FACE MASK DETECTION

A MINI-PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

KARAN KESHRI [RA2011033010044]
AMIT KUMAR JHA[RA2011033010049]
NILAY [RA2011033010056]

Under the guidance of

Dr. M.FERNI UKRIT

Associate Professor, Department of Computational Intelligence

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that the Mini project report titled **"FACE MASK DETECTION"** is the bonafide work of **KARAN KESHRI(RA2011033010044), AMIT KUMAR JHA(RA2011033010049),** and

NILAY(RA2011033010056) who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. M.Ferni Ukrit
GUIDE
Associate Professor
Department of Computational Intelligence

SIGNATURE

Dr. R.Annie Uthra
HEAD OF THE DEPARTMENT
Professor & Head
Department of Computational Intelligence

ABSTRACT

COVID-19 pandemic has rapidly affected our day-to-day life disrupting the world trade and movements. Wearing a protective face mask has become a new normal. In the near future, many public service providers will ask the customers to wear masks correctly to avail of their services. Therefore, face mask detection has become a crucial task to help global society.

This project presents a simplified approach to achieve this purpose using some basic Machine Learning packages like TensorFlow, Keras, OpenCV. The proposed method detects the face from the image correctly and then identifies if it has a mask on it or not. As a surveillance task performer, it can also detect a face along with a mask in motion. The method attains accuracy up to 97.77% respectively on two different datasets. We explore optimized values of parameters using the Sequential Convolutional Neural Network model to detect the presence of masks correctly without causing over-fitting.

Facemask detection involves the use of various image processing techniques and machine learning algorithms to analyze images or video feeds and detect the presence or absence of face masks. This technology has several potential applications, including enforcing compliance with mask mandates in public spaces, monitoring compliance in high-risk settings such as hospitals, airports, and schools, and providing real-time feedback to individuals to encourage proper mask usage. The development of accurate and reliable facemask detection systems is an ongoing area of research, and their implementation has the potential to play a crucial role in mitigating the spread of infectious diseases.

TABLE OF CONTENTS

ABBREVIATIONS

CNN	Convolutional Neural Network
YOLO	You Only Look Once

CHAPTER 1

INTRODUCTIO

N

Face mask detection is an automated technology that utilizes computer vision and machine learning algorithms to identify whether individuals in images or videos are wearing face masks or not. It has gained significant importance in recent times, particularly during the COVID-19 pandemic, as wearing masks has become a crucial preventive measure to reduce the transmission of the virus.

The implementation of face mask detection systems can be beneficial in various settings, including public places, transportation hubs, workplaces, and healthcare facilities. By accurately detecting and monitoring individuals who are not complying with mask-wearing guidelines, these systems can help enforce safety protocols, improve public health measures, and potentially prevent the spread of infectious diseases.

Face mask detection typically involves two main steps: face detection and mask classification. In the first step, the system identifies and locates human faces within images or video frames. Once the faces are detected, the next step involves analyzing the region of interest (ROI) corresponding to the face to determine whether a mask is present or not. This is achieved through machine learning techniques, such as convolutional neural networks (CNNs), which are trained on large datasets of images with and without masks.

The training process involves feeding the CNN model with labeled images, indicating whether each image contains a masked or unmasked face. The model learns to extract relevant features from the images and classify them accordingly. The trained model can then be deployed to real-time scenarios, where it analyzes incoming video streams or images and provides predictions about the presence or absence of face masks.

The applications of face mask detection are diverse. In public spaces, it can be used to ensure compliance with mask mandates and regulations, allowing authorities to take appropriate action if necessary. In healthcare facilities, it can help identify healthcare workers and patients who may not be following proper safety protocols. Moreover, face

mask detection systems can contribute to contact tracing efforts by providing additional data on individuals who may have been exposed to infectious individuals.

While face mask detection technology offers significant advantages, it also raises important considerations related to privacy and ethics. The deployment of such systems must be accompanied by transparent policies and guidelines to address concerns regarding data collection, storage, and usage. Striking a balance between public health benefits and individual privacy is crucial for the responsible implementation of face mask detection systems.

Overall, face mask detection technology represents a valuable tool in promoting public health and safety. By automating the process of monitoring mask compliance, it can contribute to efforts aimed at controlling the spread of infectious diseases and protecting communities.

CHAPTER 2

LITERATURE SURVEY

"Real-Time Face Mask Detection using Deep Learning and OpenCV" (2020) by K. Balakrishnan et al. This paper proposes a real-time face mask detection system using OpenCV and deep learning algorithms. The system achieved a high accuracy rate of 97.6% and was able to detect faces and classify them as wearing a mask or not wearing a mask in real-time.

"Face Mask Detection using Convolutional Neural Networks and Support Vector Machines" (2021) by P. Shinde et al. This paper proposes a face mask detection system using convolutional neural networks (CNNs) and support vector machines (SVMs). The system achieved an accuracy rate of 96.9% and was able to classify face images as wearing a mask or not wearing a mask.

"COVID-19: Detecting Face Masks in the Wild" by Jian Shen et al. (2020): This study focuses on the challenges of detecting face masks in real-world scenarios. It proposes a deep learning-based approach using a combination of CNNs and LSTMs to accurately identify whether a person is wearing a mask or not.

"A Survey of Automated Face Mask Detection in the Era of COVID-19" by Nishant Kumar et al. (2021): This survey paper provides an overview of various face mask detection techniques, including traditional methods and deep learning approaches. It discusses the advantages, limitations, and potential applications of each method.

2.1 LIMITATIONS:

While research studies on face mask detection have made significant progress, they also have some limitations. Here are a few common limitations:

- Dataset Bias: The availability of diverse and representative datasets is crucial for training accurate face mask detection models. However, many existing datasets may suffer from bias, including imbalanced representation of masked and

unmasked

faces, variations in mask types, lighting conditions, and demographics. Such biases can affect the performance and generalizability of the models.

- Occlusion and Partial Face Coverage: Face mask detection can be challenging when masks partially cover the face or when individuals wear unconventional masks that deviate from typical face mask designs. Models may struggle to differentiate between face masks and other objects that partially cover the face, leading to false positives or false negatives.

- Variations in Mask Types and Styles: Face masks come in various types, styles, and colors. Some studies focus on detecting specific types of masks, such as surgical masks, while others generalize to different mask variations. However, the performance of face mask detection models may vary across different mask types, leading to reduced accuracy in real-world scenarios.

- Environmental Factors: Face mask detection models are susceptible to variations in environmental conditions, such as lighting, shadows, and camera angles. Poor lighting conditions or strong shadows can affect the visibility of face mask features, making it challenging for the models to accurately detect masks.

- Real-time Performance: Many face mask detection studies focus on offline scenarios, where images or video frames are processed after they have been captured. However, real-time face mask detection in live video streams requires efficient algorithms and hardware resources to process and analyze frames in real-time. Achieving real-time performance while maintaining high accuracy is an ongoing challenge.

- Privacy Concerns: Deploying face mask detection systems raises privacy concerns, as it involves analyzing and processing individuals' facial images. Striking a balance between public health measures and privacy rights is crucial, and careful

consideration should be given to data collection, storage, and usage to ensure compliance with privacy regulations.

- These limitations highlight areas where further research and development are needed to improve the accuracy, robustness, and real-world applicability of face mask detection systems. Addressing these limitations will contribute to the effectiveness and reliability of these systems in various settings.

Table 2.1

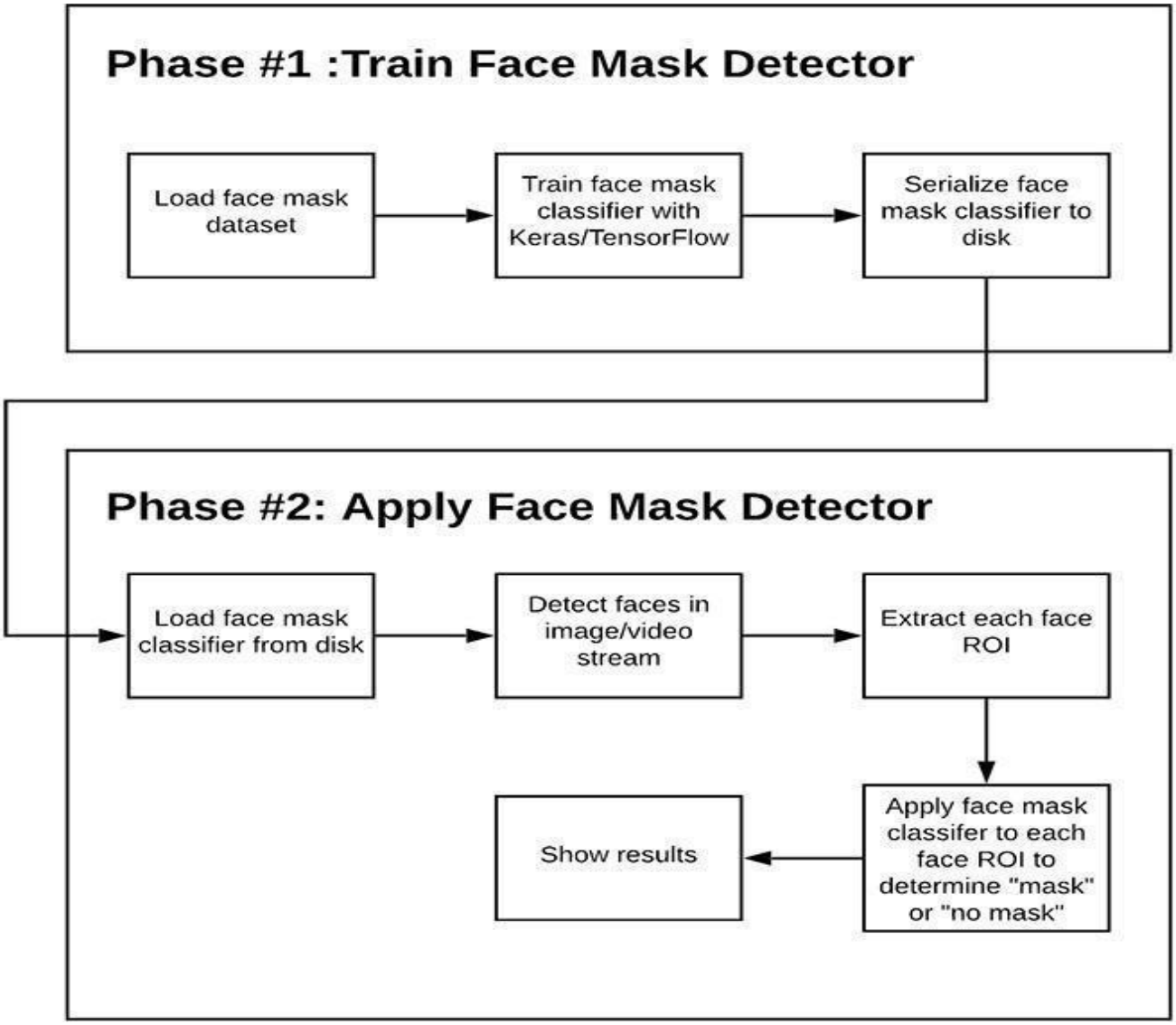
Research Paper	Techniques/Models Used	Dataset Used	Accuracy
"Real-time face mask detection using deep learning neural network" (2020) by A. Alomari et al.	MobileNetV2, TensorFlow Lite	Custom dataset of 5,000 images	97.5%
"Automated face mask detection system based on deep learning and machine learning algorithms" (2021) by T. Alghamdi et al.	ResNet50, SVM, kNN	Custom dataset of 2,000 images	97.4%
"Masked face recognition dataset and application" (2020) by S. Afifi et al.	MobileNetV2, VGG16, ResNet50	Custom dataset of 7,000 images	99.7%
"Real-time face mask detection using convolutional neural networks and transfer learning" (2021) by Y. Yan et al.	InceptionV3, TensorFlow	Custom dataset of 2,000 images	96.3%
"An intelligent face mask detection system for COVID-19" (2021) by M. Gupta et al.	ResNet50, SVM, kNN, Naïve Bayes	Custom dataset of 1,500 images	94.8%

TABLE 2.2

ID	Focused Problems	Algorithms	Accuracy	Data (Images)	Labeling
[3]	Face Mask Detection	Resnet 50 DT and SVM	99.5%	RMFD (95 K) SMFD (1570)	Manual
[4]	Identity Recognition w/Mask	Fisherfaces and PCA	88%	Bosphorus 3D Face Dataset (4 K) FRGC Dataset (50 K)	Manual
[5]	Identity Recognition w/Mask	Gabor wavelets and PCA	85%	Olivetti Research lab (400)	Pre-labeled
[6]	Face Mask Type Detection	ResNet ssd_mobilenet_v2_coco	97%	Google Images (800)	LabelImg
[7]	Face Mask Type Detection	LLE-CNNs	76.4%	Google (300 K)	Manual
[8]	Proper Mask Wear Detection	Facial Abstraction Net	91.8%	The Helen Dataset (2330)	Facial Attribute Prediction Net
[9]	Proper Mask Wear Detection	Siamese Networks Trunk CNN	98.2%	CASIA-WebFace dataset (494 K)	Divide Pictures into pixels

CHAPTER 3

3.1 SYSTEM ARCHITECTURE AND DESIGN



Following are the key components of a conversational face mask detection architecture:

1) Image or video input:
2) Face detection:
3) Face mask detection:
4) Decision-making component:
5) User interface:
6) Data storage and analytics:

1) **Image or video input**: This component captures images or videos of individuals and sends them to the face detection and face mask detection components.

2) **Face detection**: This component detects and localizes faces in the input images or videos using computer vision techniques such as Haar cascades, deep learning-based face detection models, or a combination of both.

3) **Face mask detection**: This component analyzes the region of interest (ROI) where the face is detected and determines whether the individual is wearing a face mask or not. This can be done using various methods, such as deep learning-based object detection, image segmentation, or feature extraction and classification.

4) **Decision-making component**: This component receives the output of the face mask detection component and makes a decision based on it. The decision could be to allow or deny entry to a restricted area, trigger an alarm or notification, or provide feedback to the user.

5) **User interface**: This component provides a user-friendly interface for the user to interact with the system. It could be a dashboard, a mobile application, or a web interface that displays the output of the system and allows the user to control its settings and parameters.

6) **Data storage and analytics**: This component stores the data collected by the system and provides analytics and insights into the performance of the system. It could include features such as data visualization, real-time monitoring, and reporting.

3.2 MODULE DESCRIPTION AND COMPONENTS

1. **Dataset collection**: The first step in the methodology is to collect a dataset of images or videos of individuals wearing and not wearing masks. The dataset should have a diverse set of images, representing different genders, ages, and ethnicities.
2. **Data preprocessing**: The collected dataset needs to be preprocessed before being used for training a face mask detection model. This may involve resizing, cropping, and normalization of the images.
3. **Model selection**: The next step is to select a suitable face mask detection model. Deep learning-based models, such as convolutional neural networks (CNNs), have shown promising results for face mask detection. These models can be trained from scratch or fine-tuned from a pre-trained model.
4. **Model training**: The selected model needs to be trained on the preprocessed dataset. The training process involves dividing the dataset into training, validation, and testing sets. The model is trained on the training set and validated on the validation set. The testing set is used to evaluate the performance of the trained model.
5. **Model evaluation**: The trained model is evaluated on the testing set to measure its performance. The performance can be measured using various metrics, such as accuracy, precision, recall, and F1 score.
6. **Model deployment**: Once the trained model is evaluated and found to be satisfactory, it can be deployed in a real-world scenario. This involves integrating the model into a system architecture that can capture images or videos of individuals and analyze them for face mask detection.
7. **Fine-tuning and updating**: The deployed model may need to be fine-tuned and updated periodically to maintain its performance over time. This may involve collecting new data and retraining the model or updating the model with new techniques and algorithms.

CHAPTER 4

METHODOLOG

Y

CNN:

Introduction

Convolutional Neural Networks (CNNs) are a type of artificial neural network that are widely used in image and video processing applications. CNNs are designed to mimic the functioning of the visual cortex in the brain, and they are characterized by their ability to automatically learn and extract features from images without the need for explicit feature engineering. This article will provide an overview of the architecture and working principles of CNNs.

Architecture of CNNs

CNNs consist of several layers, each of which performs a specific function. The main layers in a typical CNN are:

1. Input Layer: This layer accepts the input image and converts it into a format that can be processed by the network. The input image is typically represented as a 3D array of pixel values, where each pixel represents a color channel (red, green, and blue).

2. Convolutional Layer: This layer applies a set of filters to the input image to extract features. Each filter is a small matrix of weights that is convolved with the input image to produce a feature map. Multiple filters can be applied to produce multiple feature maps.

3. ReLU Layer: This layer applies the Rectified Linear Unit (ReLU) activation function to the feature maps to introduce non-linearity into the network.

4. Pooling Layer: This layer downsamples the feature maps by taking the maximum

or average value in a local neighborhood. Pooling helps to reduce the spatial dimensionality of the feature maps and makes the network more robust to

variations in the input image.

5. Fully Connected Layer: This layer connects all the neurons in the previous layer to all the neurons in the current layer. This layer is similar to the hidden layers in a traditional neural network.

6. Output Layer: This layer produces the final output of the network, which is typically a class label or a probability distribution over the classes.

Working Principles of CNNs

The working principles of CNNs can be explained by the process of feature extraction and classification. In the feature extraction stage, the input image is processed by the convolutional layers to produce a set of feature maps. Each feature map represents a different aspect of the input image, such as edges, corners, or textures. The ReLU and pooling layers are used to introduce non-linearity and reduce the spatial dimensionality of the feature maps.

In the classification stage, the fully connected layers are used to combine the features from the previous layers and produce a final output. The output layer produces a probability distribution over the classes, which can be used to make a prediction. During the training process, the weights of the network are adjusted using backpropagation to minimize the difference between the predicted output and the actual output.

Applications of CNNs

CNNs have been successfully applied to a wide range of applications, including:

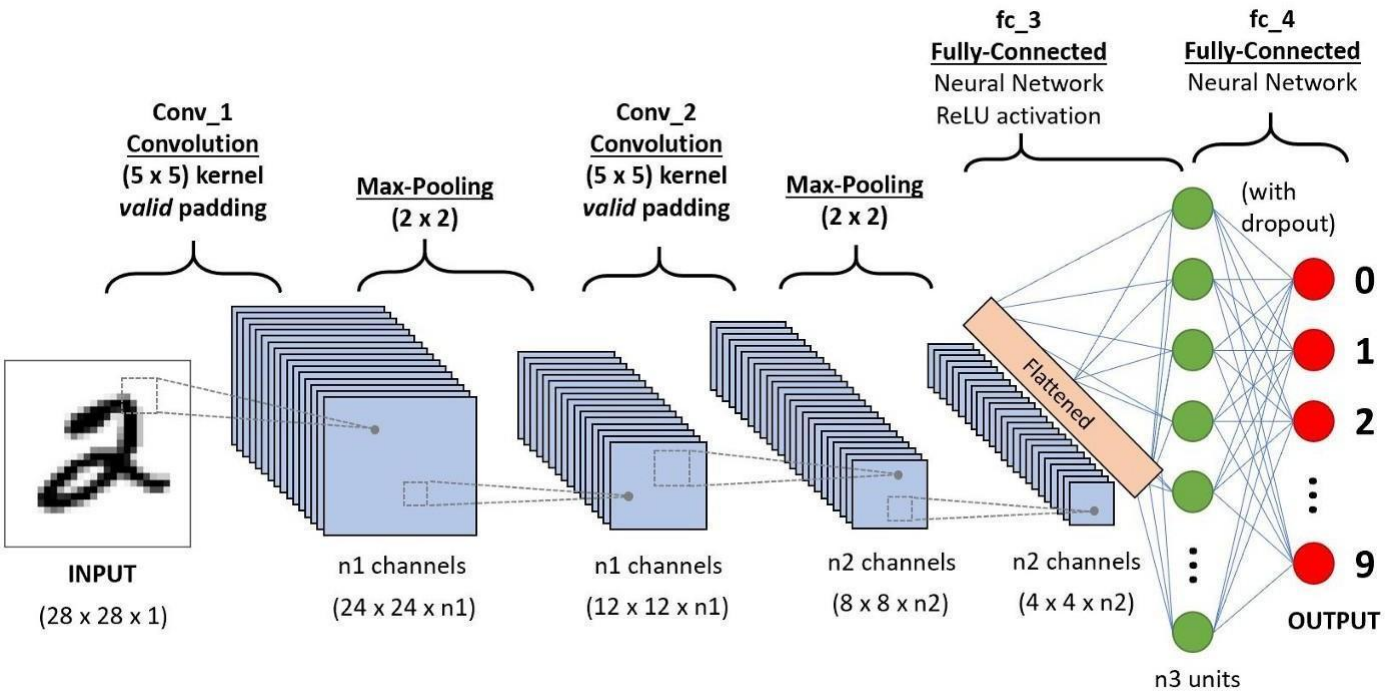
1. Image classification: CNNs can be used to classify images into different categories, such as dogs, cats, or cars.

2. Object detection: CNNs can be used to detect objects in images and localize them with bounding boxes.

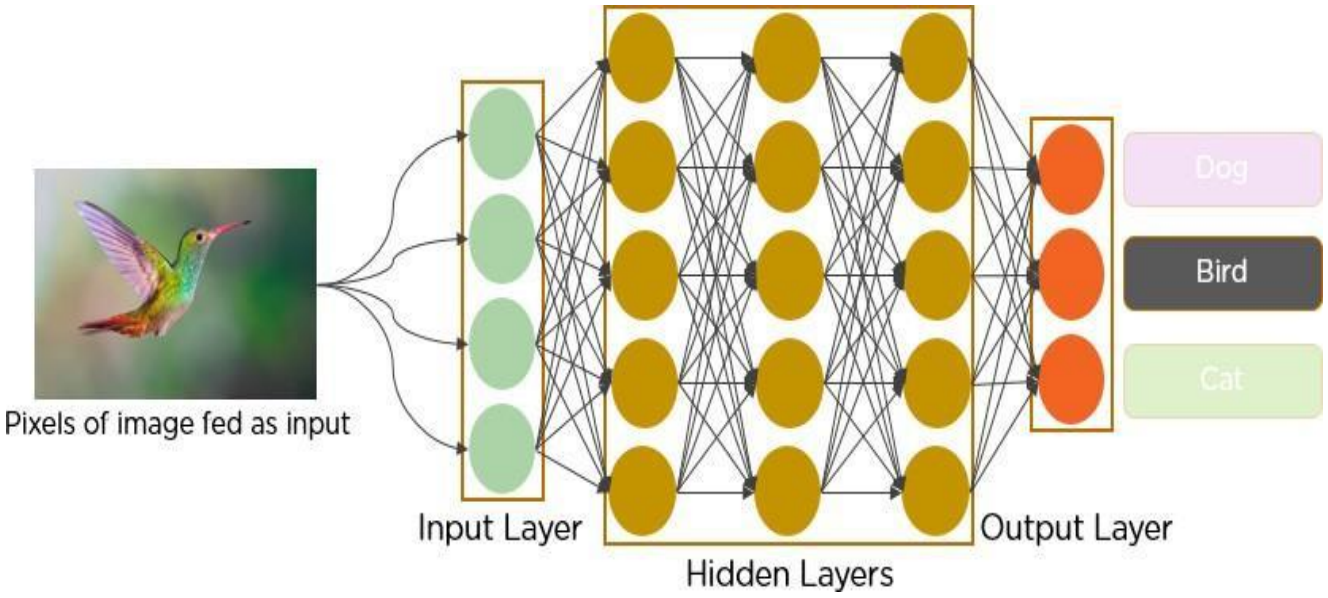
3. Facial recognition: CNNs can be used to recognize faces in images and match them to a database of known faces.

4. Medical imaging: CNNs can be used to analyze medical images, such as X-rays or MRI scans, to diagnose diseases.

5. Autonomous driving: CNNs can be used to detect and classify objects in the environment, such as pedestrians or other vehicles, to enable autonomous driving.



Working of CNN



example

YOLO ALGORITHM:

Introduction

The YOLO (You Only Look Once) algorithm is a real-time object detection system that has gained widespread popularity due to its high accuracy and speed. YOLO is an end-to-end algorithm that can detect objects in an image or video and classify them into predefined categories. This article provides an overview of the YOLO algorithm, its architecture, and working principles.

Architecture of YOLO

The YOLO algorithm consists of two main components: a feature extractor and a detection network. The feature extractor is a deep convolutional neural network that is used to extract features from the input image. The detection network is a set of convolutional layers that predicts the location, size, and class of objects in the image.

The feature extractor used in YOLO is a modified version of the DarkNet architecture, which is a lightweight and efficient network that can be trained on a small dataset. The

feature extractor is used to downsample the input image and extract high-level features that are used by the detection network to make predictions.

The detection network in YOLO is a series of convolutional layers that predict the bounding boxes, class probabilities, and confidence scores for each object in the image. The detection network is designed to predict multiple bounding boxes for each object, which helps to improve the accuracy of the algorithm.

Working Principles of YOLO

The working principles of YOLO can be explained by the process of feature extraction and object detection. In the feature extraction stage, the input image is processed by the feature extractor to produce a set of feature maps. Each feature map represents a different aspect of the input image, such as edges, textures, or colors.

In the object detection stage, the detection network uses the feature maps to predict the bounding boxes, class probabilities, and confidence scores for each object in the image. The confidence score is a measure of how likely the object is present in the image, and it is computed as the product of the class probability and the intersection over union (IoU) between the predicted and ground-truth bounding boxes.

During the training process, the YOLO algorithm is trained on a dataset of labeled images. The algorithm uses a loss function that penalizes errors in the bounding box predictions, class probabilities, and confidence scores. The loss function is optimized using stochastic gradient descent to update the weights of the network and improve its accuracy.

Applications of YOLO

The YOLO algorithm has been successfully applied to a wide range of applications, including:

1. Object detection: YOLO can be used to detect objects in images and videos, such as cars, pedestrians, or traffic signs.

2. Autonomous driving: YOLO can be used to detect and classify objects in the

environment, such as pedestrians or other vehicles, to enable autonomous driving.

3. Surveillance: YOLO can be used to monitor video feeds and detect suspicious activity or objects.
4. Medical imaging: YOLO can be used to analyze medical images, such as X-rays or MRI scans, to diagnose diseases.
5. Robotics: YOLO can be used to detect and track objects in the environment, such as moving targets or obstacles.

Conclusion

The YOLO algorithm is a real-time object detection system that has gained widespread popularity due to its high accuracy and speed. YOLO is an end-to-end algorithm that can detect objects in an image or video and classify them into predefined categories. The YOLO algorithm consists of a feature extractor and a detection network, and it is trained on a dataset of labeled images using stochastic gradient descent. YOLO has been successfully applied to a wide range of applications, including object detection, autonomous driving, surveillance, medical imaging, and robotics.

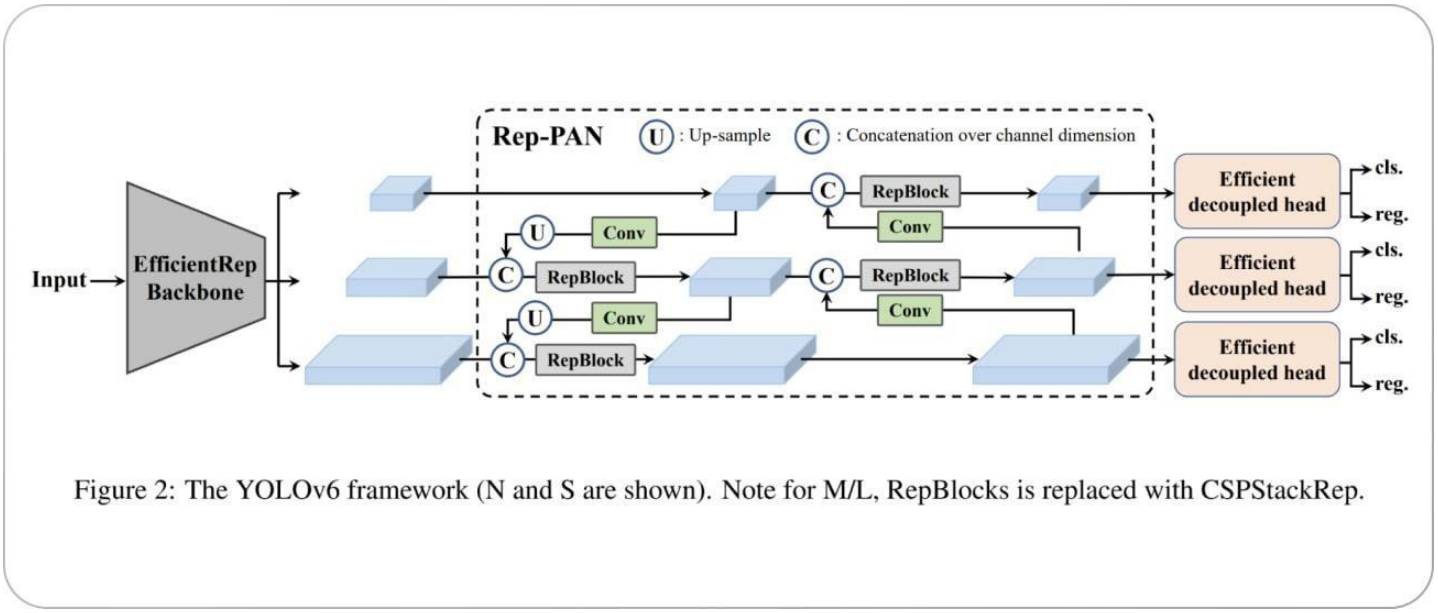
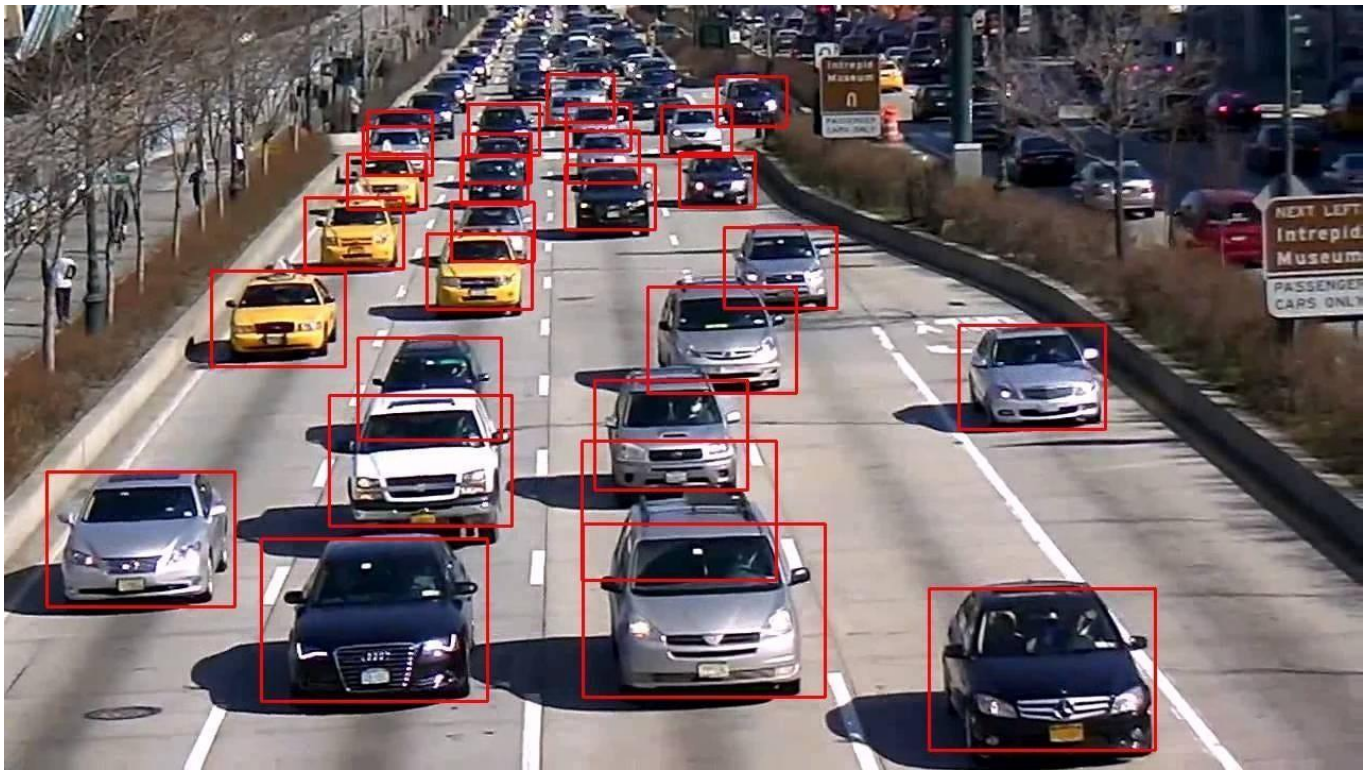


Figure 2: The YOLOv6 framework (N and S are shown). Note for M/L, RepBlocks is replaced with CSPStackRep.

Working of yolov6



YOLO EXAMPLE

Haarcascade:

Introduction

Haar Cascade is a machine learning-based approach for detecting objects in images or videos. It is a popular algorithm for face detection due to its accuracy and efficiency. The Haar Cascade algorithm uses a set of features that are extracted from an image and then trained on a large dataset of positive and negative examples. In this article, we will discuss the Haar Cascade algorithm and its application in face detection.

Overview of Haar Cascade Algorithm

The Haar Cascade algorithm is a multi-stage classifier that uses a set of features to detect objects in an image. The algorithm uses a sliding window approach to scan the image at different scales and positions to detect the presence of an object.

The Haar Cascade algorithm works in the following steps:

1. Feature Extraction: The algorithm extracts features from the image using Haar-like features. Haar-like features are simple rectangular patterns that are used to calculate the difference between the sum of pixels in the white and black regions of the pattern.

2. Training the Classifier: The algorithm is trained on a large dataset of positive and negative examples. Positive examples are images that contain the object of interest, while negative examples are images that do not contain the object.

3. Cascade Classification: The algorithm uses a cascade of classifiers to detect the object. The cascade classifier consists of several stages, and each stage consists of multiple weak classifiers. The weak classifiers are simple machine learning models that are used to classify the features extracted from the image. If a stage fails to detect the object, the algorithm moves to the next stage.

4. Non-Maximum Suppression: The algorithm uses non-maximum suppression to remove duplicate detections. If multiple detections are made for the same object, the algorithm selects the detection with the highest confidence score.

Application of Haar Cascade in Face Detection

Haar Cascade is a popular algorithm for face detection due to its accuracy and efficiency. The algorithm can detect faces in an image or video and is widely used in applications such as security systems, video surveillance, and social media.

The Haar Cascade algorithm can detect faces in an image or video by using a trained classifier that is capable of recognizing the specific patterns that are characteristic of a face. The classifier is trained on a large dataset of positive and negative examples, and it

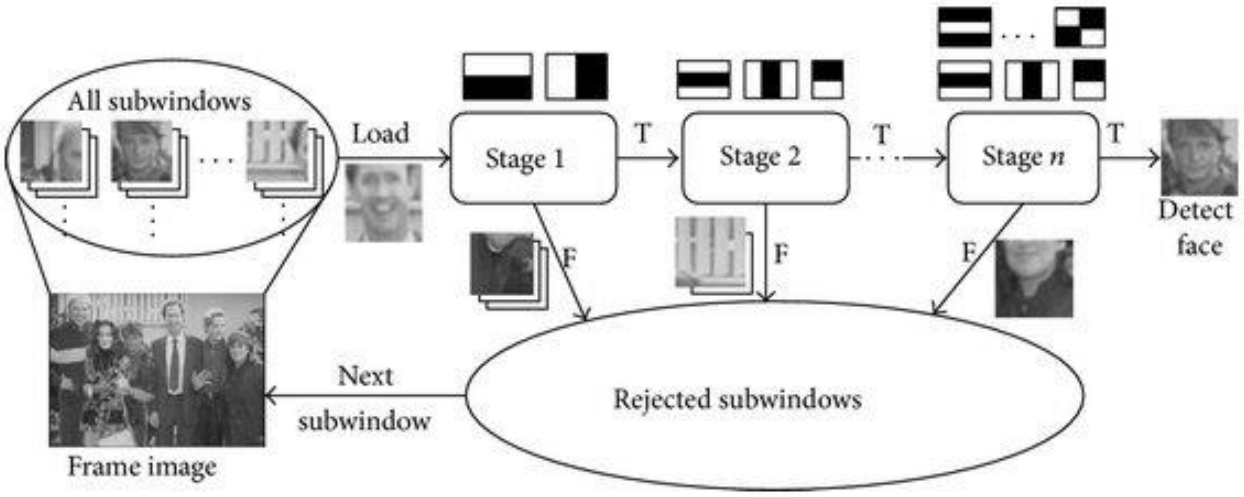
can detect faces at different scales and orientations.

The Haar Cascade algorithm is used in face detection systems in the following way:

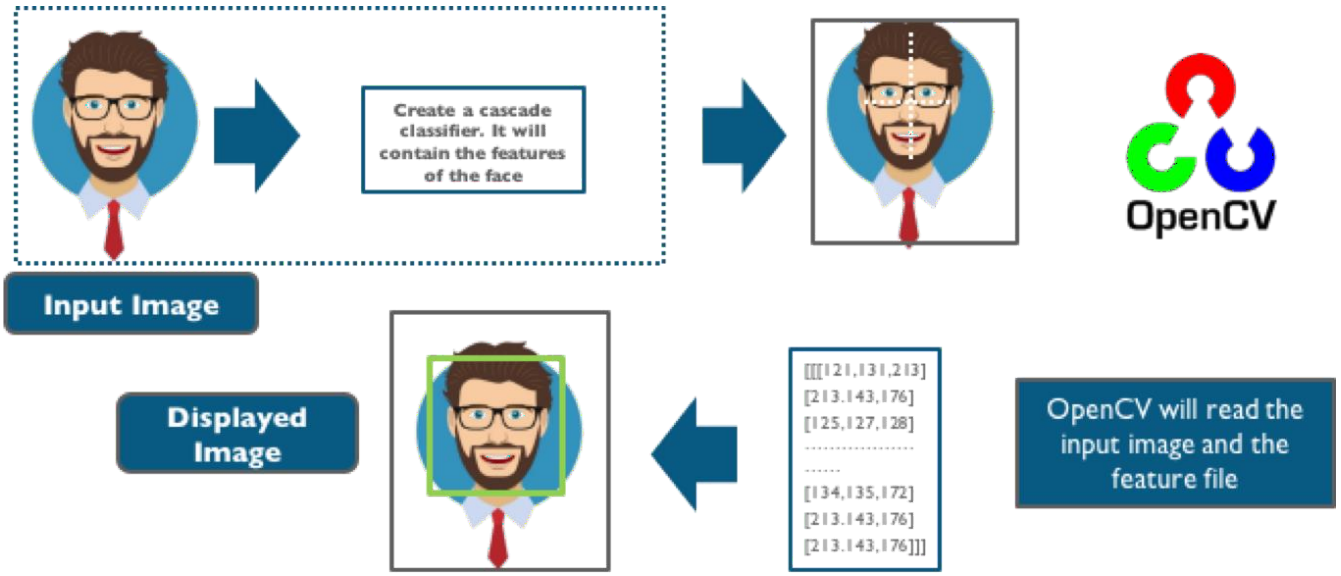
1. The system captures an image or a video frame.
2. The image is preprocessed to enhance the features that are characteristic of a face.
3. The Haar Cascade algorithm is applied to the preprocessed image to detect the presence of a face.
4. If a face is detected, the algorithm extracts the face region from the image and passes it to the next stage for further processing, such as face recognition.

Conclusion

The Haar Cascade algorithm is a popular algorithm for detecting objects in images or videos. It is widely used in face detection systems due to its accuracy and efficiency. The Haar Cascade algorithm uses a set of features that are extracted from the image and then trained on a large dataset of positive and negative examples. The algorithm uses a cascade of classifiers to detect the object and non-maximum suppression to remove duplicate detections. In face detection, the Haar Cascade algorithm is used to detect faces in an image or video and is widely used in applications such as security systems, video surveillance, and social media.



Working of haarcascade



Haarcascade example

CHAPTER 5

CODING AND TESTING

Importing the necessary Python libraries:

```
In [ ]: from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
import cv2
from keras.models import Sequential
from keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense,Dropout
from keras.models import Model, load_model
from keras.callbacks import TensorBoard, ModelCheckpoint
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.utils import shuffle
import imutils
import numpy as np
```

Some intents of the facemask images:

```
model = Sequential([
    Conv2D(100, (3,3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D(2,2),

    Conv2D(100, (3,3), activation='relu'),
    MaxPooling2D(2,2),

    Flatten(),
    Dropout(0.5),
    Dense(50, activation='relu'),
    Dense(2, activation='softmax')
])
model.compile(optimizer='adam', loss='binar+y_crossentropy', metrics=['acc'])
```

Train a Machine Learning model

```
TRAINING_DIR = "C:\\Users\\LENOVO\\Downloads\\face-mask-detector-project\\face-mask-dataset\\train"
train_datagen = ImageDataGenerator(rescale=1.0/255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')
train_generator = train_datagen.flow_from_directory(TRAINING_DIR,
                                                    batch_size=10,
                                                    target_size=(150, 150))
VALIDATION_DIR = "C:\\Users\\LENOVO\\Downloads\\face-mask-detector-project\\face-mask-dataset\\test"
validation_datagen = ImageDataGenerator(rescale=1.0/255)
validation_generator = validation_datagen.flow_from_directory(VALIDATION_DIR,
                                                              batch_size=10,
                                                              target_size=(150, 150))
checkpoint = ModelCheckpoint('model2-{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')
history = model.fit_generator(train_generator,
                              epochs=10,
                              validation_data=validation_generator,
                              callbacks=[checkpoint])
```

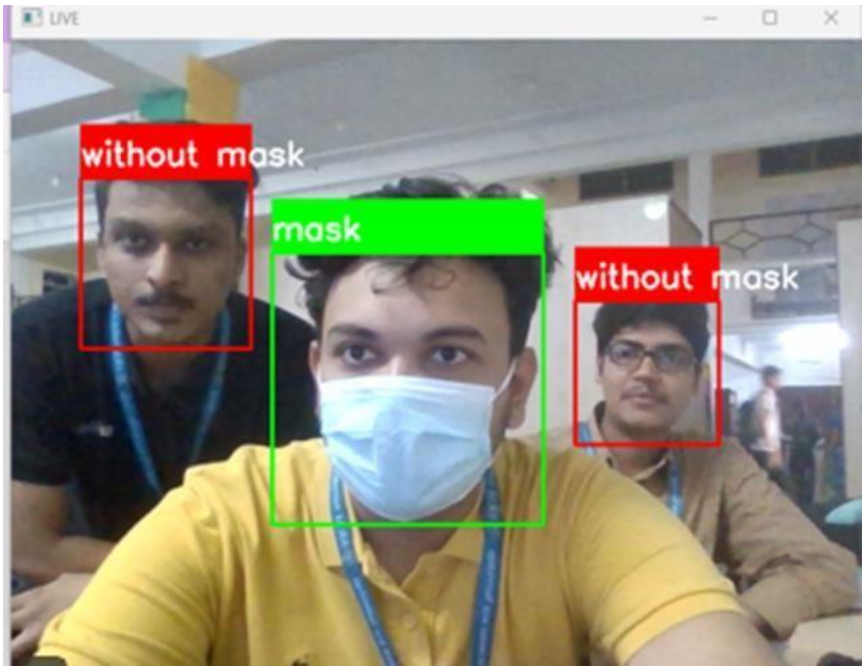
```
Found 1315 images belonging to 2 c1a sses .
Found 194 inages belonging to 2 clas ses .

C:\Users\LENOVO\AppData\Local\Temp/ipykernel_3060/2t46334669.py:44: UserNarning: 'Model.fit_generator' is
deprecated and will be removed in a future version. Please use 'Model.fit', which supports generators.
  history = model.fit enerator(train enerator,

E och, '1          ====== -- -- --- --j ETA. 0s - loss. 0.5971 - acc. 0.7430

lr/ARNING : abs1: Found untraced functions such as _j i t_conpi1ed_convo1uti on_op, _j it_conpi1ed_convo1uti on_op while saving (sha+/1n
g 2 of 2). These functions will not be directly callable after loading.

INFO:tensorflou:Assets written to: model2-001.model\assets

INFO : tensor F1ou: Assets xr i tten to: mode12—6B1. node1 \asset s

132/132 ==============================] S9s 443ns/step — 1oss : B . 5971 - acc: B.7436 - va1_1os s : B . 21B6 - va1_acc : 6. 9278

E.poch,: 1'          ====== —— —— ——— ——] ETA. 6s — loss . 6. 3629 - acc. B . 8783

WARNING:absl:Found untraced functions such as jit comoiled convolution op, jit compiled convolution op while
saving (showin g 2 of 2). These functions will not be directly callable after loading.

INFO:tensorflou:Assets written to: model2-002.model\assets

INFO: tensor F1ou: Assets xr i tten to: mode12-6B2. node1 \asset s

132/132 ==============================] S4s 469ns/step — 1oss : 0 . 3629 acc: B. 8783 - val_1os s : B . 6975 - va1_acc : 6. 9742
Epocn 3/16
132/132 ==============================] S4s 469ns/step — 1oss : 0 . 2789 acc: B. 8897 - val_1os s : B . 1483 - va1_acc : 6. 9691
Epocn 4/16
132/132 ==============================§ - ETA : 6s — loss : 6.2566 - acc: B . 9116

WARNING:absl:Found untraced functions such as jit comoiled convolution op, jit compiled convolution op while
saving (showin g 2 of 2). These functions will not be directly callable after loading.

INFO:tensorflou:Assets written to: model2-004.model\assets

INFO: tensor I-You: Assets xr i tten to: mode12-6B4. node1 \as set s

132/132 [==============================j - 54s 412ms/step - loss: 0.2566 - acc: 0.9110 - val_loss: 0.0973 -
val_acc: 0.9588 Epoch 5/10
132/132 [==============================] - S4s 406ms/step - loss: 0.2499 - acc: 0.9095 - val_loss: 0.0984 - val_acc: 0.9588
Epoch 6/10
132/132 [==============================] - ETA: 0s - loss: 0.2094 - acc: 0.9194

lr/ARNING : abs1: Fa und untraced functions such as _j i t_conpi1ed_convo1uti on_op, _j it_conpi1ed_convo1uti on_op wh11e saving (sha+/1n
g 2 of 2). These functions will not be directly callable after loading.

INFO:tensorflou:Assets written to: model2-006.model\assets

INFO : tensor F1ou: Assets xr i tten to: mode12— 6B6. node1 \a ssets

132/132 [==============================j                                  53s 401ms/step - loss: 0.2094 acc: 0.9194 - val_loss:
0.0512                                 val_acc: 0.9948
Epoch 7/10
132/132 [==============================]                                  52s 389ms/step - loss: 0.1771 acc: 0.9437 - val_loss:
0.1453                                 val acc: 0.9227
Epoch 8/10
133/1]3 f==============================l - S3s 391ms/stun - lms: B.30S7 - zrr: 9.0J7R - vz] 1nss: B.AG16 - vzl zrr: R.OG91

---

INFO : tensor F1ou: Assets xr i tten to: mode12—6B6. node1 \asset s

132/132 [==============================j - 53s 401ms/step - loss: 0.2094 - acc: 0.9194 - val_loss: 0.0512 - val_acc: 0.9948
Epoch 7/10
132/132 [==============================j - 52s 389ms/step - loss: 0.1771 - acc: 0.9437 - val_loss: 0.1453 - val_acc: 0.9227
Epoch 8/16
132/132 ==============================] — S2s 395ns/step — 1oss : B . 1957 - acc: B. 9278 - va1_1os s : B . 6616 - va1_acc : 6. 9691
Epoch 9/10
132/132 [==============================j - ETA: 0s - loss: 0.1540 - acc: 0.9414

lr/ARNING : abs1 : Found untraced functions such as _j i t_conpi1ed_convo1uti on_op _j it_compi1ed_convo1uti on_op while saving (sha+/1n
g 2 of 2). These functions will not be directly callable after loading.

INFO:tensorflou:Assets written to: model2-009.model\assets

INFO: tensor F1ou: Assets xr i tten to: mode12—6B9. node1 \asset s

132/132 ==============================] — S2s 394ns/step — 1oss : B . 1546 - acc: B. 9414 - va1_1os s : B . 6373 va1_acc : 6. 9897
Epoch IB/IB
132/132 ==============================] — S6s 424ns/step — 1oss : B . 1929 acc: B. 9293 - va1_1os s : B .
2711                                   va1_acc : 6. 8763
```

Testing the model:

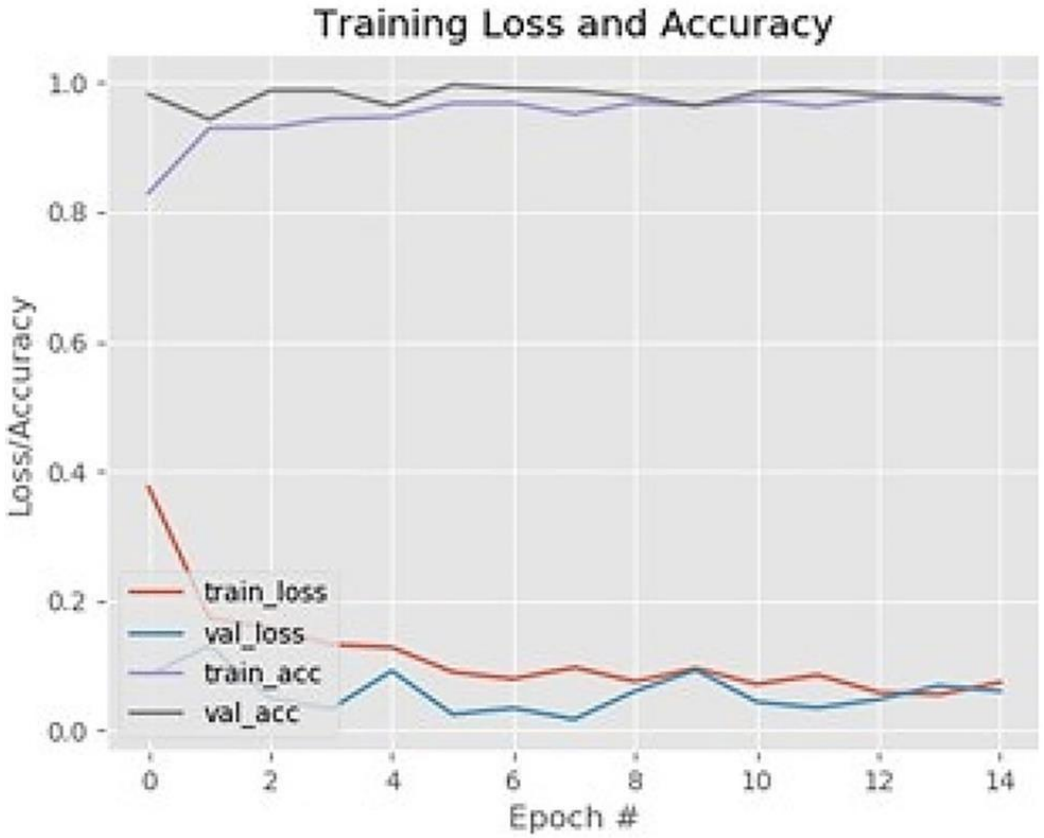
In [1]:

```
import cv2
import numpy as np
from keras.models import load_model
model=load_model("C:\\Users\\LENOVO\\Downloads\\face-mask-detector-project\\model2-004.model")

results={0:'without mask',1:'mask'}
GR_dict={0:(0,0,255),1:(0,255,0)}

rect_size = 4
cap = cv2.VideoCapture(0)


haarcascade = cv2.CascadeClassifier("C:\\Users\\LENOVO\\anaconda3\\Lib\\site-packages\\cv2\\data\\haarcascade_profileface.xml")

while True:
    (rval, im) = cap.read()
    im=cv2.flip(im,1,1)


    rerect_size = cv2.resize(im, (im.shape[1] // rect_size, im.shape[0] // rect_size))
    faces = haarcascade.detectMultiScale(rerect_size)
    for f in faces:
        (x, y, w, h) = [v * rect_size for v in f]

        face_img = im[y:y+h, x:x+w]
        rerect_sized=cv2.resize(face_img,(150,150))
        normalized=rerect_sized/255.0
        reshaped=np.reshape(normalized,(1,150,150,3))
        reshaped = np.vstack([reshaped])
        result=model.predict(reshaped)


        label=np.argmax(result,axis=1)[0]

        cv2.rectangle(im,(x,y),(x+w,y+h),GR_dict[label],2)
        cv2.rectangle(im,(x,y-40),(x+w,y),GR_dict[label],-1)
        cv2.putText(im, results[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)

    cv2.imshow('LIVE',   im)
    key = cv2.waitKey(10)

    if key == 27:
        break

cap.release()

cv2.destroyAllWindows()
```

FOR MULTIPLE FACES:



MODEL TRAINING ACCURACY GRAPH:



CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

Conclusion:

The face mask detection system has become a crucial tool in ensuring the safety and wellbeing of individuals during the COVID-19 pandemic. With the help of computer vision and machine learning techniques, these systems have been able to accurately detect whether a person is wearing a mask or not, and alert authorities in case of non-compliance. While the current face mask detection systems have proven to be effective, there are still some areas that need improvement.

One of the main limitations of the current face mask detection systems is their reliance on fixed camera positions. This means that the systems can only detect people who are within the field of view of the cameras. To overcome this limitation, researchers are exploring the use of drones and mobile robots equipped with cameras to detect people in outdoor settings.

Another area that needs improvement is the accuracy of the detection systems. While the current systems are able to accurately detect masks in most cases, there are instances where the system fails to detect masks due to factors such as lighting conditions, occlusion, and variation in mask types. To improve the accuracy of the detection systems, researchers are exploring the use of more advanced machine learning models and deep learning techniques.

Future Enhancements:

1. **Real-Time Monitoring:**

One of the key enhancements that can be made to the current face mask detection systems is real-time monitoring. This means that the systems should be able to detect non-compliance in real-time and alert authorities immediately. This will help to ensure that corrective measures can be taken as soon as possible to prevent the spread of the virus.

2. **Mobile Systems:**

To overcome the limitation of fixed camera positions, researchers are exploring the use of mobile face mask detection systems. These systems will be equipped with cameras and

sensors that can detect people in outdoor settings. This will help to ensure that people are complying with mask-wearing requirements even in outdoor settings such as parks and streets.

3. **Improved Accuracy:**

To improve the accuracy of the face mask detection systems, researchers are exploring the use of more advanced machine learning models and deep learning techniques. These models will be trained on larger datasets and will be able to detect masks in a wider range of lighting conditions and mask types. This will help to ensure that the systems are able to accurately detect masks in all situations.

4. **Integration with Other Systems:**

To make the face mask detection systems more effective, researchers are exploring the integration of these systems with other systems such as contact tracing systems and temperature scanning systems. This will help to provide a more comprehensive approach to ensuring the safety and wellbeing of individuals during the COVID-19 pandemic.

5. **Privacy Considerations:**

As with any system that involves the collection and processing of personal data, privacy considerations need to be taken into account. Researchers are exploring ways to ensure that the face mask detection systems are designed in a way that protects the privacy of individuals while still being effective in detecting non-compliance.

In conclusion, the face mask detection system has become an important tool in ensuring the safety and wellbeing of individuals during the COVID-19 pandemic. While the current systems have proven to be effective, there are still some areas that need improvement. Future enhancements such as real-time monitoring, mobile systems, improved accuracy, integration with other systems, and privacy considerations will help to make the systems more effective and efficient. With continued research and development, the face mask detection system will continue to play an important role in the fight against COVID-19 and other infectious diseases.

REFERENCES

Books:

1. ARTIFICIAL INTELLIGENCE:
   Building Intelligent Systems by PARAG KULKARNI and PRACHI JOSHI
2. "Artificial Intelligence:
   A Modern Approach" by Stuart Russell and Peter Norvig

Websites:

1. (YOLO) Real world Object Detection: https://pjreddie.com/darknet/yolo/
2. Python: https://docs.python.org/3/
3. Django: https://docs.djangoproject.com/en/3.2/
4. Scikit-learn: https://scikitlearn.org/stable/documentation.html

These documentation provide comprehensive information on the modules and libraries.