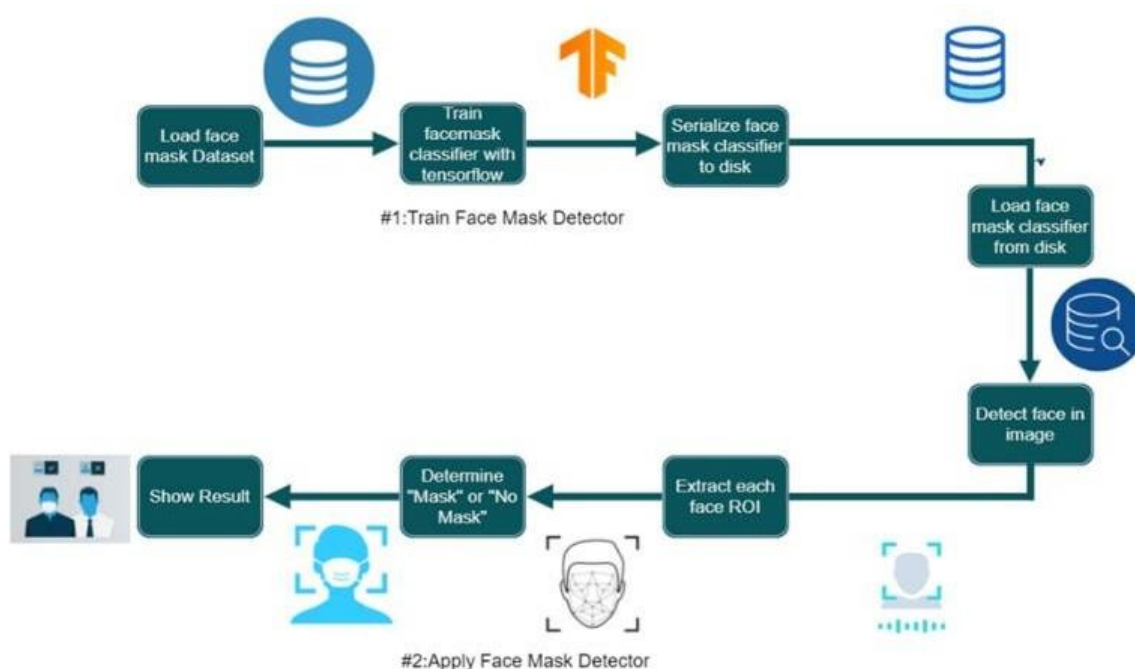


Fig 1.1 System Architecture of face mask detection

CHAPTER 2

AIM AND SCOPE OF THE PROJECT

2.1 AIM

The main aim is to create awareness and to signify the importance of wearing a mask which can prevent from getting infected with any communicable diseases.

2.2 SCOPE

Any system always has a scope for improvements and more advancement. All the systems studied under the literature survey have their own different features. All the systems proposed till date are used for better accuracy but with different drawbacks. There could be such a system where color of the mask can be identified. Here in this system many advanced features can be added according to the requirement .

CHAPTER 3

LITERATURE SURVEY

3.1 LITERATURE SURVEY

3.1.1 TITLE 1 : Deep learning, OpenCV, TensorFlow, as well as Keras are used in the study strategy to aid in the faces detection having masks.

Author : S.Shivaprasad

With assistance of this technique, safety is ensured. The technique for face detection has employed the MobileNetV2 and CNN framework as classifier; it is lightweight, has fewer parameters, which may be utilized in embedded devices (Onion Omega2 and Raspberry Pi) to perform authentic mask identification. The accuracy of the approach utilized in this study is 0.96, and the F1-score is 0.92. The data was gathered from a variety of sources and several scientists can use it to create more sophisticated models, like face recognition, facial patterns, and facial characteristics for detection method.

Pros : Got high accuracy by collecting data from different sources.

Cons : Issues when applied system with real time.

3.1.2 TITLE 2 : contemplate developing a detection system for face masks connected with digital healthcare services.

Author : Prathmesh Deval

image pre-processing. For face detection, Haar-Cascade will be used, as it is a very effective face detection method. Figure 4 depicts the system's design, which demonstrates how it works automatically to avoid the expansion of COVID19. The research uses advanced learning algorithms to recognize various facial features and determine whether or not person is employing the face mask. The system deals with Face Mask detection in real-time and also helps in reducing the transmission rate. The

system also provides few digital facilities for receptionists and doctors.

Pros : 1. Proposed systems can be implemented in many places.

2. System work in real-time helps in reducing the transmission rate.

Cons : need more research to ensure that the person put make on their faces or something else.

3.1.3 TITLE 3 : Proposed model was trained on a database of over 11,000 images of faces either with or without masks, employing numerous deep learning techniques.

Author : Jansi Rani Sella Veluswami.

A SSDNET model is being deployed for face detection, the output of which is passed to a custom-made Lightweight CNN for mask detection. On two distinct testing datasets, the model obtains a remarkable accuracy of ~ 96% and ~ 96%. The model will help government agencies and health officials fight the global pandemic.

Pros : 1. With many datasets testing getting high accuracy

2. Model will help government agencies and health officials fight the global pandemic.

Cons : Need more research to apply with live video or with security camera.

3.1.4 TITLE 4 : Contextually Multi-Scaled Region-focused Convolutional Neural Networks (CMS-RCNNs) to address various hard issues such as significant facial occlusions, incredibly lower resolutions, intense illumination, highly pose changes, video or image compression errors.

Author : Chenchen Zhu.

Proposed networks, like region-based CNNs, have a component of regional proposal and a region-of-interest (RoI) detecting components. However, outside of those networks, two major shares for improving state-of-the-art facial recognition performance.

The multi-scaled information is aggregated including both feature map and RoI traced to conciliation with narrower face region. Secondly, our suggested network, which is based on intuition of the human vision system, permits explicit body spatial reasoning. By combining WIDER FACE Datasets with Face Detection Datasets and Benchmarking, and the Face Detection Dataset as well as Benchmark, we were able to create a dataset with a high degree of variability (FDDB). The investigation results reveal that the suggested methodology, when trained on the WIDER FACE Dataset, consistently Upon that WIDER FACE Data, it surpasses strong benchmarks and consistently produces competitive results on FDDB when compared to latest state face detection approaches. One of R-shortcomings CNN's is that each image must classify 2000 region recommendations.

Pros : 1. For dealing with small face regions multi-scaled content id grouped including both regional proposal and ROI detection.

2. Inspired by the intrusive of the human vision system our suggested network provides intrusive body contextual reasoning.

Cons : Further research in needed to enhance fully joint training such that network may be trained from beginning to end.

3.1.5 TITLE 5 : proposed deep cascade convolutional network that use Fast-R-CNN, in the research they used Detection Date Set and Benchmark.

Author : K. Wang, 2016

proposed deep cascade convolutional network that use Fast-R-CNN, in the research they used Detection Date Set and Benchmark (DDSB) and Annotated Face in-the-

Wild (AFW) as a dataset for testing part. For image input, the first stage starts by Lower Stage Classification Networks (L-Cls-Net) which scan the entire image in various scales with the 95% reject of detection windows, resize image and put it in Lower Stage Calibration Networks (L-Cal-Net) for size adjustment and nearby faces. For high abandon place those into Higher Stage Classification Networks (H-Cls-Net), afterwards adjust by Higher stage Calibration Networks (H-Cal-Net) for spatial locations uses Fast-R-CNN Networks (FR-Net).

Pros : 1. Higher recall of 91.87 on the challenging FDDB benchmark.
2. Out performing the state-of-the-art methods.

Cons : By using non-maximum suppressing after L-Cal-Net. You can go very slowly.

3.1.6 TITLE 6 : Suggested an existing road proposed detection system using CNN, detection system comprises of two phases.

Author : Rongqiang Qian

first, there are a huge amount of candidate zones that can comprise intended objects and all these objects shall be passed to Fast R-CNN having the process. The dataset was recorded by set up camera on the vehicle having a pre-determined angle of shooting. The recordings were done with 1280×720 pixels resolution. We decoded movies and gathered all of frames with traffic indications. There are entire numbers of 2223 images taken, with 4204 road signs incorporated.

The results of experiments shows that the Fast R-CNN either having or without having regression with bounding boxes achieves the maximum recall rates of 90.73% as well as 88.33%, respectfully, whereas the comparable rates of accuracy are 14.49% and 71.23%. When compared to the region proposal stage recall rate of 89.74%, Fast R-CNN using regression of bounding box improves by 0.99%, while Fast R-CNN with no regression loses of bounding box 1.41% positive sectors.

Pros : 1. A hybrid region suggestion approach that takes into consideration color and edge complementary information.

2. Rapid R-CNN for classification with bounding box regression at the same time.

Cons : Selective searches not adopted in system.

3.1.7 TITLE 7 : suggested face identification algorithm relying on Fast R-CNN that uses three algorithms to discover the candidate area of such face that might exists in image.

Author : Qi hang Wang 2018

The candidate zone is fed into trained convolution neural network, which produces a final convolution attribute (ROI) depending on Fast R-CNN network architecture after a sequence of convolution and pooling procedures. The ROI is then fed into the two full interconnections (ROI).

Pros : 1. To retrieve the feature autonomously, this method efficiently use deep convolution network.

2. Eliminate classic face detection model's reliance on manual attributes.

Cons: 1. The requirement for a high number of sample in order to construct.

2. The use of selective search object concept extraction procedure takes longer.

3.1.8 TITLE 8 : Recognize human faces autonomously, it uses sophisticated deep learning algorithm relying on machine vision.

Author : Lin Jiang

A multiscale rapid RCNN procedure depending on upper or lower layers (UPL-RCNN) is suggested to purely recognize a range of human faces. The +e network is made up of components that perform spatial affine transformations and feature region constituents (ROI). Face detection relies heavily on this technology. To begin with, multiscale data can be bundled in detection to handle small portion of face. The approach may then execute contextual sophismand spatial transformations, such as zooming, trimming, and rotating, using the inspirations of the visual perception. The results of comparative studies reveal that this technology can not only accurately recognize different faces but also outperforms fast R-CNN in terms of performance. The drawbacks with FAST R-CNN most time taking during detection is done by the selective search region.

Pros : High precision time consumption is reduced. There is no correction mark.

Cons : Not only the approach effectively recognize human faces, but it can also do quickly.

3.1.9 TITLE 9 : Suggested Faster-R-CNN, yet They've switched to a region proposal network as a replacement for the proposal approach (RPN).

Author : Ren et. Al

The RPN is just a fully convolutional (FCN) network that gets every size image as an input as well as outputs a bunch of rectangular aspirant object recommendations. Every object concept has an objectness scoring rate that determines whether or not the proposal comprises an object. Figure 8 depicts the structure of FASTER-R-CNN.

Pros : Good performance on small object. Better in five types of chosen targets that YOLOv3-Tiny within lower speed

Cons: The proposed network's performance on small target is quite poor.

3.1.10 TITLE 10 : Faster R-CNN has been suggested. It is comprised of two modules. Regional Proposal Networks (RPN), first is a fully convolutional networks that generates item suggestions for the second module.

Author : Huaizu Jiang

the Faster R-CNN has been suggested. It is comprised of two modules. Regional Proposal Networks (RPN), first is a fully convolutional networks that generates item suggestions for the second module. The Fast-R-CNN detector seems to be second one, and its aim is to refine ideas. They employed the WIDER face database, which comprises of 12,880 images with 159,424 faces. On three benchmark datasets, the outcome indicates cutting-edge face identification performance, and through RPN, several convolutional layers can be used without increasing the computational overhead.

Pros : Designed to detect a wide range of objects.

Cons: Additional research is needed into the unique features of human faces.

3.1.11 TITLE 11 : For generating region proposals, Mask R-CNN uses exactly regional proposal networks (RPN) by Faster-R-CNN

Author : Kaiming He

For generating region proposals, Mask R-CNN uses exactly regional proposal networks (RPN) by Faster-R-CNN. Instead of using a RoI pooling layer, the researchers used a RoI Align layer on region suggestions to align retrieved features with object's input position. After that, the aligned Rols are input into the Mask R-final CNN's stage, which produces three outputs: a bounding box offset, a class label, and

binary object mask. Each RoI is masked using a tiny fully convolutional neural network

Pros : Simple to train and adds only a small overhead to Faster R-CNN Running at 5 fps

Cons: Need more research for optimization.

3.2 INFERENCES FROM LITERATURE SURVEY

The various literature surveys describe various methodologies which involves numerous CNN models. Since mask detection is purely image segmentation, CNN model serves as a very good base for recognition and detection. In some specific cases, combination of CNN models is used which in order to provide more accuracy. The idea of CNN combination is a unique concept and has a wide scope for many complex problems. While the mentioned surveys are unique and well-performing in many aspects they however, do not meet the expectations due to the following reasons:-Compromising computation time for accuracy:0Computation time is an important and an essential feature for detection, it is very important for a camera to make sound and accurate detections. The use of high-end models that offer high accuracy, drastically effects the run time, and ultimately depletes the computation time.

CHAPTER 4

PROPOSED METHODOLOGY

4.1 SOFTWARE REQUIREMENTS

Hardware Machine Learning Libraries : TensorFlow, OpenCV

Scripting language : Python

Database : MongoDB / Cloud Storage (AWS)

Platform : Linux , Windows

API tools : Coco API/ YOLOv2 API

Dataset : Mnist , CIFAR-10

4.2 HARDWARE REQUIREMENTS

The hardware requirements are needed to serve as the basis for implementation of the system and hence should be an absolute and coherent specification of the entire system. The software engineers use the hardware requirements as the starting point for the system design. It indicates what the system performs and what the system should execute.

Hardware: 8GB RAM, 6GB Graphics, i7 Processor

Graphics Driver: NVIDIA GTX 1050

4.3 PYTHON

Python is often used as a support language for software developers, for build control and management, testing, and in many other ways. SCons for build control. Buildbot and Apache Gump for automated continuous compilation and testing. Roundup or Trac for bug tracking and project management.

4.4 APPLICATIONS OF PYTHON

The major applications of java include:

- Desktop applications
- Database connection

- Web servers and application servers
- Web applications
- Games
- Mobile applications

4.5 FEATURES OF PYTHON

The important features of java are:

Easy to Code

Python is a very high-level programming language, yet it is effortless to learn. Anyone can learn to code in Python in just a few hours or a few days. Mastering Python and all its advanced concepts, packages and modules might take some more time. However, learning the basic Python syntax is very easy, as compared to other popular languages like C, C++, and Java.

Let us now look at the next feature of python.

Easy to Read

Python code looks like simple English words. There is no use of semicolons or brackets, and the indentations define the code block. You can tell what the code is supposed to do simply by looking at it.

● **PLATFORM INDEPENDENT**

Unlike other languages like C, C++, Python is platform independent as it can be run on multiple platforms. It is also called "code once, run anywhere." A program runs on hardware or software called a platform.

There are Software based and Hardware based platform environments in which Java provides Software based environments.

The Python platform is different from most other platforms. It is a software-based

platform that runs on top of other hardware-based platforms. Java platform has two components:

1. Runtime Environment
2. API(Application Programming Interface)

Python can be run on multiple platforms. It can be compiled and converted as bytecode, which is a platform independent code.

● **SECURED**

Python can be used to develop various virus-free systems and is more secure because of no explicit pointer and virtual machine sandboxes are used to run the programs.

Class loader is used to load classes into Java Virtual Machine dynamically. This adds security due to the separation of classes from the local file system from those that are imported from network sources. Bytecode verification is used to check the code fragments for illegal code that violate access rights to objects. Security manages helps to determine the resources of a class which has access such as reading and writing to local disk.

● **ROBUST**

The meaning of Robust is strong. Python is robust because:

- Python applies well-built memory management techniques.
- Python consists of exception handling as well as type checking.
- By using Python Virtual Machine, the objects not used by the Java Application can be eliminated. Python also provides automatic garbage collection.
- Security problems can't be avoided easily due to lack of pointers.

● **ARCHITECTURE NEUTRAL**

No implementation dependent features make java architecture neutral. This leads to fixed size of primitive types. In Python, 4 bytes are required for both 32 and 64-bit architectures whereas, in C programming, INT data type occupies 2 bytes of memory

for 32-bit architecture.

- ***PORTABLE***

Python is portable as it facilitates the user to carry the JPython bytecode to any platform. Java doesn't require any implementation.

- ***HIGH PERFORMANCE***

Python is faster compared to traditional interpreted languages. It is because the byte code is “close” to native code. This is a bit slower than compiled languages like C, C++, etc.

- ***DISTRIBUTED***

Python is distributed as it allows users to create distributed applications in Java. RMI and EJB are implemented for creating distributed applications. This feature of Java allows the users to access files by calling the methods from any machine on the internet.

- ***MULTI THREADED***

Thread executes concurrently like a separate program. Multiple threads are used to deal with many tasks at the same time. It does not occupy memory for each thread as they share the common memory area. Threads are important for web applications, multi-media etc.

- ***DYNAMIC***

Python is also known as a dynamic language as it supports the dynamic loading of classes. Dynamic loading means that on demand, classes are loaded. The functions from its native languages are also supported. Finally, automatic memory management as well as dynamic compilation is also supported.

4.6 LIBRARIES

4.6.1 NUMPY

NumPy- Also known as numerical Python, is a library used for working with arrays. It is also a general-purpose array-processing package that provides comprehensive mathematical functions, linear algebra routines, Fourier transforms, and more. NumPy aims to provide less memory to store the data compared to python list and also helps in creating n-dimensional arrays. This is the reason why NumPy is used in Python.

NumPy is a python library mainly used for working with arrays and to perform a wide variety of mathematical operations on arrays. NumPy guarantees efficient calculations with arrays and matrices on high-level mathematical functions that operate on these arrays and matrices.

4.6.2 OPEN CV

OpenCV is a Python library that allows you to perform image processing and computer vision tasks. It provides a wide range of features, including object detection, face recognition, and tracking.ENCV.

OpenCV (Open-Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

4.6.3 TENSOR FLOW

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

TensorFlow is an end-to-end open-source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this

class focuses on using a particular TensorFlow API to develop and train machine learning models.

4.6.4 KERAS

Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation.

The final layers in a CNN are fully (densely) connected layers. In Keras, these layers are created using the `Dense()` class. The Multilayer Perceptron (MLP) part in a CNN is created using multiple fully connected layers. In Keras, a fully connected layer is referred to as a Dense layer.

4.6.5 PANDAS

Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data. It is used for data analysis in Python and developed by Wes McKinney in 2008.

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.

CHAPTER 5

PROJECT DESIGN AND IMPLEMENTATION

5.1 EXISTING SYSTEM

Different literature reviews discuss diverse techniques using many CNN models. The CNN model makes a very good foundation for recognition and detection because mask detection is essentially picturing segmentation. Combinations of CNN models are employed in particular specific situations to increase accuracy. The concept of CNN combination is distinct and has a broad application to numerous challenging issues. Even if the aforementioned surveys are exceptional and successful in many ways, they fall short of expectations for the following reasons: -Compromising accuracy for computing speed: In order for a camera to make reliable and precise detections, computation time is a crucial and critical aspect for detection. using sophisticated models with high precision.

5.2 PROPOSED SYSTEM

In order to predict whether a person has put on a mask, the model requires learning from a well-curated dataset, as discussed later in this section. The model uses Convolution Neural Network layers (CNN) as its backbone architecture to create different layers. Along with this, libraries such as OpenCV and Keras are also used. The proposed model is designed in three phases: Data pre-processing, CNN model training and MobileNet.v2

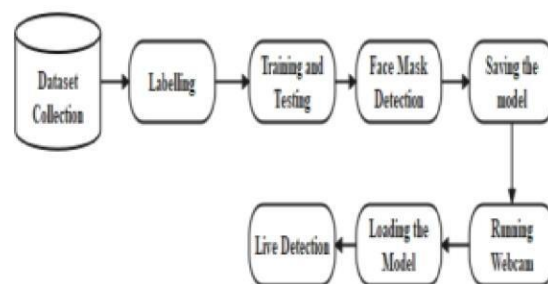


Fig 5.1.1 Flowchart

5.2.2 DATASET

The dataset used is our own collected dataset. The images were taken from each individual with and without wearing face masks. To increase the size of the dataset (for achieving better model performance), some augmentation techniques (like rotating, zooming, and blurring) have been performed on the collected images. The final version of our dataset included images labeled as with mask and without mask. The dataset has dimensions of (480x640) , and its color space is red , green and blue.

5.2.3 DATA PRE-PROCESSING

The accuracy of a model is dependent on the quality of the dataset. The initial data cleaning is done to eliminate the faulty pictures discovered in the dataset. The images are resized into a fixed size of 96 x 96, which helps to reduce the load on the machine while training and to provide optimum results. The images are then labelled as being with or without masks. The array of images are then transformed to a NumPy array for quicker computation. Along with that, the *preprocess input* function from the MobileNetV2 is also used. Following that, the data augmentation technique is utilized to increase the quantity of training dataset and also improve it's quality. A function *ImageDataGenerator* is used with appropriate values of rotation, zoom, horizontal or vertical flip, to generate numerous versions of the same picture. The training samples has been increased to elude over-fitting. It enhances generalization and robustness of the trained model. The whole dataset is then divided into training data and test data in a ratio of 8:2 by randomly selecting images from the dataset. The *stratify* parameter is used to keep the same proportion of data as in the original dataset in both the training and testing datasets.

5.2.4 TRAINING OF CNN MODEL

The classification of a supervised learning CNN model is done after its training to classify the trained images to their respective classes by learning important visual patterns. TensorFlow and Keras are the primary building blocks for the proposed model. In this study, 80% of the dataset contributes to the training set and the rest to the testing set. The input image is pre-processed and augmented using the steps

described above. There is a total of 5 *Conv2D* layers with ReLu activation functions with a 3 x 3 filter and 5 Max-Pooling Layers with a filter size of 2 x 2. Flatten and Dense are used as the fully connected layers. The output layer uses softmax as its activation function. This results in 2,818,658 trainable parameters in this Convolutional Neural Network.

5.2.5 DEPLOYMENT

In the last step the CNN model is integrated into a web-based application that is hosted in order to be shared easily with other users and they can upload their image or live video feed to the model to recognize facial masks and then get the predicted result.

5.2.6 CNN ALGORITHM FOR PERSON DETECTION

So, for analysis, video footage of people wearing and not wearing masks is taken into account. The experiment is set up so that participants must face the cameras as they enter crowded areas, which have cameras positioned at their entrances. In order to capture videos with people's heads properly aligned for a better ability to detect masks, In the videos, cameras with a resolution of full hd and a frame rate of The cameras were positioned at the same level as the 25 frames per second (roughly 6 feet above the ground). Several improvements in CNN's feature extractor (like VGG, Res Net or Mobile Net). For the initial stage of person detection, the CNN-mobilenet-v1-coco model is utilized, with the weights pretrained using COCO data. The COCO dataset, which is frequently used to compare the effectiveness of deep learning techniques, has 80 classes. The model is then changed once again to only recognise the person class.

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 RESULTS

The results are more of what was expected of the model. The mask recognition is implemented using the camera as a medium and shows accurate results. When the persons face is in the camera frame, model will detect the face and a green or a red frame will appear over the face (Fig 6.1.1 , Fig 6.1.2). A person who is not wearing mask will get a red frame over his face in camera while the person who is wearing mask will get a red frame. The result is also visible written on top left of the result frame. A percentage match can also be seen on the top of the result frame. The model works even if the side view of the face is visible to the camera. It can also detect more than one face in single camera frame. Overall, the model shows the accurate results.

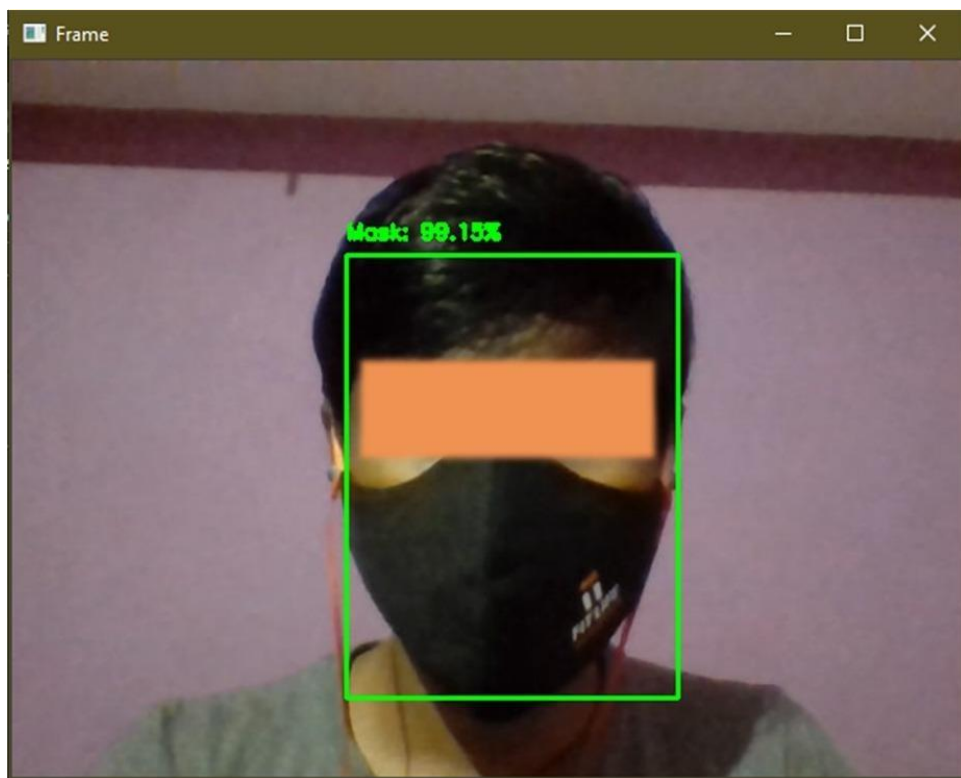


Fig 6.1.1 Testing with mask

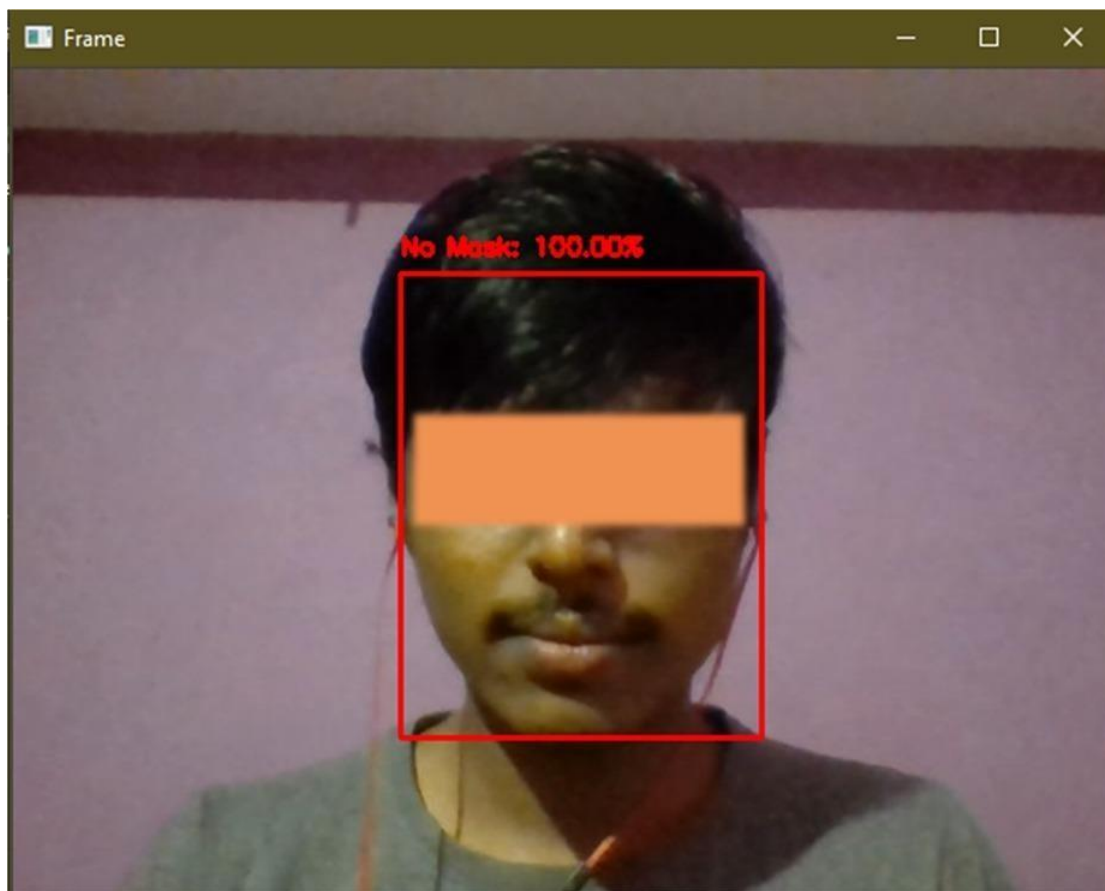


Fig 6.1.2 Testing without mask

6.2 Future Scope

Across the globe more than 70 countries have made it mandatory to wear masks at all public places. If you want to go out you have to cover your face in schools, supermarkets, transports, offices, stores and at all public places. Most retail stores use software to count the total number of customer visit to the store. For this reason, we plan to update the mask recognition system and announce it as an open source project. This system can be implemented in these retail shops and the result can be seen on the digital and promotional screens This app can be used with any current USB, IP, or CCTV cameras for identifying people who do not wear a mask. The live video mask detection function can be introduced in web and desktop applications so that the operator may know whether users are not wearing masks and warning messages can be lost. If anyone isn't wearing a mask, images should be submitted to software operators. Furthermore, we can mount an alarm device that will emit a beep

sound if anyone enters the area without wearing a mask. Only people wearing face masks can enter using this software, which can be linked to the entrance gates. This system could be connected in hospital gates, schools, malls and at many more public places.



Fig 6.2.1 Detecting Masks in public Places

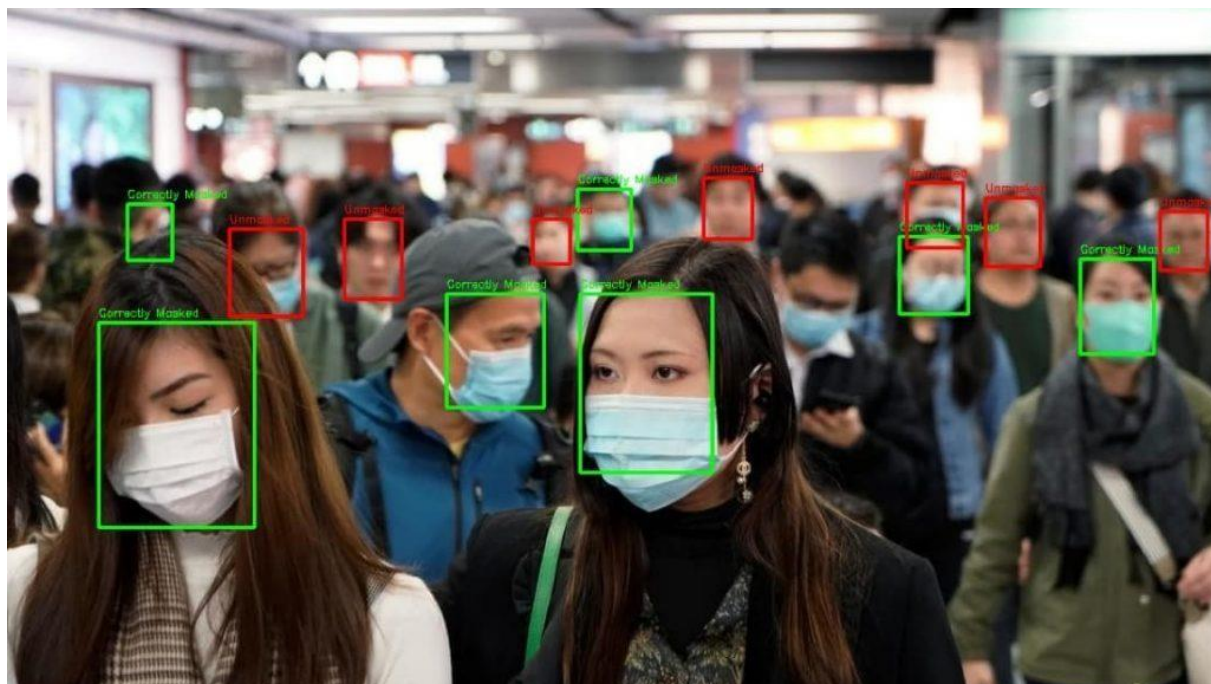


Fig 6.2.2 Detecting Masks



Fig 6.2.3 Description of masks properly



Fig 6.2.4 Detection of masks

CHAPTER 7

CONCLUSION

Measures must be taken to control the spread of the COVID19 pandemic. This face mask recognition system is a very good and efficient way to do so. The system will separate the people from the crowd who are not wearing mask. The identification of people, violating the COVID norms increases the adaptability of the face mask detection system for the public sake. If applied in a correct way, the face mask detection system could be used to make sure our safety and for others too. This approach gives not only helps in achieving high precision but also enhance the face detection tempo considerably. The system can be applied in many areas like metro stations, markets, schools, railway stations and many other crowded places to monitor the crowd and to ensure that everyone is wearing mask. Finally, this work can be used for future researchers and enthusiasts. Firstly, this model can be used in any high-definition camcorders, this will make sure that this model is not limited to only face mask detection system. Secondly, this can be used for biometric scans with a mask on the face.

REFERENCES

1) Z. Zhang, B. Bowes

The future of artificial intelligence (AI) and machine learning (ML) in landscape design: a case study in coastal Virginia, USA

J. Digital Landscape Arch., 2019 (4) (2019), pp.2-9

2) S.Y. Kung, M.W. Mak

Machine learning for multimodality genomic signal processing

IEEE Signal Process. Mag., 23 (3) (2006), pp. 117-121

3) D. Duarte, F. Nex, N. Kerle, G. Vosselman

Satellite image classification of building damages using airborne and satellite image samples in a deep learning approach

ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 4(2) (2018), pp. 89-96

4) P. Gupta, N. Saxena, M. Sharma, J. Tripathi

Deep neural network for human face recognition

Int. J. Eng. Manufact., 8 (1) (2018), pp. 63-71

5) K.J. Bhojane, S.S. Thorat

A review of face recognition based car ignition and security system

Int. Res. J. Eng. Technol., 05 (01) (2018), pp. 532-533

6) S. Mahmud, J. Kim

An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network

(2021), pp. 11-15

7) Sushovan Chaudhury, Manik Rakhra, Naz Memon, Kartik Sau, Melkamu

Teshome Ayana

Breast cancer calcifications: identification using a novel segmentation approach

Comput. Math. Methods Med., 2021 (2021), Article 9905808, [10.1155/2021/9905808](https://doi.org/10.1155/2021/9905808)
13 pages

8) J.T. Sunny, S.M. George

Applications and challenges of human activity recognition using sensors in a smart environment

2 (04) (2015), pp. 50-57

9) D. Bhamare, P. Suryawanshi

Review on reliable pattern recognition with machine learning techniques

Fuzzy Inf. Eng., 10 (1) (2019), pp. 1-16, 10.1080/16168658.2019.1611030

10) T. Meenpal

Facial mask detection using semantic segmentation

2019 4th International Conference on Computing, Communications and Security (ICCCS) (October) (2019), pp. 1-5, 10.1109/CCCS.2019.8888092

APPENDIX:

A. SOURCE CODE

```
from flask import Flask
import os
import cv2
import numpy as np
import wget
import matplotlib.pyplot as plt
import object_detection
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util

app = Flask(__name__)

@app.route('/', methods=['GET'])
def index():
    return "Hello World"

@app.route('/predict', methods=['GET', 'POST'])
```

```

def predict():
    CUSTOM_MODEL_NAME = 'PROJECTS'
    PRETRAINED_MODEL_NAME =
'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
    PRETRAINED_MODEL_URL =
'http://download.tensorflow.org/models/object_detection/tf2/202007
11/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz'
    TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
    LABEL_MAP_NAME = 'label_map.pbtxt'
    paths = {
        'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
        'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
        'APIMODEL_PATH': os.path.join('Tensorflow','models'),
        'ANNOTATION_PATH': os.path.join('Tensorflow',
'workspace','annotations'),
        'IMAGE_PATH': os.path.join('Tensorflow',
'workspace','images'),
        'MODEL_PATH': os.path.join('Tensorflow',
'workspace','models'),
        'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow',
'workspace','pre-trained-models'),
        'CHECKPOINT_PATH': os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME),
        'OUTPUT_PATH': os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'export'),
        'TFJS_PATH':os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'tfjsexport'),
        'TFLITE_PATH':os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'tfliteexport'),

```

```

        'PROTOC_PATH':os.path.join('Tensorflow','protoc')
    }
    files = {
        'PIPELINE_CONFIG':os.path.join('Tensorflow',
'workspace','models', CUSTOM_MODEL_NAME, 'pipeline.config'),
        'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'],
TF_RECORD_SCRIPT_NAME),
        'LABELMAP': os.path.join(paths['ANNOTATION_PATH'],
LABEL_MAP_NAME)
    }
    labels = [{'name':'Hello', 'id':1},
        {'name':'thanks', 'id':2},
        {'name':'GoodJob','id':3},

    ]

    config =
config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'
])

    pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
    with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:
        proto_str = f.read()
        text_format.Merge(proto_str, pipeline_config)

    pipeline_config.model.ssd.num_classes = len(labels)
    pipeline_config.train_config.batch_size = 4
    pipeline_config.train_config.fine_tune_checkpoint =
os.path.join(paths['PRETRAINED_MODEL_PATH'],

```

```

PRETRAINED_MODEL_NAME, 'checkpoint', 'ckpt-0')
    pipeline_config.train_config.fine_tune_checkpoint_type =
"detection"
    pipeline_config.train_input_reader.label_map_path=
files['LABELMAP']

pipeline_config.train_input_reader.tf_record_input_reader.input_pa
th[:] = [os.path.join(paths['ANNOTATION_PATH'], 'train.record')]
    pipeline_config.eval_input_reader[0].label_map_path =
files['LABELMAP']

pipeline_config.eval_input_reader[0].tf_record_input_reader.input_
path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'test.record')]

    config_text = text_format.MessageToString(pipeline_config)
    with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "wb") as f:

        f.write(config_text)

    configs =
config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'
])
    detection_model =
model_builder.build(model_config=configs['model'],
is_training=False)

    ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
    ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-

```

```
5')).expect_partial()
```

```
@tf.function
```

```
def detect_fn(image):#this is for detecting
```

```
    image, shapes = detection_model.preprocess(image)
```

```
    prediction_dict = detection_model.predict(image, shapes)
```

```
    detections = detection_model.postprocess(prediction_dict,  
shapes)
```

```
    return detections
```

```
category_index =
```

```
label_map_util.create_category_index_from_labelmap(files['LABEL  
MAP'])
```

```
cap = cv2.VideoCapture(cv2.CAP_DSHOW)#FOR READING THE  
WEB CAM
```

```
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
```

```
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
```

```
while cap.isOpened():
```

```
    ret, frame = cap.read()
```

```
    image_np = np.array(frame)
```

```
    input_tensor =
```

```
tf.convert_to_tensor(np.expand_dims(image_np, 0),  
dtype=tf.float32)
```

```
    detections = detect_fn(input_tensor)
```

```
num_detections = int(detections.pop('num_detections'))
```

```
detections = {key: value[0, :num_detections].numpy()
```



```

        for key, value in detections.items()}
detections['num_detections'] = num_detections

detections['detection_classes'] =
detections['detection_classes'].astype(np.int64)

label_id_offset = 1
image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
    image_np_with_detections,
    detections['detection_boxes'],
    detections['detection_classes']+label_id_offset,
    detections['detection_scores'],
    category_index,
    use_normalized_coordinates=True,
    max_boxes_to_draw=5,
    min_score_thresh=.8,
    agnostic_mode=False)

cv2.imshow('object detection',
cv2.resize(image_np_with_detections, (800, 600)))

if cv2.waitKey(10) & 0xFF == ord('q'):
    cap.release()
    cv2.destroyAllWindows()
    break if __name__ == '__main__':
app.run(debug=True)

```

B. SCREENSHOTS



7.3.1 DETECTION OF MASKS



(a) Without masks

7.3.2 DETECTION WITHOUT MASK



(b) With masks

7.3.3 DETECTION WITH MASK

PAPER REPORT

Detection of Face Mask using Deep Learning Model from Real-time Data

Rithik Joseph G

Rohan Kumar K

Nithya S

Abstract

Significant developments in image processing and computer vision for face mask identification have been sparked by the COVID-19 epidemic. There are a number of methods for detecting face masks, including a recently proposed one that makes use of Keras, and OpenCV. This model is a useful safety resource that can be implemented quickly. Embedded devices like the NVIDIA Jetson Nano and Raspberry Pi may participate in the real-time mask detection process thanks to this method, which employs the Single Shot Multibox Detector as a face detector and the MobilenetV2 architecture as a lightweight classifier. With an accuracy of 0.9264 and an F1 of 0.93, the model performs admirably. The dataset supplied in this work was obtained from a variety of sources, and may be used by other researchers to construct more complex replicas for tasks including face credit, facial landmarks, and facemask component discovery.

Keywords: - Bottleneck, Data augmentation, Fine tuning, MobileNetV2 Convolutional Neural Network,.

Introduction

After WHO declared COVID-19 a pandemic, researchers have been trying to learn more about the virus. According to [3], many people have worked to slow the virus's spread. There is currently no cure or vaccination for this. Authorities in Indonesia and other countries are counting on measures such as quarantines and mask use in public to slow the spread of the COVID-19 virus. [4]-[6]. Since the New Normal has been established, it is now mandatory for all citizens to wear a face mask whenever they are in public or have any sort of social interaction [7]. Bantul Jogjakarta, DKI Jakarta, and the all have laws requiring citizens to wear a mask whenever they leave their homes, with monetary penalties imposed for those who do not [8]. The second is called "Lebak Banten," and it refers to the government's policy of fining those who don't wear

masks when cleaning facilities marked with a certain sign. The government of Banjarmasin, Kalimantan Selatan, for instance, forces everyone who does not wear a face mask in public to do some sort of physical penalty, such doing push-ups, [9]. Yet, there are challenges for authorities in keeping tabs on a big population with varying habits [10]. The availability of the data promptly and accurately is the first step towards a solution that will allow the authorities to properly control the application of the legislation. To identify mask-wearers from non-mask-wearers, one possibility is to apply automated face mask identification on a regional scale [11, 12]. Artificial neural networks, a network of virtual neurons that can learn complex functions through a series of non-linear transformations, are now widely used for challenging classification tasks like speech recognition and image recognition, thanks to the development of deep learning (DL) techniques. Whereas majority of the research employed shallow neural networks or neural networks with hierarchical features [13], deeper networks have also been deployed. Hence, the potential of deep neural networks to tackle face mask identification has not been fully investigated [14, 15].

In this study, we contemporary a two-stage convolutional neural network (CNN) architecture: the first stage recognises human faces, to label such faces as "Mask" or "No Mask" and draw bounding boxes around them with the corresponding class name. This method was then applied to moving images. The faces are then followed from frame to frame using an object tracking method, making the detections invulnerable to motion blur's background noise. In order to monitor safety breaches, encourage the usage of face masks, and guarantee a risk-free workplace, this system can be combined with a camera of some kind.

2 Works Cited

Naseri et al. [16] provide a novel hybrid optimisation method they call Adaptive Sailfish (ASMFO). Next, the detected face images are fed into a hybrid method called Hybrid ResMobileNet (HResMobileNet)-based classification, where the fine-tuned ASMFO parameters are used to generate trustworthy mask detection results. Nonetheless, the suggested mask identification model using IoT is compared to existing classifiers and typical meta-heuristic algorithms via a variety of criteria derived from three industry-standard datasets. An experimental investigation is conducted to evaluate the proposed framework against other meta-heuristic algorithms and current classifiers. When compared to the SVM, CNN, VGG16-LSTM, ResNet 50, MobileNetv2.

Pham et al. [17] provide a comprehensive strategy for balancing accuracy and detection time by creating a novel face mask dataset and improving the YOLOv5 baseline. Specifically, we improve YOLOv5 by integrating a coordinate attention (CA) module into the original design in two separate ways: using YOLOv5s-CA and YOLOv5s-C3CA. In specifically, we train three distinct models using a Kaggle dataset consisting of 853 pictures labelled as either "without a mask" (NM), "with a mask" (M), or "incorrectly worn mask" (IWM). Experiments show that our enhanced YOLOv5 with the CA module can detect objects in an image in just 8 milliseconds, with an accuracy of 93.9% at mAP@0.5 (up from 87% at baseline) (125 FPS). 7110 images and over 3500 labels for three categories drawn from YouTube videos. Both the YOLOv5-CA we propose and the state-of-the-art detection methods are trained on our 7110 image dataset (i.e. YOLOX, YOLOv6, and YOLOv7). Our data shows that the YOLOv5-CA performs better than previous versions, with a mAP@0.5 of 96.8%. Comparing the new and improved YOLOv5-CA model to previous state-of-the-art works, the results reveal that it achieves higher quality results.

Kumar and Bansal [18] introduced a model called Caffe-MobileNetV2 (CMNV2). For mask identification, we use the MobileNetV2, while the fast feature embedding Caffe model is put to use in the face detector's convolutional architecture. In this research, we add five new layers to the previously studied MobileNetV2 architecture in order to increase the classification accuracy while decreasing the number of required training parameters for face mask detection. Experiments showed that the suggested technique was effective, with a 99.64% success rate on still photos and a high success rate on photographs collected in real time. When compared to other models of its kind, this one has an error rate of only 0.36% while maintaining a perfect accuracy rate of 100%, a recall rate of 99.28%, and a f1-score of 99.64. Although first designed for use on PCs, face mask detection has now found broad usage in mobile devices and artificial intelligence. Biometrics encompasses a wide range of techniques for determining an individual's true identity, including computer-based masked-face identification.

Agarwal et al. [19] developed a framework for face mask identification. The ELM is widely used in the biomedical industry for image processing, as well as classification and regression issues, because of its real-time data processing capabilities. Our research is the first to propose using ELM as a classifier for identifying faces concealed by masks. Some examples of pre-trained DCNNs that are examined for potential use

in feature selection include Xception, Vgg16, Vgg19, ResNet50, ResNet101, and ResNet152. When used as a classifier, ELM improves the accuracy of the ResNet152 transfer learning model. The ResNet152 - ELM hybrid architecture is the best among the selected transfer learning models and demonstrates so when compared to many other classifiers utilised for the face mask detection operation, as shown by the performance assessment through various ablation experiments on testing accuracy. This study is novel in validating ResNet152 + ELM as a framework for real-time face mask recognition..

Ramadhan et al. [20] set out to examine some of the most common designs in use at the time to find the best efficient approach to this issue. ResNet50, VGG11, InceptionV3, EfficientNetB4, and YOLO were some of the models used (You Only Look Once). Based on experimental findings on the MaskedFace-Net dataset, EfficientNetB4 outperformed the YOLOv4 architecture (93.40%), the InceptionV3 design (87.30%), the YOLOv3 architecture (86.35%), ResNet50 (84.41%), VGG11 (84.38%), and the YOLOv2 architecture (78.75%) in terms of accuracy. It's important to remember that YOLO's model was trained on a collection of labelled and preprocessed MaskedFace-Net images. Using pre-trained weights from the COCO dataset, the model was able to complete its training more quickly.

3. Proposed System

In the next part, we will discuss how a deep learning model was used to handle face mask detection in this study. The proposed flowchart is shown in Figure 1.

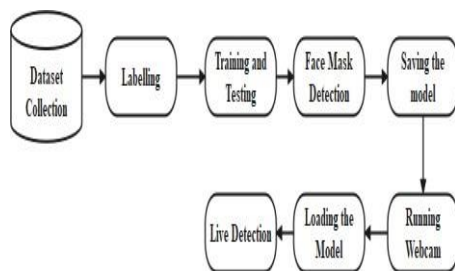


Fig.1. Projected flow Diagram.

3.1. Dataset

Our own data collection serves as the basis for this dataset. Images of all quality levels and backgrounds are shown, from close-ups to long shots to those in which the masks are improperly worn. All the pictures were taken by several people, and from various

vantage points. There are a total of 765 photos in the dataset, including 383 training images, 226 validation images, and 156 test images. The RGB data set has dimensions of (480x640), and its colour space is red, green, and blue.

3.2. Data Pre-Processing

Pre-processing images is an essential step in acquiring high-quality model inputs. As the proposed method acquires the data set from a variety of sources, including photographs with varying levels of light, distance, and blurriness, data augmentation is used to supplement the training set with additional images. This helps with the model's ability to generalise. The image used for training has undergone transformations including rotation, shearing, zooming, and contrast enhancement, among others. After capturing the scene from a variety of angles, we give it a name and a number..

3.3. Proposed fine-tuned MobileNetV2 model

The MobileNetV2 network (https://www.tensorflow.org/api_docs/) was used to categorise the photos in our dataset. With MobileNetV2, there are two distinct building pieces. The first is the residual block with stride 1, both of which are employed in the process of reduction. Figure 2 provides specifics on the MobileNetV2 model.

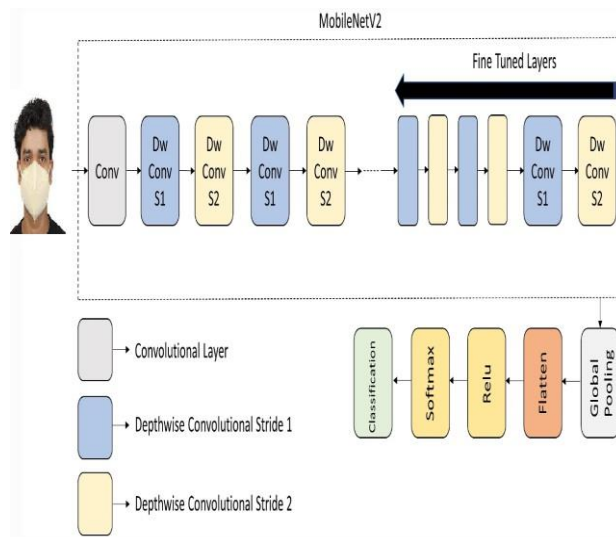


Figure 2: A MobileNet-based CNN perfect with a novel fine-tuning instrument for Face mask finding.

The categorization layer is one of MobileNetV2's 155 total layers. We utilised this approach to carry over data from a similar, completed project. Our proposed model

consists of 154 convolutional base network layers that have already been trained, plus two extra layers for pre-processing and task categorization.

Before we did any fine-tuning, we let the whole model train for 50 epochs. The initial round of tweaking involved opening the remaining 50 layers of the establishing brand-new training loops to run the full model through 80 iterations of training. The second round of fine-tuning began with a step function opening the last convolutional base model layers. With this method, we closed five of the most recently opened fifty layers every eight iterations.

Our third method differs from the previous two in that, rather than completing the fine-tuning procedure in a lessening or rising order, we use a predetermined exponential equation to calculate the sum of epochs and which layers to open based on the outcome. To arrive at these numbers, we utilised the following equation::

$$y = e^{\frac{-2x+9}{8}} + 1$$

where x is a positive integer between 1 and 50 that corresponds to the number of epochs in which the unfrozen top coatings of the basic model are defined, and y is the value determined by fine-tuning. Empirical observation led to the establishment of this reckoning. This may be accomplished by substituting another exponentially diminishing equation.

This formula indicates that the total training time for the system was 80 epochs in the last layers, with a subsequent exponential drop to a value of 1 in the 102nd layer. This means more general information learned during training may be retained by our model. Generally speaking, the earliest layers of CNN models learn generic features like edges, forms, and textures, whereas the later layers of CNN models might have specialised learning attributes. The sum of training cycles required to go from the last layer to a certain depth is exponentially reduced by this equation. In all, we performed 4000 training procedures on Model 1, 2200 on Model 2, and 723 on Model 3. The following settings were utilised during training: Adam as rate of 0.0001..

4. Results and Discussion

Using Label Img. a faster CNN and MobileNet, we assembled a novel dataset for studying face mask identification. We first used v2 to train on the original dataset, then augmented the data and retrained. The Python Keras library was utilised in the

development of the final code. The video was processed with OpenCv to extract frames and facial mask movements. The photos were processed using the numpy package. Figures 3-5 depict the detection of individuals wearing and not wearing masks..



Figure 3. Sample individual non mask person resulted images.

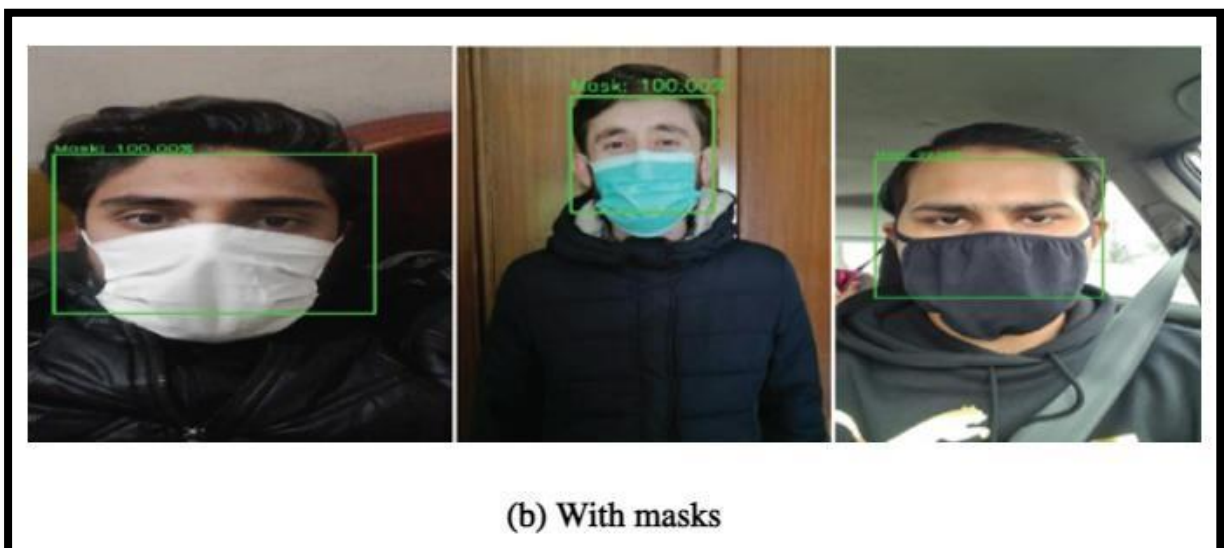


Figure 4. Sample individual with mask person resulted images.



Figure 5. Sample multiple person with mask resulted images.

4.1. Performances evaluation

To assess the effectiveness of the projected approach, several measures are considered, including the confusion, accuracy, recall, and accuracy matrix (ACC) and F1 score. The dimensions used are specified as follows.

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad (2)$$

$$Recall = TP / (TP + FN) \quad (3)$$

$$Precision = TP / (TP + FP) \quad (4)$$

$$F1 - measure = 2 \times (Precision \times Recall) / (Precision + Recall) \quad (5)$$

While TP designates a True Positive, FN a False Negative, you can see how these terms are used in the table below. When dealing with imbalanced data, accuracy and recall are crucial criteria (i.e., F-score). Recall quantifies how many pertinent outcomes were accounted for, whereas accuracy describes how close the actual outcome was to the predicted one. Low FN is shown by a high recall score, whereas low FP is reflected by a high resolution score. Accuracy and recall values that are higher than average suggest that the classifier is successful at retrieving relevant information.

Table 1: Comparative analysis of Projected Model

Method	Recall	Precision	F1-measure	Accuracy

LeNet	71.01	81.15	15.30	96.85
ResNet	76.95	84.58	18.71	97.33
VGGNet	81.06	87.21	11.14	97.90
AlexNet	85.47	90.22	36.42	97.79
DenseNet	89.11	92.45	10.64	95.00
Proposed	91.59	96.34	57.95	98.50

Various pre-trained CNN models are tested with our collected dataset and results are averaged. In the investigation of accuracy, the existing pre-trained models such as LeNet, AlexNet, ResNet, VGGNet and DenseNet achieved nearly 95% to 97%, where the proposed model achieved 98.50%. The LeNet, ResNet and VGGNet achieved nearly 81% to 87% of precision, AlexNet, DenseNet achieved nearly 90% to 92% and proposed model achieved 96.34% of precision. From this analysis, it is clearly proves that the proposed model achieved better performance in predicting the face mask from real-time images. Figure 6 and 7 provides the graphical analysis of proposed model with existing pre-trained models.

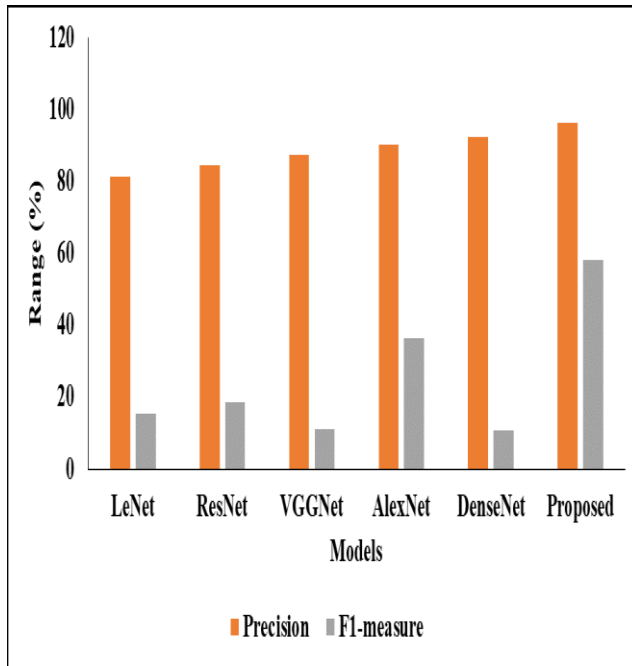


Figure 6: Graphical comparison of proposed model

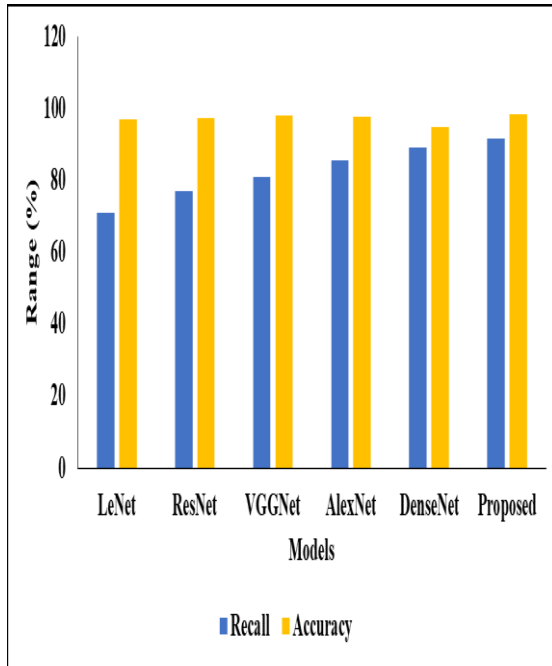


Figure 7: Analysis of projected pre-trained model

5. Conclusion

This work introduced a Facial Mask Detector with two separate phases of operation. After benchmarking against Dlib and MTCNN, the first step employs a pretrained Retina Face model for accurate face identification. We developed a dataset with both masked and unmasked faces so that it would be objective. Three alternative lightweight Face Mask Classifier replicas were trained using the newly produced dataset, and the model was ultimately chosen to classify faces as masked or non-masked using the NASNetMobile framework. In addition, we used Centroid Tracking in our algorithm to boost its functionality with moving images. This technology can be simply deployed for automatic nursing of the usage of face masks at factories, making them safer as the globe attempts to return to normalcy and people resume in-person employment.

References

- [1] Kumar, A., Kalia, A. and Kalia, A., 2022. ETL-YOLO v4: A face mask detection algorithm in era of COVID-19 pandemic. *Optik*, 259, p.169051.
- [2] Chennam, K. K., Uma Maheshwari, V., & Aluvalu, R. (2022). Maintaining IoT Healthcare Records Using Cloud Storage. In *IoT and IoE Driven Smart Cities* (pp. 215-233). Springer, Cham.
- [3] Farman, H., Khan, T., Khan, Z., Habib, S., Islam, M. and Ammar, A., 2022. Real-time face mask detection to ensure COVID-19 precautionary measures in the developing countries. *Applied Sciences*, 12(8), p.3879.
- [4] Habib, S., Alsanea, M., Aloraini, M., Al-Rawashdeh, H.S., Islam, M. and Khan, S., 2022. An Efficient and Effective Deep Learning-Based Model for Real-Time Face Mask Detection. *Sensors*, 22(7), p.2602.
- [5] Sadhana, S., Pandiarajan, S., Sivaraman, E., & Daniel, D. (2021). AI-based Power Screening Solution for SARS-CoV2 Infection: A Sociodemographic Survey and COVID-19 Cough Detector. *Procedia Computer Science*, 194, pp. 255-271, <https://doi.org/10.1016/j.procs.2021.10.081>.
- [6] Agarwal, C., Kaur, I. and Yadav, S., 2022, July. Hybrid CNN-SVM Model for Face Mask Detector to Protect from COVID-19. In *Artificial Intelligence on Medical Data: Proceedings of International Symposium, ISCMM 2021* (pp. 419-426). Singapore: Springer Nature Singapore.
- [7] S. Kasthuri, Dr. A. Nisha Jebaseeli, "An efficient Decision Tree Algorithm for analyzing the Twitter Sentiment Analysis", *Journal of Critical Reviews*, (Scopus Indexed), P.No. 1010 – 1018, Volume-7, Issue-4, July - 2020, ISSN: 2394-5125.
- [8] Konar, A. and Sunkaria, R.K., 2022, April. A Review on Face Mask Detection Techniques for COVID-19 prevention. In *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)* (pp. 2145-2150). IEEE.
- [9] Raviteja Kocherla, M. Chandra Sekhar & Ramesh Vatambeti (2022) Enhancing the energy efficiency for prolonging the network life time in multi-conditional multi-sensor based wireless sensor network, *Journal of Control and Decision*, DOI: 10.1080/23307706.2022.2057362
- [10] Rezoana, N., Hossain, M.S. and Andersson, K., 2022, February. Face mask detection in the era of covid-19: A cnn-based approach. In *Proceedings of the Third International Conference on Trends in Computational and Cognitive Engineering:*

TCCE 2021 (pp. 3-15). Singapore: Springer Nature Singapore.

[11] N. B. K, V. G. Biradar, A. K. S. Mallya, S. S. Sabat, M. S. G and P. K. Pareek, "Identification of Intracranial Hemorrhage using ResNeXt Model," 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), 2022, pp. 1-5, doi: 10.1109/MysuruCon55714.2022.9972396

[12] Jhaveri, Rutvij H., A. Revathi, Kadiyala Ramana, Roshani Raut, and Rajesh Kumar Dhanaraj. "A Review on Machine Learning Strategies for Real-World Engineering Applications." *Mobile Information Systems* 2022 (2022). <https://doi.org/10.1155/2022/1833507>

[13] Ai, M.A., Shanmugam, A., Muthusamy, S., Viswanathan, C., Panchal, H., Krishnamoorthy, M., Elminaam, D.S.A.

