

FACE MASK DETECTOR

*Project Report submitted in partial fulfillment
for the award of the degree of*

BACHELOR OF TECHNOLOGY

Submitted by

Amjad Ali Khan

(18SCSE1010251)

Shankar Yadav

(18SCSE1010190)

IN

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

Under the Supervision of

Lalit Sharma

Associate Professor



(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

2021-2022

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled "FACE MASK DETECTION USING MACHINE LEARNING " in partial fulfillment of the requirements for the award of B.Tech submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of August, 2021 to December, 2021, under the supervision of Lalit Sharma Assistant Professor, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Amjad Ali Khan
18SCSE1010251
Shankar Yadav
18SCSE1010190

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Lalit Sharma
Assistant Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Amjad Ali Khan(18SCSE1010251) and Shankar Yadav(18SCSE1010190) has been held on ____and his/her work is

Recommended for the award of B.Tech in Computer Science And Engineering.

Signature of Examiner(s)

Signature of Supervisor(s)

Signature of Project Coordinator

Signature of Dean

(School of Computing Science and Engineering)

Date: November, 2021

Place: Greater Noida

Acknowledgement

It is feasible to stay up with the developing trends with the continuous revolution in Artificial Intelligence and Machine Learning, where breakthroughs happen in the blink of an eye. The human race as a whole is born with an attitude of excellence. It is the environment that determines whether or not the outcome of this mindset is visible. The report preparation provides a link between a student and current trends in order to enhance awareness. I would like to express my heartfelt gratitude to Mentor Mrs Lalit Sharma, Assistant Professor, Galgotias University, Greater Noida, for her insightful advice and constructive criticism.

The author wishes to express his heartfelt gratitude to our mentor, without whose permission, wise counsel, and capable supervision, we would not have been able to complete our research in this manner.

Finally, I want to express my gratitude to everyone who has helped me directly.

Abstract

A face mask has been declared a mandatory biosafety measure by the World Health Organization (WHO) in the wake of the COVID-19 pandemic outbreak. This has resulted in issues with current facial recognition systems, which has prompted the development of this investigation. An article in this journal describes the development of an algorithm that can recognize people even when they are wearing a mask from photographs taken by a camera. It is necessary to use a classification model based on the Tensorflow architecture and the OpenCv's face detector in order to achieve this. As a result, using these stages, it is possible to identify the location of the face and determine whether or not it is wearing a face mask. A set of observations consisting of 13,359 images is generated for the purpose of training the facial recognition models; 52.9 percent of the observations contain images with a face mask and 47.1 percent of the observations contain images without a face mask.

Introduction

Humanity has experienced some genuinely life-altering catastrophes in 2020, with the COVID-19 epidemic being the most significant. For the sake of public health and well-being, COVID-19 has issued a demand for stringent measures to be taken to prevent the spread of disease. Starting with basic cleanliness standards, people are taking every care to protect their personal and society's safety, moving on to hospital treatments. Personal protective equipment includes face masks. People wear face masks when they leave their homes. Authorities must strictly enforce the wearing of face masks by all participants in groups or in public locations. There should be a plan in place to make sure that everyone always follows this essential safety rule. A face mask detection system can be used to check this. To detect a person's mask, you must look at their face. There are two components to finding masks: finding faces and finding masks on those faces. Figuring out if there is a mask on someone's face is the first stage in figuring out whether or not there is a mask on that person's face. There are many uses for facial recognition, including biometrics, law enforcement, and security, but it can also be applied in other fields. Detector systems are being developed and applied all around the world. An improved detector is needed since the planet can't afford any more corona cases.

LITERATURE REVIEW

The review of the literature is divided into three major categories. The literature related to image classification using deep learning techniques is discussed in the first of the three categories described above. The concepts of the Internet of Things (IoT) are discussed in the second section of this section. The literature related to the combination of Internet of Things devices and deep learning techniques is discussed briefly in the third category.

With OpenCV, Keras/TensorFlow, and Deep Learning, we can detect whether someone is wearing a mask.

We'll go over our two-phase COVID-19 face mask detector, as well as the specifics of how our computer vision/deep learning pipeline will be put into action. Following that, we'll take a look at the dataset that will be used to train our custom face mask detector. After that, I'll show you how to create a Python script to train a face mask detector on our dataset using Keras and TensorFlow, which you can download [here](#). We'll use this Python script to train a face mask detector and then look at the results to see how well it worked. The trained COVID-19 face mask detector will be used in the implementation of two additional Python scripts, which will be used to perform the following tasks:

1. Recognize COVID-19 face masks in digital images.
2. Detecting the presence of face masks in real-time video streams

Finally, we'll take a look at the results of our face mask detector application to round out this post.

I'll also offer some additional suggestions for ways to make the document even better.

Two-phase COVID-19 face mask detector

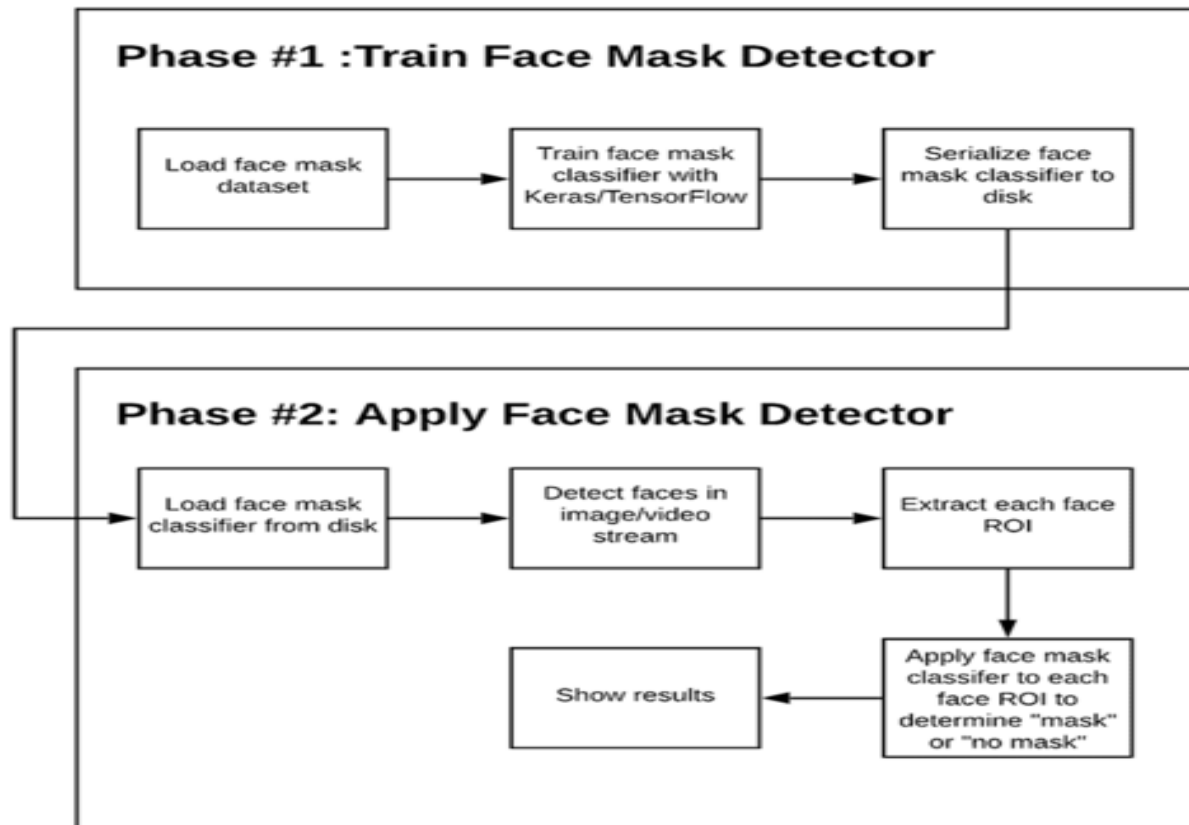


Figure 1: Phases and individual steps for building a COVID-19 face mask detector with computer vision and deep learning using Python, OpenCV, and TensorFlow/Keras.

In order to train a custom face mask detector, we need to break our project into two distinct phases, each with its own respective sub-steps (as shown by **Figure 1** above):

1. **Training:** Here we'll focus on loading our face mask detection dataset from disk, training a model (using Keras/TensorFlow) on this dataset, and then serializing the face mask detector to disk
2. **Deployment:** Once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifying each face as with mask or without mask

We'll go over each of these phases and their associated subsets in greater detail later in this tutorial, but for now, let's take a look at the dataset that we'll be using to train our COVID-19 face mask detector in the next section.

DATASET

- 1) Pre-training dataset: One shortage of deep learning is that it needs a large amount of training data to make the model converge and achieve satisfactory accuracy. There is no existing accurately labeled masked face dataset that meet our needs, and it is hard for us to collect thousands of training images and label them in a short period of time. According to the paper by Ross et al. [11], if the training sample of our own is insufficient, we can pre-train the model with some large dataset, then fine-tune the model with our own dataset. Comparing some existing face detection datasets, we choose the WIDER FACE dataset [26] as our pre-training dataset for its' large amount of training data and high degree of variability in scale, pose, occlusion, and illumination. Some of the images and annotations are shown in Fig. 5



Fig. 5. Part of the images and annotations in the WIDER FACE dataset [26]

- 2) MASKED FACE dataset: In order to train and evaluate our proposed masked face detector, we propose a MASKED FACE dataset. This dataset consists of 200 images collected from the internet, we carefully label the images and randomly select 160 images as the training set, and the rest 40 images as the testing set. Some of the images and annotations are shown in Fig. 6



Fig. 6. Part of the images and annotations in the MASKED FACE dataset.

B. training process

- 1) pre-train models: Since the three models in our proposed masked face detector are all binary classifiers, we only need to prepare two kinds of training samples, the positive and the negative. We randomly crop out rectangle regions whose Intersection over-Union (IoU) ratio are larger than 0.65 to a ground-truth annotation in the WIDER FACE dataset as the pre-training positive samples, and select the rectangle regions with IoU ratio less than 0.3 as negatives. We totally collect 286,346 positives and 859,038 negatives, the ratio of positive and negative is 1: 3. Then we resize all the training samples into 12×12 for training Mask-1, and 24×24 for training Mask-2 and Mask-3. 2) fine-tune models: a) fine-tune Mask-1: Similar to pre-training, we randomly crop out rectangle regions with IoU ratio larger than 0.65 to a ground-truth annotation in the MASKED FACE training set as the fine-tuning

positive samples, and select the rectangle regions with IoU ratio less than 0.3 as negatives. We totally collect 2,788 positives and 8,364 negatives, the ratio of positive and negative is still 1: 3. Then we resize all the training samples into 12×12 to pre-train Mask-1. b) Fine-tune Mask-2: For fine-tuning Mask-2, we use the Mask-1's positive training samples as the positives. Since the classification ability of Mask-2 should be higher than Mask1, the negative training samples of Mask-2 must be harder than the Mask-1's. For collecting Mask-2's negative training samples, we firstly use Mask-1 to densely scan the MASKED FACE training set to choose a threshold T 1 at 99% recall rate, then we use Mask-1 with threshold T 1 to evaluate the MASKED FACE training set. After the evaluation, we select the remaining detection windows with IoU ratio less than 0.3 to any ground-truth annotation as Mask-2's negative samples. Finally, we collect 5,576 negatives, we make the ratio of positive and negative to be 1:2. c) fine-tune Mask-3: Similar to fine-tuning Mask-2, we use the same positive training samples as Mask-1's, and we should make the Mask-3's negative samples harder than Mask2's. For collecting Mask-3's negative training samples, we use a two-stage cascade consist of Mask-1(threshold is T 1) and Mask-2 to densely scan the MASKED FACE training set to choose a threshold T 2 for Mask-2 at 97% recall rate. Again, we use the two-stage cascade to evaluate the MASKED FACE training set. After the evaluation, we select the remaining detection windows with IoU ratio less than 0.3 to any groundtruth annotation as Mask-3's negative samples. Still, we collect 5,576 negatives to make the ratio of positive and negative to be 1: 2.

C. Evaluation of our proposed method

We test our proposed masked face detection algorithm on the MASKED FACE testing set. We use the evaluation criterion in the PASCAL VOC [27] to evaluate the final detection windows. If the IoU ratio of a final detection window to one of the ground-truth annotations is larger than 0.5, we mark it as a correct one. Testing on the MASKED FACE testing set, we can achieve the best performance of 86.6% accuracy at 87.8% recall, some qualitative detection results are shown in Fig. 7.



Fig. 7. Some qualitative results of our masked face detection algorithm on the MASKED FACE testing set.

Deep learning is one of the major subfield of machine learning framework. Machine learning is the study of design of algorithms, inspired from the model of human brain. Deep learning is becoming more popular in data science fields like robotics, artificial intelligence (AI), audio & video recognition and image recognition. Artificial neural network is the core of deep learning methodologies. Deep learning is supported by various libraries such as Theano, TensorFlow, Caffe, Mxnet etc., Keras is one of the most powerful and easy to use python library, which is built on top of popular deep learning libraries like TensorFlow, Theano, etc., for creating deep learning model. Keras provides essential reflections and building units for the generation and transmission of ML arrangements at high iteration rates. It makes full use of TensorFlow's scalability and cross-platform features. Keras' primary data structures are layers and models [19]. Keras is utilized to implement all of the layers in the CNN model. It aids in the compilation of the overall model, as well as the conversion of the class vector to the binary class matrix in data processing.

Overview of Keras

Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is deep learning framework developed by Microsoft. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks.

Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.

Features

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features –

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

TensorFlow.

TensorFlow is an open source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy to understand framework. TensorFlow, an interface for expressing machine learning algorithms, is utilized for implementing ML systems into fabrication over a bunch of areas of computer science, including sentiment analysis, voice recognition, geographic information Extraction, computer vision, text summarization, information retrieval, computational drug discovery and flaw detection to pursue research [18]. In the proposed model, the whole Sequential CNN architecture (consists of several layers) uses TensorFlow at backend. It is also used to reshape the data (image) in the data processing.

Why is TensorFlow So Popular?

TensorFlow is well-documented and includes plenty of machine learning libraries. It offers a few important functionalities and methods for the same.

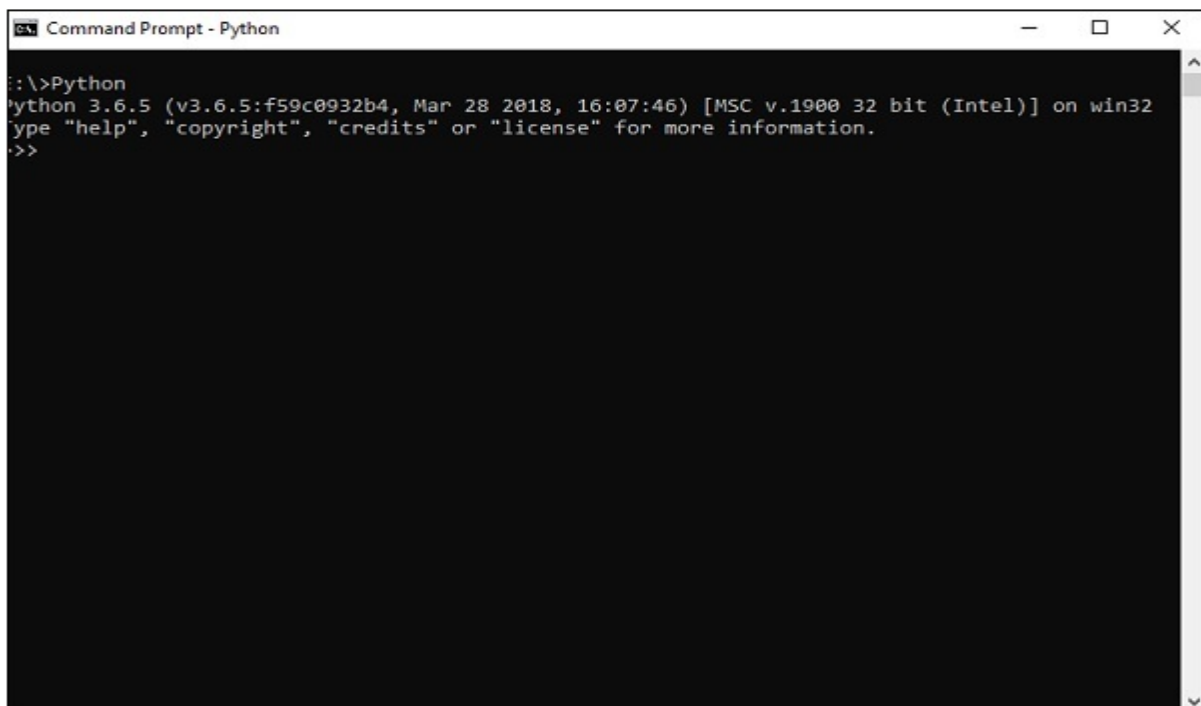
TensorFlow is also called a “Google” product. It includes a variety of machine learning and deep learning algorithms. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embedding and creation of various sequence models.

TensorFlow - Installation

To install TensorFlow, it is important to have “Python” installed in your system. Python version 3.4+ is considered the best to start with TensorFlow installation.

Consider the following steps to install TensorFlow in Windows operating system.

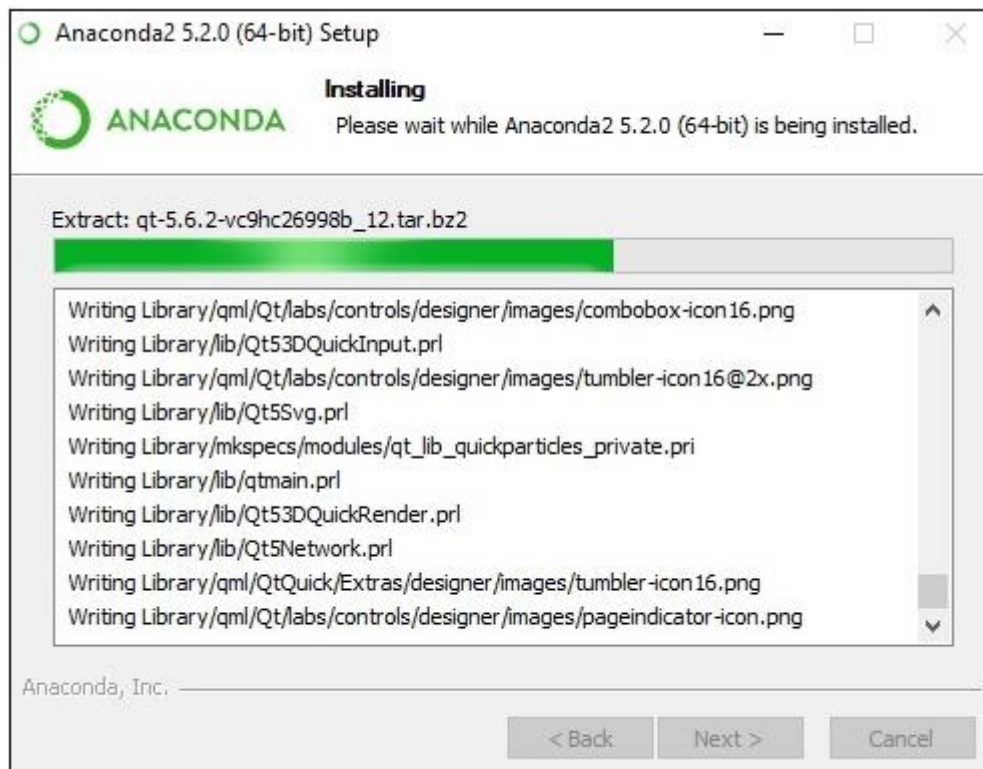
Step 1 – Verify the python version being installed



```
Command Prompt - Python
C:\>Python
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

Step 2 – A user can pick up any mechanism to install TensorFlow in the system. We recommend “pip” and “Anaconda”. Pip is a command used for executing and installing modules in Python.

Before we install TensorFlow, we need to install Anaconda framework in our system.



After successful installation, check in command prompt through “conda” command. The execution of command is displayed below –

```
C:\Users\Radhika>conda
usage: conda [-h] [-V] command ...

conda is a tool for managing and deploying applications, environments and packages.

Options:
positional arguments:
  command
  clean                Remove unused packages and caches.
  config               Modify configuration values in .condarc. This is modeled
                       after the git config command. Writes to the user .condarc
                       file (C:\Users\Radhika\condarc) by default.
  create               Create a new conda environment from a list of specified
                       packages.
  help                 Displays a list of available conda commands and their help
                       strings.
  info                 Display information about current conda install.
  install              Installs a list of packages into a specified conda
                       environment.
  list                 List linked packages in a conda environment.
  package              Low-level conda package utility. (EXPERIMENTAL)
  remove               Remove a list of packages from a specified conda environment.
  uninstall            Alias for conda remove. See conda remove --help.
  search               Search for packages and display associated information. The
                       input is a MatchSpec, a query language for conda packages.
                       See examples below.
```

Step 3 – Execute the following command to initialize the installation of TensorFlow –

```
pip install tensorflow-gpu
```

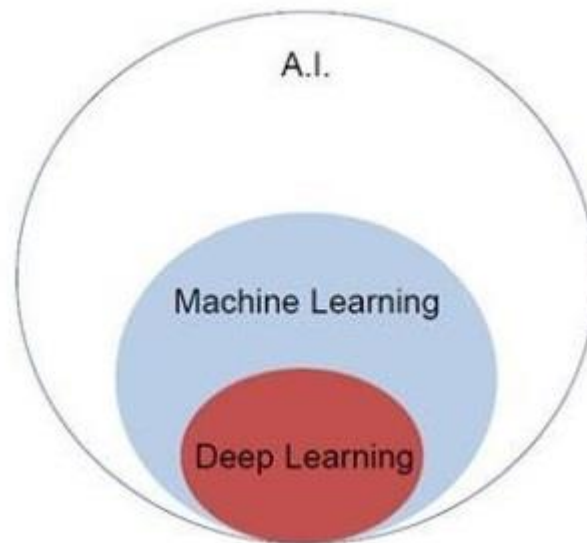
```
Command Prompt - pip install tensorflow
Requirement already satisfied: termcolor>=1.1.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: numpy>=1.13.3 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.14.5)
Requirement already satisfied: grpcio>=1.8.6 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: wheel>=0.26 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.31.1)
Requirement already satisfied: six>=1.10.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.11.0)
Requirement already satisfied: absl-py>=0.1.6 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.2.2)
Requirement already satisfied: astor>=0.6.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.6.2)
Requirement already satisfied: gast>=0.2.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: tensorboard<1.9.0,>=1.8.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.8.0)
Requirement already satisfied: setuptools in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (39.2.0)
Requirement already satisfied: html5lib==0.9999999 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.9999999)
Requirement already satisfied: bleach==1.5.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.5.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (2.6.11)
Requirement already satisfied: werkzeug>=0.11.10 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.14.1)
Installing collected packages: tensorflow
```

Understanding Artificial Intelligence

Artificial Intelligence includes the simulation process of human intelligence by machines and special computer systems. The examples of artificial intelligence include learning, reasoning and self-correction. Applications of AI include speech recognition, expert systems, and image recognition and machine vision.

Machine learning is the branch of artificial intelligence, which deals with systems and algorithms that can learn any new data and data patterns.

Let us focus on the Venn diagram mentioned below for understanding machine learning and deep learning concepts.



Machine learning includes a section of machine learning and deep learning is a part of machine learning. The ability of program which follows machine learning concepts is to improve its performance of observed data. The main motive of data transformation is to improve its knowledge in order to achieve better results in the future, provide output closer to the desired output for that particular system. Machine learning includes “pattern recognition” which includes the ability to recognize the patterns in data.

The patterns should be trained to show the output in desirable manner.

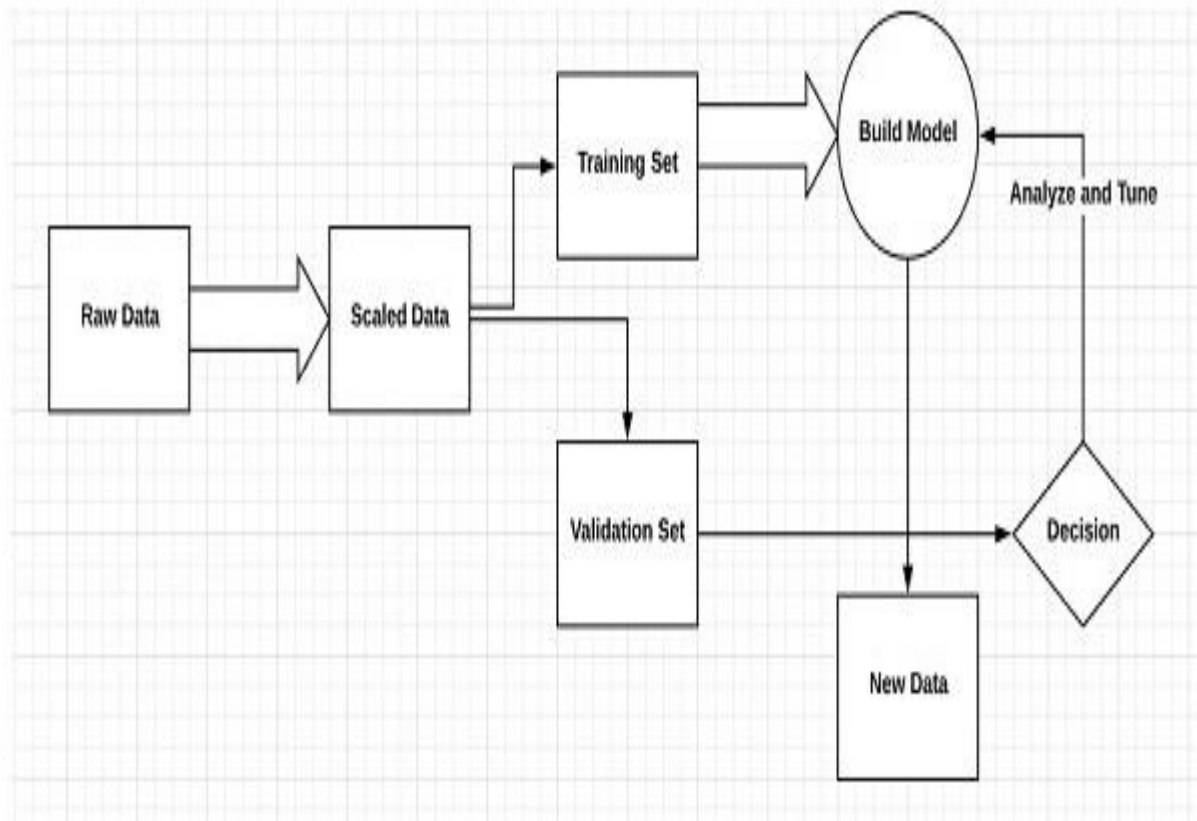
Machine learning can be trained in two different ways –

- Supervised training
- Unsupervised training

Supervised Learning

Supervised learning or supervised training includes a procedure where the training set is given as input to the system wherein, each example is labeled with a desired output value. The training in this type is performed using minimization of a particular loss function, which represents the output error with respect to the desired output system.

After completion of training, the accuracy of each model is measured with respect to disjoint examples from training set, also called the validation set.

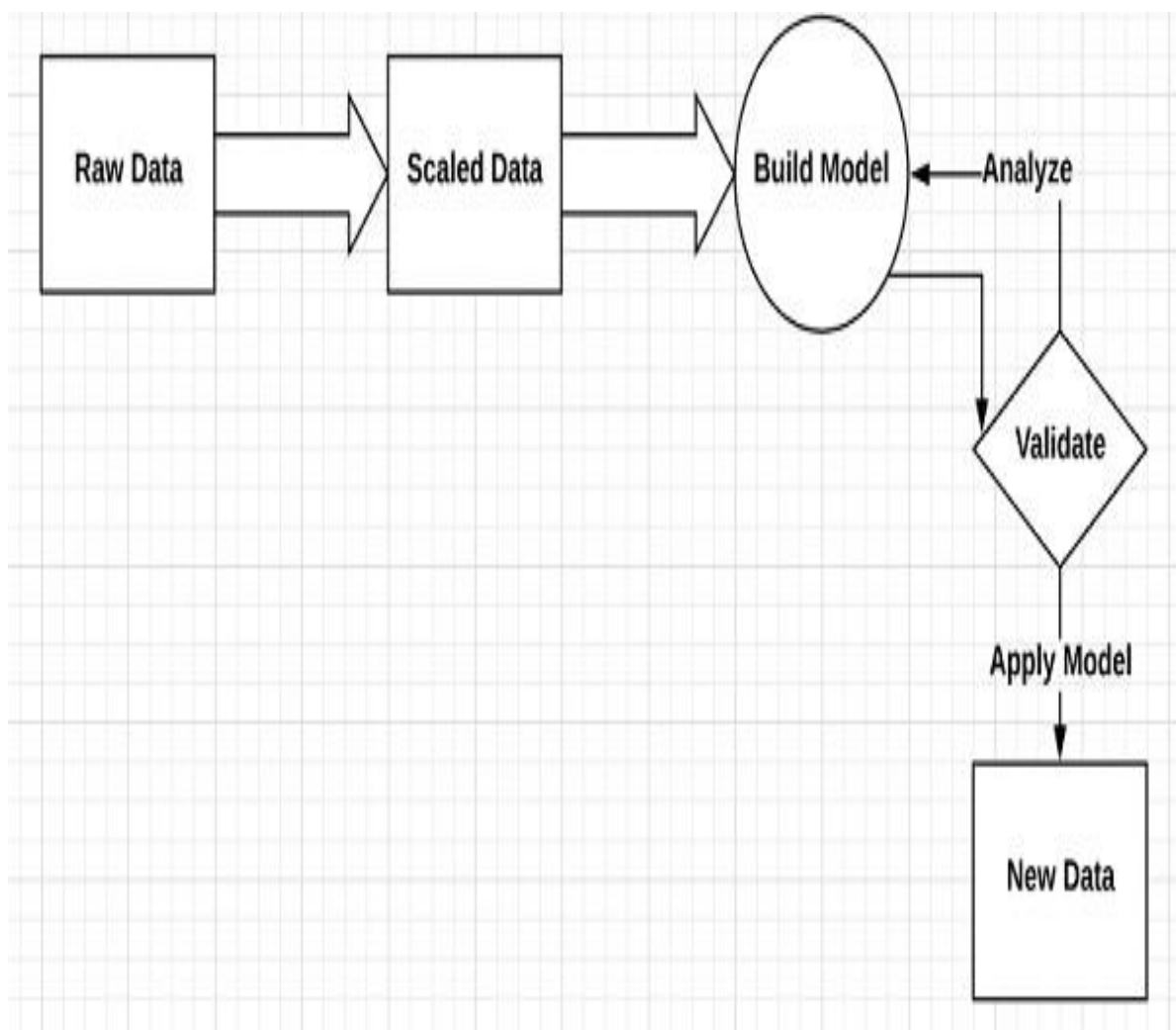


The best example to illustrate “Supervised learning” is with a bunch of photos given with information included in them. Here, the user can train a model to recognize new photos.

Unsupervised Learning

In unsupervised learning or unsupervised training, include training examples, which are not labelled by the system to which class they belong. The system looks for the data, which share common characteristics, and changes them based on internal knowledge features. This type of learning algorithms are basically used in clustering problems.

The best example to illustrate “Unsupervised learning” is with a bunch of photos with no information included and user trains model with classification and clustering. This type of training algorithm works with assumptions as no information is given.



OpenCV

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

Let's start the chapter by defining the term "Computer Vision".

OpenCV (Open Source Computer Vision Library) is a free and open-source computer vision and machine learning software library that is used to differentiate and recognise faces, recognise objects, group movements in recordings, trace progressive modules, follow eye gesture, track camera actions, expel red eyes from photographs taken with flash, find comparative photographs from an image database, perceive landscape and set up markers to overlay it with enhanced reality, and so on. The resizing and colour conversion of data images are accomplished through the use of these OpenCV features, which are described in detail below.

Computer Vision

Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modelling and replicating human vision using computer software and hardware.

Computer Vision overlaps significantly with the following fields –

- **Image Processing** – It focuses on image manipulation.
- **Pattern Recognition** – It explains various techniques to classify patterns.
- **Photogrammetry** – It is concerned with obtaining accurate measurements from images.

Computer Vision Vs Image Processing

Image processing:- deals with image-to-image transformation. The input and output of image processing are both images.

Computer vision :- is the construction of explicit, meaningful descriptions of physical objects from their image. The output of computer vision is a description or an interpretation of structures in 3D scene.

Applications of Computer Vision

Here we have listed down some of major domains where Computer Vision is heavily used.

Robotics Application

- Localization – Determine robot location automatically
- Navigation
- Obstacles avoidance
- Assembly (peg-in-hole, welding, painting)
- Manipulation (e.g. PUMA robot manipulator)
- Human Robot Interaction (HRI) – Intelligent robotics to interact with and serve people

Medicine Application

- Classification and detection (e.g. lesion or cells classification and tumor detection)
- 2D/3D segmentation
- 3D human organ reconstruction (MRI or ultrasound)
- Vision-guided robotics surgery

Industrial Automation Application

- Industrial inspection (defect detection)
- Assembly
- Barcode and package label reading
- Object sorting
- Document understanding (e.g. OCR)

Security Application

- Biometrics (iris, finger print, face recognition)
- Surveillance – Detecting certain suspicious activities or behaviours

Transportation Application

- Autonomous vehicle
- Safety, e.g., driver vigilance monitoring

Features of OpenCV Library

Using OpenCV library, you can –

- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

OpenCV was originally developed in C++. In addition to it, Python and Java bindings were provided. OpenCV runs on various Operating Systems such as windows, Linux, OSX, FreeBSD, Net BSD, Open BSD, etc.

This tutorial explains the concepts of OpenCV with examples using Java bindings.

OpenCV Library Modules

Following are the main library modules of the OpenCV library.

Core Functionality

This module covers the basic data structures such as Scalar, Point, Range, etc., that are used to build OpenCV applications. In addition to these, it also includes the multidimensional array **Mat**, which is used to store the images. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.core**.

Image Processing

This module covers various image processing operations such as image filtering, geometrical image transformations, color space conversion, histograms, etc. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.imgproc**.

Data Processing

a) **Data Visualization:** Data visualisation is the process of converting abstract data into meaningful representations through the use of encodings to facilitate knowledge exchange and insight finding. It is beneficial to investigate a certain trend in the dataset [7]. The total number of photos in the dataset is represented in both 'with mask' and 'without mask' categories. The category=os.listdir(data path) command categorises the list of directories in the specified data path. ['With mask', 'without mask'] is now the variable categories. Then, in order to get the number of labels, we must use labels=[i for I in range(categories)] to identify those categories. The labels are set to [0, 1]. Each category is now mapped to its corresponding label using the label dict=dict(zip(categories, labels)) function, which produces an iterator of tuples in the form of a zip object, with the elements in each passed iterator coupled together in the order they were supplied. 'with mask': 0,'without mask': 1'with mask': 1'with mask': 1'with mask': 1'with mask': 1'with mask': 1'with mask': 1'with

(b) **Grayscale picture conversion:** Modern descriptor-based image recognition systems frequently function on grayscale photos without expanding on the method used to convert from color-to-grayscale images. Using robust descriptors, the color-to-grayscale approach has little impact. The amount of training data needed to attain good performance could be increased by introducing unnecessary information. In order to save time and reduce processing requirements, grayscale is used to extract descriptors rather than working directly on colour images [8].

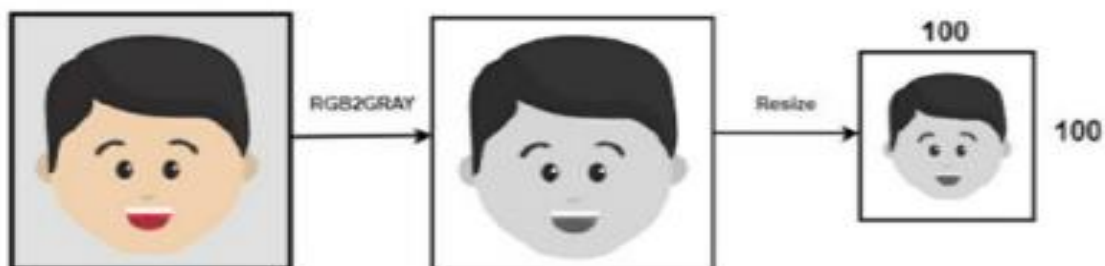


Fig. 3. Conversion of a RGB image to a Gray Scale image of 100 x 100 size

The function `cv2.cvtColor (input image, flag)` is used to alter the colour space. Depending on the value of the flag, the sort of conversion will be performed [9]. `Cv2.COLOR_BGR2GREY` is utilised for grey conversion in this instance. A fixed-size input image is required for deep CNNs. As a result, the dataset's photos must all have the same size. The grayscale image is enlarged to 100×100 pixels using `cv2.resize ()`.

c) **Image Reshaping:** This is the input for relegating an image, where each channel is represented by a unique pixel in three dimensions. The 3D feature tensor must have the same size in all of the photos. The analogous feature tensors for images and images, on the other hand, are not always coextensive. In most cases, CNNs will only accept images that have been finely adjusted. This leads to a number of issues during the course of data collecting and model deployment. However, if the input images are re-configured before they are augmented into the network, this constraint can be overcome. [11]. The pixel range between 0 and 1 is converged by normalising the photos. A four-dimensional array is then constructed using `data=np.reshape(Data, (Data. Shape[0],img size))` where 1 represents a grayscale image. A neural network's final output is a categorical representation of the data since the last layer has two outputs - one with a mask and the other without a mask.

MASKED FACE DETECTION APPROACH

The ideal way to identify whether person is approaching towards the camera or going away is to find out the distance between the person and the camera. As person approaches towards the camera, distance between person and camera will decrease and face detection can be triggered. To find out distance between person and camera, pinhole camera model is used. Pinhole camera model is simple camera model in which, light enters from the scene or distant objects, but only a single ray enters from any particular point. This particular point is then “projected” onto an imaging surface. Due to this, the 2016 Conference on Advances in Signal Processing (CASP) Cummins College of Engineering for Women, Pune. Jun 9-11, 2016 978-1-5090-0849-0/16/\$31.00 ©2016 IEEE 196 image on this image plane (or projective plane) is always in focus, and the size of the image relative to the distant object is given by camera’s focal length. For ideal pinhole camera, the distance from the pinhole aperture to the screen is precisely the focal length. This is shown in Figure 1, where f indicates the focal length of the camera, Z is the distance from the camera to the object, X indicates the length of the object, and x indicates the object’s image on the imaging plane

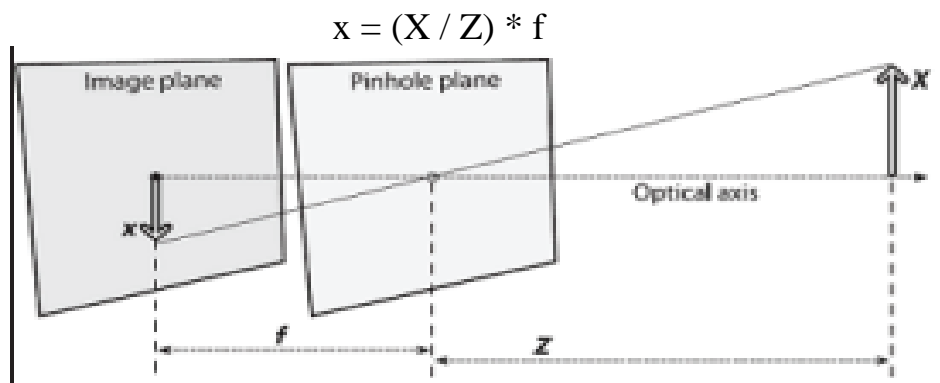


Figure 1: Pinhole camera model

Using Equation 2, Focal length in pixels = (Actual Distance * Distance from Vanishing Point) / Camera Height

Where, Actual Distance is the distance of the object from the camera. Distance from vanishing point can be calculated using Equation 3, Distance from Vanishing Point = End Y - Vanishing Point Y

Where, End Y is the end coordinate of the object (position at which the object is touching the ground) and Vanishing point Y is the Y coordinate of the vanishing point. Focal Length In Pixels, should be calculated for multiple observations and the average value from these observations should be used for configuration. Camera height is Height of the camera from the ground [10]. Using above Equation 3, distance of object from camera can be calculated and if it is decreasing with time, it indicates that person is approaching towards the camera and face detection can be activated. If face detection returns detection, it means there is no mask. If face detector does not detect anything, it implies that person is wearing a mask. B. Eye Line Detection In this step of masked face detection, eye line is detected in output window of person detection. Eyes and eyebrows are regions with low gray-level compared to other parts of face; their locations will correspond to the local valley of the horizontal projection histogram. So the eye line detection algorithm can be reduced as a valley finding procedure on the horizontal gray value projection histogram [11]. We extract top 30% part of detected window which is obtained as a result of person detection and calculate its horizontal projection histogram. The valley in horizontal projection Histogram corresponds to eye line. As shown in Figure 2, as valley is present at 220 location in histogram (pointed by red dot), 220th row in input image corresponds to eye line. As eye line is detected, we consider that person is approaching towards the camera and activate face detection. If eye line is detected and face is not detected, it indicates that it is mask. As shown in Figure. 3, for images with poor resolution, there are two valley points leading to conflict. As this creates an ambiguity, there are chances of error in true detection of eye line. C. Facial Part Detection Facial part detection based masked face detection is achieved in two parts. Face detection followed by facial part detection. Face detection as well as detection of facial parts like eyes, nose and mouth is implemented by Viola Jones's algorithm. Some of the advantages of Viola-Jones algorithm are, it is F

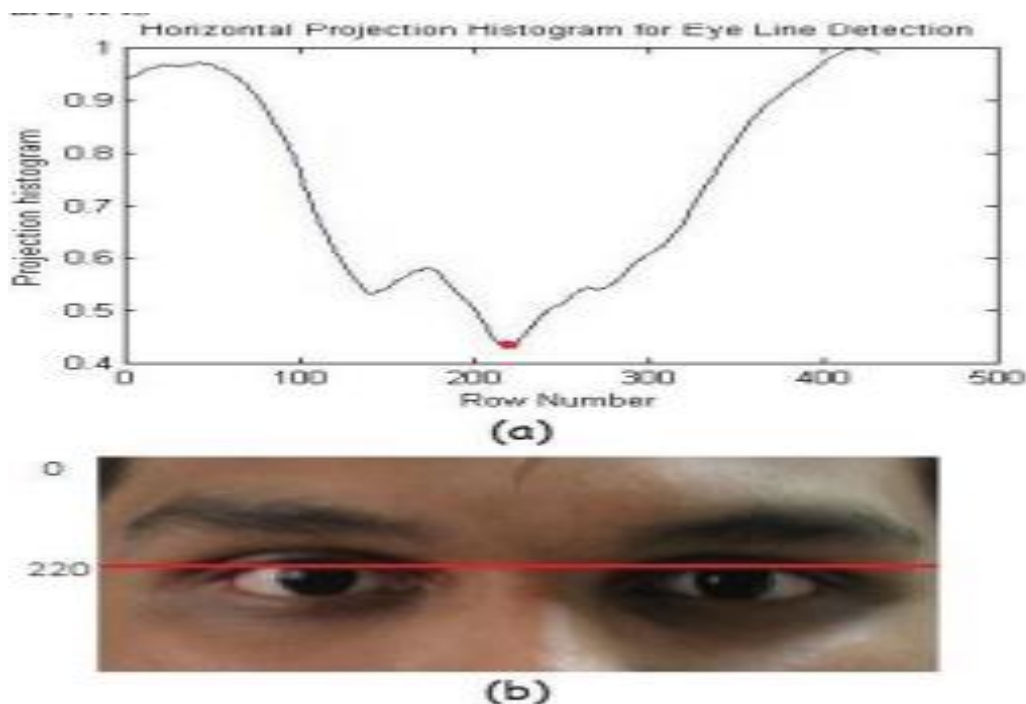
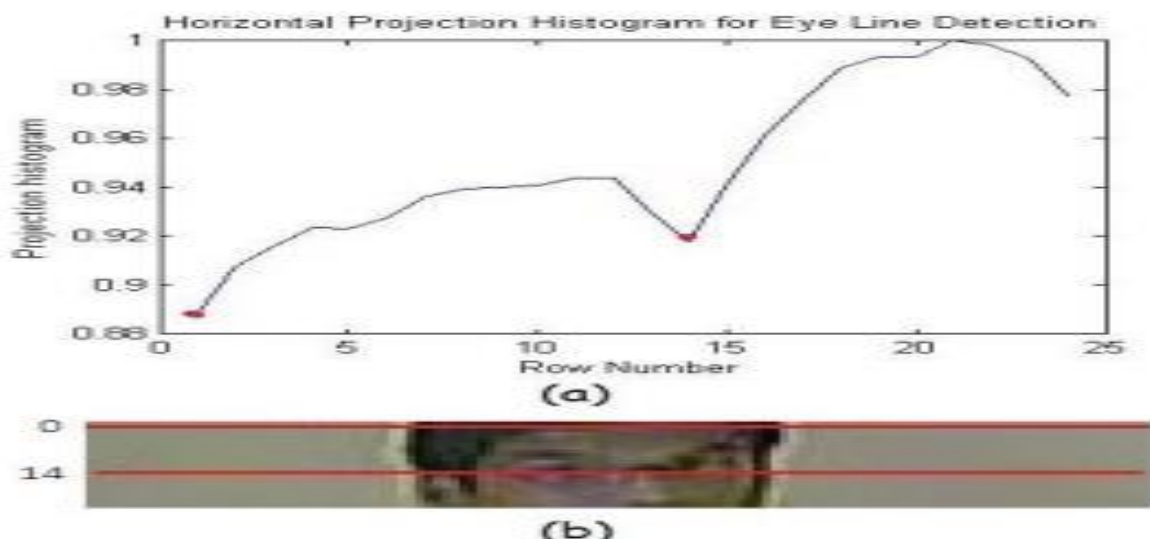


Figure 2: Horizontal projection histogram has valley at 220 as shown in (a) and 220th row in image corresponds to eyeline.

robust with very high true detection rate and very low false positive rate, it is real time and at least 2 frames must be processed per second. The algorithm has four stages, namely, Haar Feature Selection, Integral Image creation, Adaboost Training and Cascading Classifiers
Viola Jones face detection procedure classifies images based on the value of simple features. There are three features, namely two rectangle, three rectangle and four rectangle. The value of a two-

rectangle feature is computed by taking the difference between the sum of the pixels within two rectangular regions. These regions must be of same size and shape and are horizontally or vertically adjacent (see Figure. 4). Similarly three and four rectangle features can be computed. These rectangle features can be computed very rapidly using an intermediate representation for the image which is called as the integral image [13]. After formation of integral image, the AdaBoost learning algorithm is used to boost the classification performance of a simple learning algorithm. AdaBoost does this by combining a collection of weak classification functions to form a stronger classifier. This is followed by cascaded classifier.



Aforementioned Viola Jones algorithm based on simple features was implemented to detect face, eyes, nose and mouth in MATLAB. If face is detected, it is followed by detection of eyes, nose and mouth. If face is detected and eyes, nose and mouth are detected, it is not mask. But if face is detected and eyes and either of mouth or nose are detected, it indicates that it is mask. Eye Detection Facial part detection takes more time to execute (10 minutes to process 525 frames). So we propose eye detection step [9] [10], which takes comparatively less time (6 minutes to process 525 frames). In the first step, eye detection

Training of Model

a) Building the model using CNN architecture: CNN has risen to prominence in a variety of computer vision applications. Sequential CNN is currently being used in the current approach. MaxPooling and Rectified LinearUnit (ReLU) layers follow the first convolution layer. 200 filters are used to train the Convolution layer. The 2D convolution window has a kernel size of 3 x 3, which specifies its height and width. The model's first layer must be given information about the input's shape in order for it to do its job properly. Shape reckoning is possible in the following layers. As a result, the input shape is defined as data in this situation. [1:] returns the dimensions of the data array from the first occurrence. As long as the input volume is non-zero padded and the spatial dimensions are permitted to truncate, "default padding" is acceptable. The Conv2D class's activation parameter is set to "relu." Using gradient-descent methods, it can be easily optimised because it represents an almost linear function. Compared to other activation functions, it performs and generalises better in deeplearning. The output volume's spatial dimensions can be reduced via Max Pooling. Shape of output = (input shape - pool size + 1) / strides, where strides is the default value (1,1) [15] With 100 filters and a kernel size of 3 x 3, the second Convolution layer is built up as depicted in Figure 4.

Following this, we have the ReLu layer and then the MaxPooling layer. Data is fed into a fully connected neural network classifier by passing via a Flatten layer that turns a matrix of features into a vector. A Dropout layer with a 50% chance of setting inputs to zero is added to the model to reduce overfitting. Finally, 64 neurons with a ReLu activation function are inserted into the Dense layer of neurons. Uses Softmax activation for final layer (Dense) with two outputs for two categories

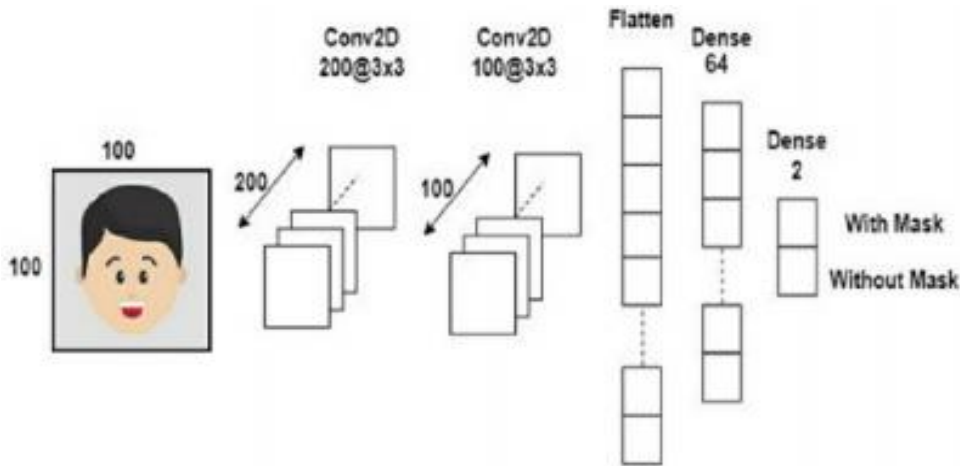


Fig. 4. Convolutional Neural Network architecture

Compilation is the first step in the process of learning, and it is essential. "adam" is the optimizer here. multiclass log loss, or categorical cross entropy, is employed as a loss function (the objective that the model tries to minimize). The measurements are set to "accuracy" because this is a classification challenge.

Splitting the data into subsets and training a CNN model: Models must be trained on a specific dataset and tested on a different dataset after creating the blueprint for data analysis. When making a prediction, accurate results may be achieved with the use of a good model and an optimum train split. The test size is set to 0.1, which means that 90% of the data in the dataset is used for training and 10% is used for testing. Model Checkpoint keeps tabs on the validation failure rate. The Sequential model is then used to fit the images in the training and test sets. In this case, 20% of the training data is used for validation. In order to maintain a trade-off between accuracy and the risk of overfitting, the model is trained for 20 epochs. A visual illustration of the suggested model is shown in Fig. 5.

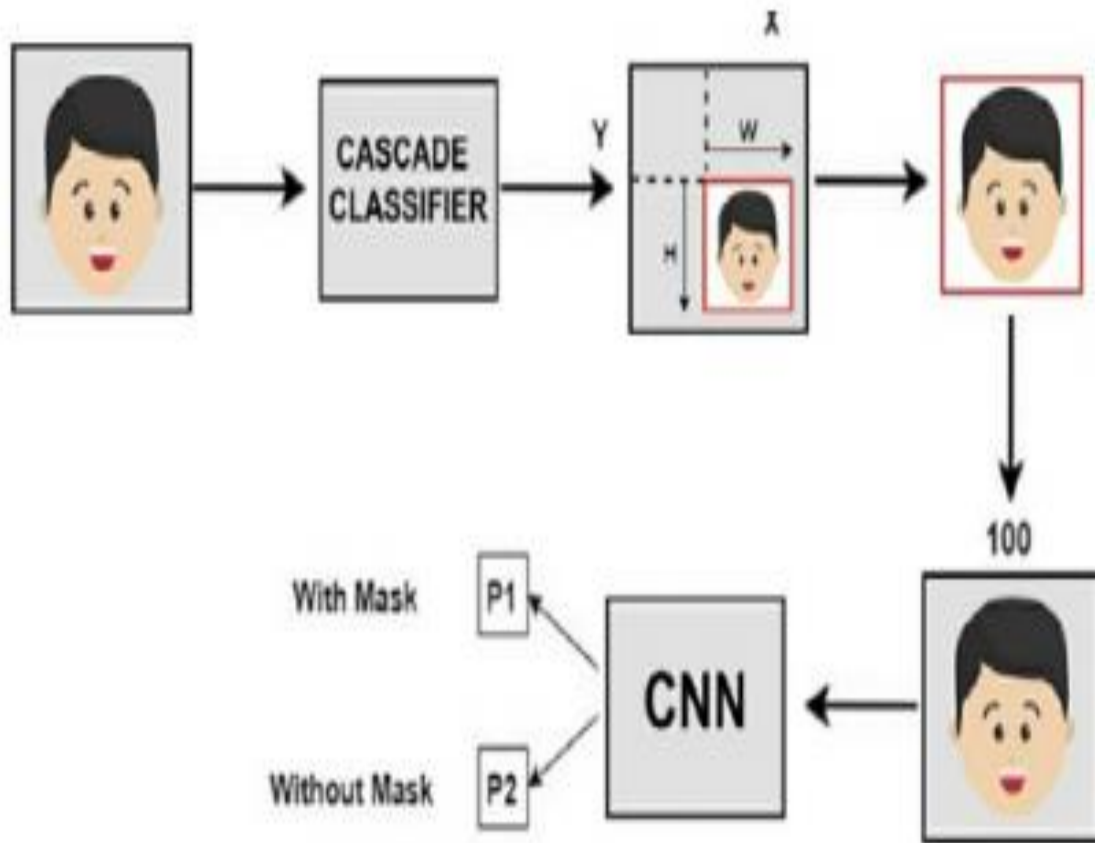


Fig. 5. Overview of the Model

RESULT AND ANALYSIS

Two datasets are used to train, validate, and test the model. An accuracy of 95.77 percent was achieved using dataset 1, as shown in fig.7. Figure 6 shows how this improved precision reduces the error's cost. For example, dataset 2 has many faces in the frame, as well as various masks with varied colours. As can be shown in Fig. 9, the model has an accuracy of 94.58 percent on dataset 2. The training and validation loss for dataset 2 are shown in comparison in Fig. 8. MaxPooling is largely responsible for this level of precision. It reduces the amount of parameters the model has to learn while still providing basic translation invariance. Using a sample-based discretization method, the input image is reduced in dimensionality and then down-sampled. Optimally, there are 64 neurons, which isn't excessive. A decrease in performance can be caused by an increase in the number of neurons and filters. Optimized filter parameters and pool size aid in removing the major portion (face) of the image in order to correctly detect mask presence without over-fitting.

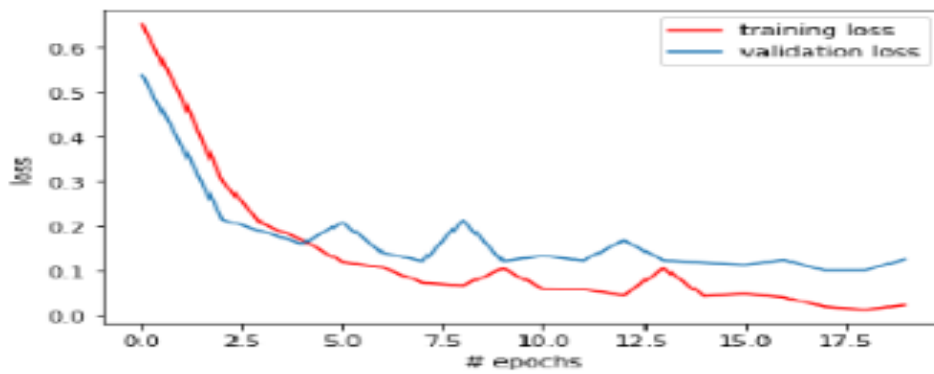


Fig. 6. # epochs vs loss corresponding to dataset 1

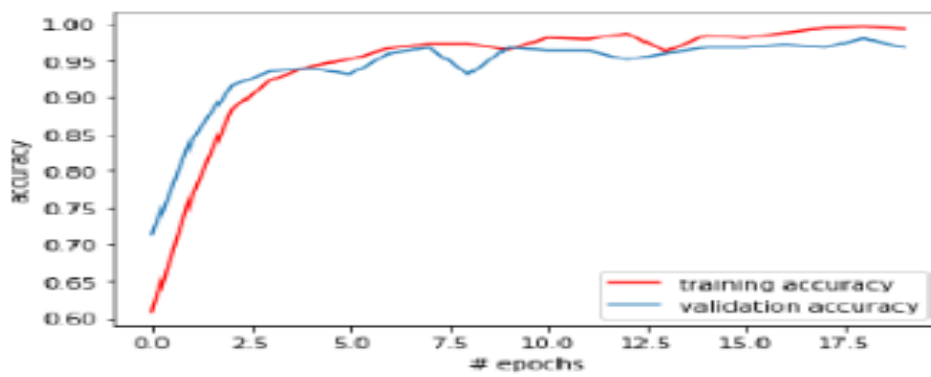


Fig. 7. # epochs vs accuracy corresponding to dataset 1

Faces that have been partially obscured by a mask, hair, or hand can be detected by the system. Annotated masks and hand-covered faces are differentiated based on the degree of occlusion in four key areas: the nose, mouth, chin, and eye. Therefore, a mask that completely covers the face, including the nose and chin, will only be referred to as "with mask" by the model.

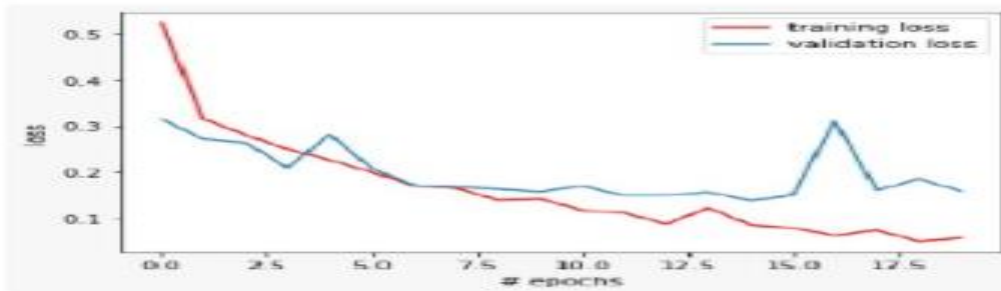


Fig. 8. # epochs vs loss corresponding to dataset 2

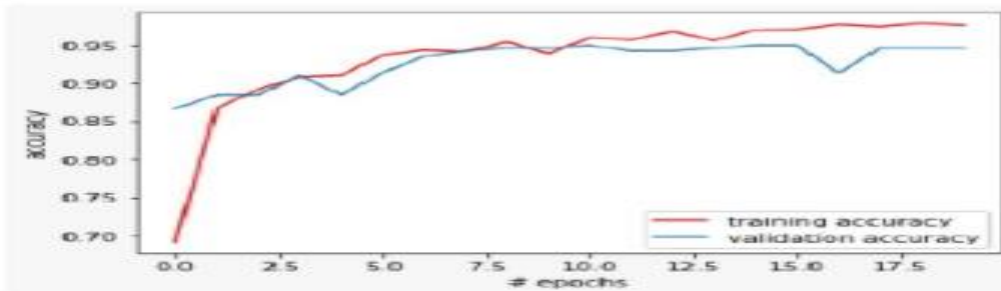


Fig. 9. # epochs vs accuracy corresponding to dataset 2

Fig. 8. # Epochs vs loss corresponding to dataset 2

Fig. 9. # Epochs vs accuracy corresponding to dataset 2

The main challenges faced by the method mainly comprise of varying angles and lack of clarity. Indistinct moving faces in the video stream make it more difficult. However, following the trajectories of several frames of the video helps to create a better decision – “with mask” or “without mask”.

CONCLUSIONS

In this study, we begin by laying out the rationale for our research. Finally, we depicted the model's learning and performance tasks. Accuracy has been attained with the use of basic ML tools and simplified approaches. A wide range of uses are possible. As a result of the Covid-19 problem, wearing a mask may become mandatory in the near future. Masks are required by several public service providers in order to use their services. The public health care system will greatly benefit from the model's implementation. Eventually, it will be possible to tell if someone is wearing the mask correctly if that feature is added. It is possible to update the model so that it can tell if the mask is virus-prone or not, i.e., if it is surgical, N95, or otherwise.

REFERENCES

- [1] W.H.O., “Coronavirus disease 2019 (covid-19): situation report, 205”. 2020
- [2] “Coronavirus Disease 2019 (COVID-19) – Symptoms”, Centers for Disease Control and Prevention, 2020. [Online]. Available: <https://www.cdc.gov/coronavirus/2019-ncov/symptoms-testing/symptoms.html>. 2020.
- [3] “Coronavirus — Human Coronavirus Types — CDC”, Cdc.gov, 2020. [Online]. Available: <https://www.cdc.gov/coronavirus/types.html>. 2020.
- [4] W.H.O., “Advice on the use of masks in the context of COVID-19: interim guidance”, 2020.
- [5] M. Jiang, X. Fan and H. Yan, “RetinaMask: A Face Mask detector”, arXiv.org, 2020. [Online]. Available: <https://arxiv.org/abs/2005.03950>. 2020.
- [6] B. Suvarnamukhi and M. Seshashayee, “Big Data Concepts and Techniques in Data Processing”, International Journal of Computer Sciences and Engineering, vol. 6, no. 10, pp. 712-714, 2018. Available: 10.26438/ijcse/v6i10.712714.
- [7] F. Hohman, M. Kahng, R. Pienta and D. H. Chau, “Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers,” in IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 8, pp. 2674-2693, 1 Aug. 2019, doi: 10.1109/TVCG.2018.2843369.
- [8] C. Kanan and G. Cottrell, “Color-to-Grayscale: Does the Method Matter in Image Recognition?”, PLoS ONE, vol. 7, no. 1, p. e29740, 2012. Available: 10.1371/journal.pone.0029740.
- [9] Opencv-python-tutroals.readthedocs.io. 2020. Changing Colorspaces — Opencv-Python Tutorials 1 Documentation. [online] Available at:https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html. 2020.
- [10] M. Hashemi, “Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation”, Journal of Big Data, vol. 6, no. 1, 2019. Available: 10.1186/s40537-019-0263-7 . 2020.
- [11] S. Ghosh, N. Das and M. Nasipuri, “Reshaping inputs for convolutional neural network: Some common and uncommon methods”, Pattern Recognition, vol. 93, pp. 79-94, 2019. Available: 10.1016/j.patcog.2019.04.009.
- [12] R. Yamashita, M. Nishio, R. Do and K. Togashi, “Convolutional neural Networks: an overview and application in radiology”, Insights into Imaging, vol. 9, no. 4, pp. 611-629, 2018. Available: 10.1007/s13244-018-0639-9.
- [13] “Guide to the Sequential model - Keras Documentation”, Faroit.com, 2020. [Online]. Available: <https://faroit.com/keras-docs/1.0.1/getting-started/sequential-model-guide/>. 2020.
- [14] Nwankpa, C., Ijomah, W., Gachagan, A. and Marshall, S., 2020. Activation Functions: Comparison Of Trends In Practice And Research for Deep Learning. [Online] arXiv.org. Available at: <https://arxiv.org/abs/1811.03378>. 2020.
- [15] K. Team, “Keras documentation: MaxPooling2D layer”, Keras.io, 2020. [Online]. Available: https://keras.io/api/layers/pooling_layers/

max pooling2d/. 2020.

[16] “prajnasb/observations”, GitHub, 2020. [Online]. Available: <https://github.com/prajnasb/observations/tree/master/experiements/data>. 2020.

[17] “Face Mask Detection”, Kaggle.com, 2020. [Online]. Available: <https://www.kaggle.com/andrewmvd/face-mask-detection>. 2020.

[18] “TensorFlow White Papers”, TensorFlow, 2020. [Online]. Available: <https://www.tensorflow.org/about/bib>. 2020.

[19] K. Team, “Keras documentation: About Keras”, Keras.io, 2020. [Online]. Available: <https://keras.io/about>. 2020.

[20] “OpenCV”, Opencv.org, 2020. [Online]. Available: <https://opencv.org/>. 2020.

[21] D. Meena and R. Sharan, “An approach to face detection and recognition,” 2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE), Jaipur, 2016, pp. 1-6, doi: 10.1109/ICRAIE.2016.7939462.

[22] S. Ge, J. Li, Q. Ye and Z. Luo, “Detecting Masked Faces in the Wild with LLE-CNNs,” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR

