**SIX MONTHS INDUSTRIAL TRAINING FIRST SYNOPSIS ON**

**‘PYTHON WITH MACHINE LEARNING’**

FOR THE PARTIAL FULFILLMENT OF THE

REQUIREMENT FOR THE AWARD OF THE DEGREE OF

**‘BACHELOR OF TECHNOLOGY**

**COMPUTER SCIENCE AND ENGINEERING SESSION 2017-2021’**



**GURU NANAK DEV UNIVERSITY, RC, GURDASPUR**

*PROJECT REPORT ON*

# Face mask Detection

**Submitted To: Submitted By:**

Ms. Mini Ahuja Palak Bhatia

2017CSA1750

B. Tech CSE (8th Sem)

|  |
| --- |
| **TABLE OF CONTENT** |
| • **INTRODUCTION TO COMPANY** |
| • **INTRODUCTION TO PYTHON** |
| • **PYTHON FEATURES** |
| • **INTRODUCTION TO THE PROJECT**  ➢ **Project Description** |
| • **PACKAGES USED**   * **cv2** * **argparse** * **numpy** * **imutils** * **tensorflow/keras** |
| • **HARDWARE AND SOFTWARE**  **REQUIREMENTS** |
| **1. Import the libraries and load the dataset** |
| **2.Preprocess the data** |
| 3. Create the model |
| 4. Train the model |
| 5. Evaluate the model |
| 6. . Running real time embedded system |
| 7. Screenshots |

# COMPANY PROFILE

**EXCELLENCE TECHNOLOGY ( ET)** is India based leading strategic IT Company offering integrated IT solutions with the vision to provide Excellence in software solution. We at

EXCELLENCE 4 TECHNOLOGY bring innovative ideas and cutting-edge technologies into business of customers. EXCELLENCE TECHNOLOGY is having rich experience in providing high technology end to end solutions in



**MOBILE APP AND WEB DEVELOPMENT.**

**PHILOSOPHY**

* To impart hardcore practical quality training among students/developers about latest technologies trending today.

* To share knowledge of information security and create awareness in the market. The solution to clients' as per the International standard practices and governance.

* To support good business practices through continual employee training and education

* To equip a local team with a strong knowledge of international best practices and international expert support to provide practical advisories in the best interests of our clients

**SERVICES AVAILABLE:**

* RISK Management Services
* Quality Control
* Business Process Re-Engineering
* Network Risk Analysis
* Software Testing
* Mobile Application Testing
* Wireless Penetration Testing
* Network Penetration Testing
* Application Security Testing

With the EXCELLENCE TECHNOLOGY experience the incredible services such as agile software development and the problems related to outsourcing. We comprise of the team of experienced and professional members who with their skills efficiently get the job done and innovatively help you to transform your ideas into the successful business.

**COMPANY’S CLIENTS**



**Programming Language Used:**

**INTRODUCTION TO PYTHON**

* **Overview**

Python is an interpreter, interactive, object-oriented high-level language. Its syntax resembles pseudo-code, especially because indentation is used to identify blocks. Python is a dynamically typed language and does not require variables to be declared before they are used. Variables “appear” when they are first used and “disappear” when they are no longer needed. Python is a scripting language like Tcl and Perl. Because of its interpreted nature, it is also often compared to Java. Unlike Java, Python does not require all instructions to reside inside classes. Python is also a multi-platform language, since the Python interpreter is available for a large number of standard operating systems, including MacOS, UNIX, and Microsoft Windows. Python interpreters are usually written in C, and thus can be ported to almost any platform which has a C compiler.

* **Evolution of technology**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

* **Python Features:**

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

## INTRODUCTION TO PROJECT

**Name of the Project: Face Mask Detection**

### Objective of the Project

The corona virus COVID-19 pandemic is causing a global health crisis so the effective protection methods are wearing a face mask in public areas according to the World Health Organization (WHO). The COVID-19 pandemic forced government’s across the world to impose lockdowns to prevent virus transmissions. Reports indicate that wearing facemasks while at work clearly reduces the risk of transmission. An efficient and economic approach of using AI to create a safe environment in a manufacturing setup. A hybrid model using deep and classical machine learning for face mask detection will be presented. A face mask detection dataset consists of with mask and without mask images , we are going to use OpenCV to do real-time face detection from a live stream via our webcam. We will use the dataset to build a COVID-19 face mask detector with computer vision using Python, OpenCV, and Tensor Flow and Keras. Our goal is to identify whether the person on image/video stream is wearing a face mask or not with the help of computer vision and deep learning.

To go about the python project, we’ll:

* + Detect faces
  + Classify into the person with mask and without mask
  + Check the accuracy of wearing mask
  + Put the results on the live video and display it

**The Dataset**

For this python project, we’ll use the Adience dataset; the dataset is available in the public domai. This dataset serves as a benchmark for face photos and is inclusive of various real-world imaging conditions like noise, lighting, pose, and appearance. The images have been collected from Flickr albums and distributed under the Creative Commons (CC) license. It has a total of 6,000 photos of 3,000 subjects in mask accuracy and is about 1GB in size. The models we will use have been trained on this dataset.

#### Description of the Project

To make machines more intelligent, the developers are diving into machine learning and deep learning techniques. A human learns to perform a task by practicing and repeating it again and again so that it memorizes how to perform the tasks. Then the neurons in his brain automatically trigger and they can quickly perform the task they have learned. Deep learning is also very similar to this. It uses different types of neural network architectures for different types of problems. **For example –** object recognition, image and sound classification, object detection, image segmentation, etc.

In this python project, we implemented a CNN to detect mask and accuracy of mask from a single picture of a face.

**Prerequisites**

You’ll need to install OpenCV (cv2) to be able to run this project. You can do this with pip-

pip install opencv-python

Other packages you’ll be needing are math and argparse, but those come as part of the standard Python library.

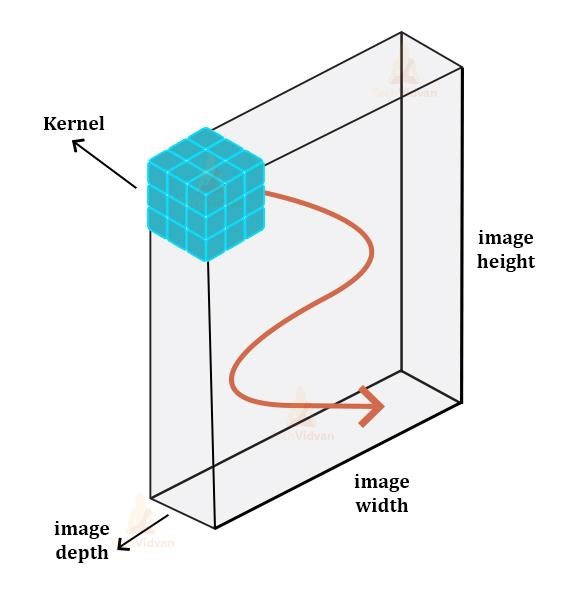
**What is OpenCV?**

OpenCV is short for Open Source Computer Vision. Intuitively by the name, it is an open-source Computer Vision and Machine Learning library. This library is capable of processing real-time image and video while also boasting analytical capabilities. It supports the Deep Learning frameworks TensorFlow, Caffe, and PyTorch.

**Convolution Neural Network:**

A Convolutional Neural Network or CNN is a Deep Learning Algorithm which is very effective in handling image classification tasks. It can capture the Temporal and Spatial dependencies in an image with the help of filters or kernels.

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The kernel is just like a small window sliding over the large window in order to extract the spatial features and in the end, we get feature maps.



The architecture of a ConvNet is analogous to that of the connectivity pattern of

Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlaps to cover the entire visual area.

#### Technology and Dataset Used

1. **Deep Learning:**

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabelled. Also known as deep neural learning or deep neural network.

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

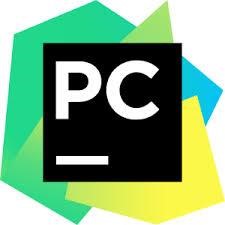
However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information.

#### 2. The Dataset:

#### For this python project, we’ll use the Adience dataset; the dataset is available in the public domain. This dataset serves as a benchmark for face photos and is inclusive of various real-world imaging conditions like noise, lighting, pose, and appearance. The images have been collected from Flickr albums and distributed under the Creative Commons (CC) license. It has a total of 26,580 photos of 2,284 subjects in eight age ranges (as mentioned above) and is about 1GB in size. The models we will use have been trained on this dataset.

#### Essential Libraries and Tools Used

1. **PyCharm: PyCharm** is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) used in [computer programming,](https://en.wikipedia.org/wiki/Computer_programming) specifically for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) language. It is developed by the [Czech](https://en.wikipedia.org/wiki/Czech_Republic) company [JetBrains.](https://en.wikipedia.org/wiki/JetBrains) It provides code analysis, a graphical debugger, an integrated unit tester, integration with [version](https://en.wikipedia.org/wiki/Revision_control)

[control systems](https://en.wikipedia.org/wiki/Revision_control) (VCSes), and supports web development with [Django](https://en.wikipedia.org/wiki/Django_(web_framework)) as well as [data science](https://en.wikipedia.org/wiki/Data_science) with [Anaconda.](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution))

PyCharm is [cross-platform,](https://en.wikipedia.org/wiki/Cross-platform) with [Windows,](https://en.wikipedia.org/wiki/Windows) [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux) versions. The Community Edition is released under the [Apache License,](https://en.wikipedia.org/wiki/Apache_License) and there is also Professional Edition with extra features – released under a [proprietary license.](https://en.wikipedia.org/wiki/Proprietary_software)

1. **A**[**rgparse**](https://docs.python.org/3/library/argparse.html#module-argparse)**:**

The [argparse](https://docs.python.org/3/library/argparse.html" \l "module-argparse" \o "argparse: Command-line option and argument parsing library.) module makes it easy to write user-friendly command-line interfaces. The program defines what arguments it requires, and [argparse](https://docs.python.org/3/library/argparse.html" \l "module-argparse" \o "argparse: Command-line option and argument parsing library.) will figure out how to parse those out of [sys.argv](https://docs.python.org/3/library/sys.html" \l "sys.argv" \o "sys.argv). The [argparse](https://docs.python.org/3/library/argparse.html" \l "module-argparse" \o "argparse: Command-line option and argument parsing library.) module also automatically generates help and usage messages and issues errors when users give the program invalid arguments.

.**ArgumentParser**(prog=None, usage=None, description=None, epilog=None, parents=[], formatter\_class=argparse.HelpFormatter, prefix\_chars='-', fromfile\_prefix\_chars=None, argument\_default=None, conflict\_handler='error', add\_help=True, allow\_abbrev=True, exit\_on\_error=True)

Create a new [ArgumentParser](https://docs.python.org/3/library/argparse.html" \l "argparse.ArgumentParser" \o "argparse.ArgumentParser) object. All parameters should be passed as keyword arguments. Each parameter has its own more detailed description below, but in short they are:

* [prog](https://docs.python.org/3/library/argparse.html#prog) - The name of the program (default: sys.argv[0])
* [usage](https://docs.python.org/3/library/argparse.html#usage) - The string describing the program usage (default: generated from arguments added to parser)
* [description](https://docs.python.org/3/library/argparse.html#description) - Text to display before the argument help (default: none)
* [epilog](https://docs.python.org/3/library/argparse.html#epilog) - Text to display after the argument help (default: none)
* [parents](https://docs.python.org/3/library/argparse.html#parents) - A list of [ArgumentParser](https://docs.python.org/3/library/argparse.html" \l "argparse.ArgumentParser" \o "argparse.ArgumentParser) objects whose arguments should also be included
* [formatter\_class](https://docs.python.org/3/library/argparse.html#formatter-class) - A class for customizing the help output
* [prefix\_chars](https://docs.python.org/3/library/argparse.html#prefix-chars) - The set of characters that prefix optional arguments (default: ‘-‘)
* [fromfile\_prefix\_chars](https://docs.python.org/3/library/argparse.html#fromfile-prefix-chars) - The set of characters that prefix files from which additional arguments should be read (default: None)
* [argument\_default](https://docs.python.org/3/library/argparse.html#argument-default) - The global default value for arguments (default: None)
* [conflict\_handler](https://docs.python.org/3/library/argparse.html#conflict-handler) - The strategy for resolving conflicting optionals (usually unnecessary)
* [add\_help](https://docs.python.org/3/library/argparse.html#add-help) - Add a -h/--help option to the parser (default: True)
* [allow\_abbrev](https://docs.python.org/3/library/argparse.html#allow-abbrev) - Allows long options to be abbreviated if the abbreviation is unambiguous. (default: True)
* [exit\_on\_error](https://docs.python.org/3/library/argparse.html#exit-on-error) - Determines whether or not ArgumentParser exits with error info when an error occurs. (default: True)

*Changed in version 3.5:*allow\_abbrev parameter was added.

*Changed in version 3.8:*In previous versions, allow\_abbrev also disabled grouping of short flags such as -vv to mean -v -v.

1. **Numpy:** NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

* **Operations using NumPy:**

Using NumPy, a developer can perform the following operations −

* + Mathematical and logical operations on arrays.
  + Fourier transforms and routines for shape manipulation.
  + Operations related to linear algebra.
  + NumPy has in-built functions for linear algebra and random number generation.
* **NumPy – A Replacement for MatLab :**

NumPy is often used along with packages like SciPy (Scientific Python) and Mat−plotlib (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

**It is open source, which is an added advantage of NumPy.**

**The best way to enable NumPy is to use an installable binary package specific to your operating system. These binaries contain full SciPy stack (inclusive of NumPy, SciPy, matplotlib, IPython, SymPy and nose packages along with core Python).**

|  |
| --- |
| **5. Matplot:** Matplot library is a python library used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc. It supports a very wide variety of graphs and plots namely - histogram, bar charts, power spectra, error charts etc. It is used along with NumPy to provide an environment that is an effective open source alternative for MatLab. It can also be used with graphics toolkits like PyQt and wxPython.  ***Types of Plots:***  *There are various plots which can be created using python matplotlib.*  Some of them are listed below:    **Fig: Types of plots**  There are several toolkits which are available that extend python matplotlib functionality. Some of them are separate downloads, others can be shipped with the matplotlib source code but have external dependencies.  **Basemap:** It is a map plotting toolkit with various map projections, coastlines and political boundaries.  **Cartopy:** It is a mapping library featuring object-oriented map projection definitions, and arbitrary point, line, - polygon and image transformation capabilities.  **Excel tools:** Matplotlib provides utilities for exchanging data with Microsoft Excel.  **Mplot3d:** It is used for 3-D plots.  **Natgrid:** It is an interface to the natgrid library for irregular gridding of the spaced data. |

➢ **NumPy - Ndarray Object:**

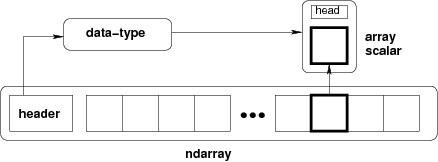
The most important object defined in NumPy is an N-dimensional array type called Ndarray. It describes the collection of items of the same type. Items in the collection can be accessed using a

zero-based index.

Every item in an Ndarray takes the same size of block in the memory. Each element in Ndarray is an object of data-type object (called dtype).

Any item extracted from ndarray object (by slicing) is represented by a

Python object of one of array scalar types. The following diagram shows a −



**Fig: Relationship between ndarray, data type object (dtype) and array scalar type**

An instance of ndarray class can be constructed by different array creation routines described later in the tutorial. The basic ndarray is created using an array function in NumPy as follows:

**NumPy. Array**

It creates an ndarray from any object exposing array interface, or from any method that returns an array.

NumPy. Array (object, dtype = None, copy = True, order = None, subok = False, ndmin = 0)

The ndarray object consists of contiguous one-dimensional segment of computer memory, combined with an indexing scheme that maps each item to a location in the memory block. The memory block holds the elements in a row-major order (C style) or a column-major order (FORTRAN or MatLab style).

* **Imutils:**

Before we continue to the code we need install imutils.  
Imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

For installing:

Open your Command Prompt and install it via:

pip install imutils

• **There are three ways to use Matplotlib:**

* **pyplot**: The module used so far in this article
* **pylab:** A module to merge Matplotlib and NumPy together in an environment closer to MATLAB
* **Object-oriented way**: The Pythonic way to interface with Matplotlib

**matplotlib.pyplot** is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

**6. Pandas:** Pandas is an opensource Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named [Numpy,](https://www.activestate.com/products/python/python-packages/) which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages, Pandas works well with many other [data science](https://www.activestate.com/products/python/python-data-science/) modules inside the Python ecosystem, and is typically included in every Python distribution, from those that come with your operating system to commercial vendor distributions like Active State’s [ActivePython.](https://platform.activestate.com/featured-projects)

➢ **Key Features of Pandas:**

* Fast and efficient Data Frame object with default and customized indexing.
* Tools for loading data into in-memory data objects from different file formats.
* Data alignment and integrated handling of missing data.
* Reshaping and pivoting of date sets.
* Label-based slicing, indexing and subsetting of large data sets.
* Columns from a data structure can be deleted or inserted.
* Group by data for aggregation and transformations.
* High performance merging and joining of data.
* Time Series functionality.

##### Pandas deal with the following three data structures: −

* Series
* Data Frame
* Panel

These data structures are built on top of Numpy array, which means they are fast.

**Dimension & Description**

The best way to think of these data structures is that the higher dimensional data structure is a container of its lower dimensional data structure. For example, DataFrame is a container of Series, Panel is a container of DataFrame.

|  |  |  |
| --- | --- | --- |
| Data  Structure | Dimensions | Description |
| Series | 1 | 1D labelled homogeneous array, size immutable. |
| Data Frames | 2 | General 2D labelled, size-mutable tabular structure with potentially heterogeneously typed columns. |
| Panel | 3 | General 3D labelled, size-mutable array. |

Data structures of Pandas

**Creating a Data Frame from Dictionary of Series:**

Dictionary of Series can be passed to form a DataFrame. The resultant index is the union of all the series indexes passed.

**Example:**

**import pandas as pd**

**d = {'one' : pd.**Series**([1, 2, 3], index=['a', 'b', 'c']), 'two' : pd.**Series**([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])} df = pd.**DataFrame**(d) print df**

**Its output is as follows − one two a 1.0 1 b 2.0 2**

1. **3.0 3**
2. **NaN 4**

* **tensorflow/keras:**
* **KERAS:**

Like TensorFlow, Keras is an open-source, ML library that’s written in Python. The biggest difference, however, is that Keras wraps around the functionalities of other ML and DL libraries, including TensorFlow, Theano, and CNTK. Because of TF’s popularity, Keras is closely tied to that library.

Many users and data scientists, us included, like using Keras because it makes TensorFlow much easier to navigate—which means you’re far less prone to make models that offer the wrong conclusions.

Keras builds and trains neural networks, but it is user friendly and modular, so you can experiment more easily with deep neural networks. Keras is a great option for anything from fast prototyping to state-of-the-art research to production. The key advantages of using Keras, particularly over TensorFlow, include:

* **Ease of use.** The simple, consistent UX in Keras is optimized for use cases, so you get clear, actionable feedback for most errors.
* **Modular composition.** Keras models connect configurable building blocks, with few restrictions.
* **Highly flexible and extendable.** You can write custom blocks for new research and create new layers, loss functions, metrics, and whole models.

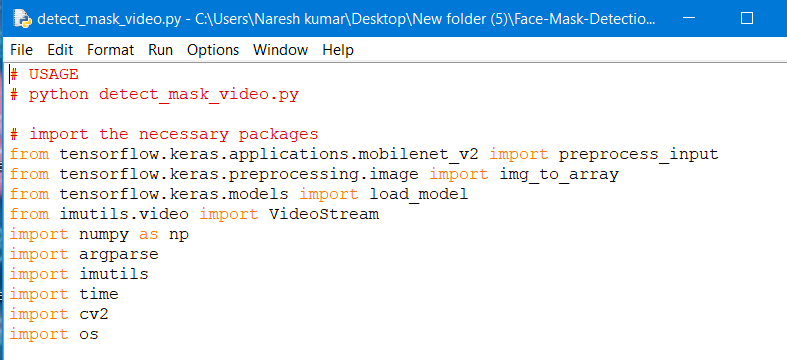
So here, we use Keras because it offers something unique in machine learning i.e single API that works across several ML frameworks to make that work easier.

**Hardware and Software Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Requirements** | **Type** | **Requirement Description** |
| 1. | Hardware  Requirements | Processer | i3 or above with a Supported  GPU |
| RAM | 8 GB RAM |
| Hard Disk space | 100 GB Free disk spaces |
| 2. | Software  Requirements | Operating System | Windows 10/ Windows server 2012 |
| Prerequisite | Python (3+), Keras,  Annaconda and supporting Libraries |
| Other | Administrator & internet access is required in the  windows machine, it should be open environment. |
| Application access | VPN access (If required),  Portal access, Application access, shared point access, SMTP port & credentials. |
| Browser | Google chrome, for JupyterNotebook |

### 1. Import the libraries and load the dataset

First, we are going to import all the modules that we are going to need for training our model. The Keras library already contains some datasets and MNIST is one of them. So we can easily import the dataset and start working with it. The **mnist.load\_data()** method returns us the training data, its labels and also the testing data and its labels.



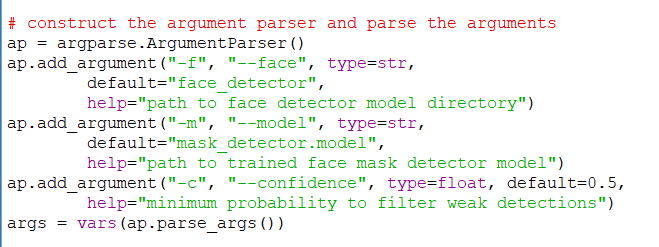
### 2. Preprocess the data

The image data cannot be fed directly into the model so we need to**perform some operations and process the data** to make it ready for our neural network.

This dataset consists of 4095 images belonging to two classes:

With \_mask: 2165 images

Without \_mask: 1930 images

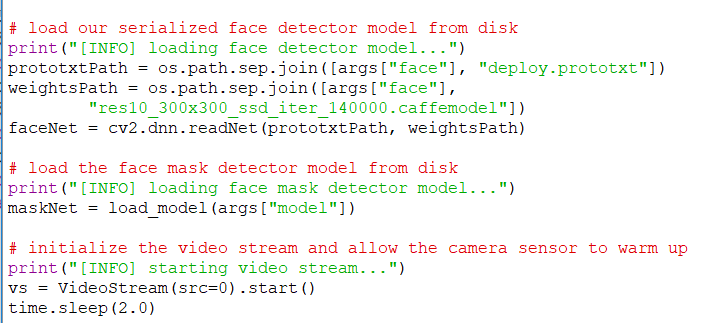


### 3. Create the model

In the part we’ll learn about mask detection, including the steps required to automatically predict the mask wearing by a person from an image or a video stream (and why mask detection is best treated as a classification problem rather than a regression problem).

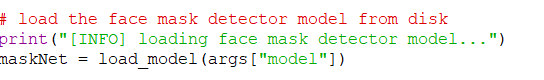
From there, we’ll discuss our deep learning-based mask detection model and then learn how to use the model for both:

* Face mask detection in static images
* Face mask detection in real-time video streams



### 4. Train the model

Once your face detector has produced the bounding box coordinates of the face in the image/video stream, you can move on to Stage #2 — identifying the person who were masks or who is not.

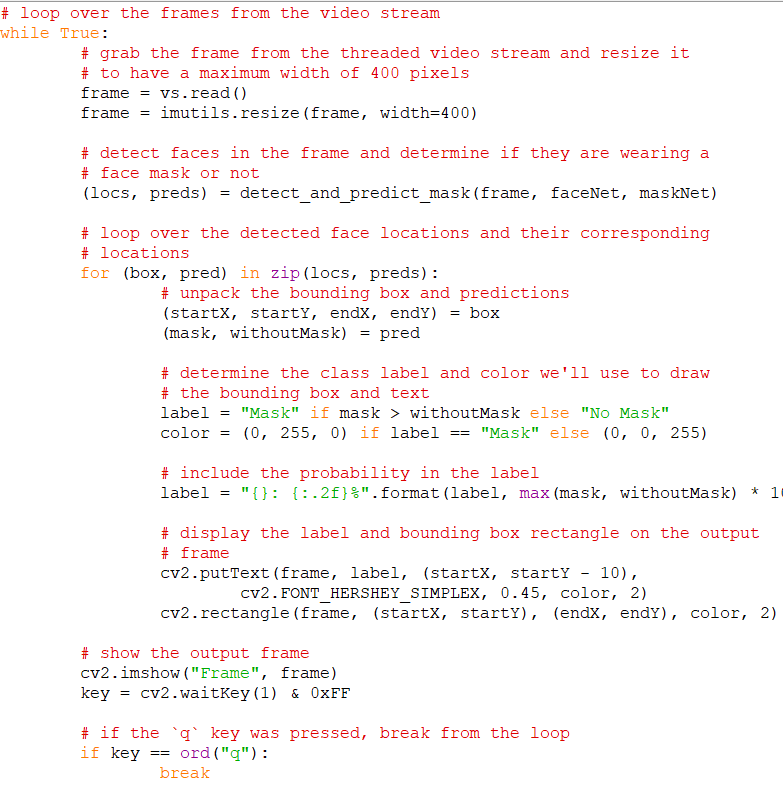


### 5. Evaluate the model

Mask detection is the process of automatically discerning the age of a person solely from a photo or video of their face.

Typically, you’ll see mask detection implemented as a two-stage process:

* Stage #1: Detect faces in the input image/video stream
* Stage #2: Display on the screen the person is wearing the mask or not.



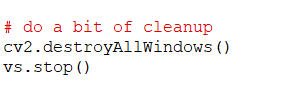
### 6. Running real time embedded system

For Stage #1, any face detector capable of producing bounding boxes for faces in an image can be used, including but not limited to Haar cascades, HOG + Linear SVM, Single Shot Detectors (SSDs), etc.

Exactly which face detector you use depends on your project:

* Haar cascades will be very fast and capable of running in real-time on embedded devices — the problem is that they are less accurate and highly prone to false-positive detections
* HOG + Linear SVM models are more accurate than Haar cascades but are slower. They also aren’t as tolerant with occlusion (i.e., not all of the face visible) or viewpoint changes (i.e., different views of the face)
* Deep learning-based face detectors are the most robust and will give you the best accuracy, but require even more computational resources than both Haar cascades and HOG + Linear SVMs

When choosing a face detector for your application, take the time to consider your project requirements — is speed or accuracy more important for your use case? I also recommend running a few experiments with each of the face detectors so you can let the empirical results guide your decisions.



**Complete Source Code:**

**# USAGE**

**# python detect\_mask\_video.py**

**# import the necessary packages**

**from tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input**

**from tensorflow.keras.preprocessing.image import img\_to\_array**

**from tensorflow.keras.models import load\_model**

**from imutils.video import VideoStream**

**import numpy as np**

**import argparse**

**import imutils**

**import time**

**import cv2**

**import os**

**def detect\_and\_predict\_mask(frame, faceNet, maskNet):**

**# grab the dimensions of the frame and then construct a blob**

**# from it**

**(h, w) = frame.shape[:2]**

**blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300),**

**(104.0, 177.0, 123.0))**

**# pass the blob through the network and obtain the face detections**

**faceNet.setInput(blob)**

**detections = faceNet.forward()**

**# initialize our list of faces, their corresponding locations,**

**# and the list of predictions from our face mask network**

**faces = []**

**locs = []**

**preds = []**

**# loop over the detections**

**for i in range(0, detections.shape[2]):**

**# extract the confidence (i.e., probability) associated with**

**# the detection**

**confidence = detections[0, 0, i, 2]**

**# filter out weak detections by ensuring the confidence is**

**# greater than the minimum confidence**

**if confidence > args["confidence"]:**

**# compute the (x, y)-coordinates of the bounding box for**

**# the object**

**box = detections[0, 0, i, 3:7] \* np.array([w, h, w, h])**

**(startX, startY, endX, endY) = box.astype("int")**

**# ensure the bounding boxes fall within the dimensions of**

**# the frame**

**(startX, startY) = (max(0, startX), max(0, startY))**

**(endX, endY) = (min(w - 1, endX), min(h - 1, endY))**

**# extract the face ROI, convert it from BGR to RGB channel**

**# ordering, resize it to 224x224, and preprocess it**

**face = frame[startY:endY, startX:endX]**

**face = cv2.cvtColor(face, cv2.COLOR\_BGR2RGB)**

**face = cv2.resize(face, (224, 224))**

**face = img\_to\_array(face)**

**face = preprocess\_input(face)**

**# add the face and bounding boxes to their respective**

**# lists**

**faces.append(face)**

**locs.append((startX, startY, endX, endY))**

**# only make a predictions if at least one face was detected**

**if len(faces) > 0:**

**# for faster inference we'll make batch predictions on \*all\***

**# faces at the same time rather than one-by-one predictions**

**# in the above `for` loop**

**faces = np.array(faces, dtype="float32")**

**preds = maskNet.predict(faces, batch\_size=32)**

**# return a 2-tuple of the face locations and their corresponding**

**# locations**

**return (locs, preds)**

**# construct the argument parser and parse the arguments**

**ap = argparse.ArgumentParser()**

**ap.add\_argument("-f", "--face", type=str,**

**default="face\_detector",**

**help="path to face detector model directory")**

**ap.add\_argument("-m", "--model", type=str,**

**default="mask\_detector.model",**

**help="path to trained face mask detector model")**

**ap.add\_argument("-c", "--confidence", type=float, default=0.5,**

**help="minimum probability to filter weak detections")**

**args = vars(ap.parse\_args())**

**# load our serialized face detector model from disk**

**print("[INFO] loading face detector model...")**

**prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])**

**weightsPath = os.path.sep.join([args["face"],**

**"res10\_300x300\_ssd\_iter\_140000.caffemodel"])**

**faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)**

**# load the face mask detector model from disk**

**print("[INFO] loading face mask detector model...")**

**maskNet = load\_model(args["model"])**

**# initialize the video stream and allow the camera sensor to warm up**

**print("[INFO] starting video stream...")**

**vs = VideoStream(src=0).start()**

**time.sleep(2.0)**

**# loop over the frames from the video stream**

**while True:**

**# grab the frame from the threaded video stream and resize it**

**# to have a maximum width of 400 pixels**

**frame = vs.read()**

**frame = imutils.resize(frame, width=400)**

**# detect faces in the frame and determine if they are wearing a**

**# face mask or not**

**(locs, preds) = detect\_and\_predict\_mask(frame, faceNet, maskNet)**

**# loop over the detected face locations and their corresponding**

**# locations**

**for (box, pred) in zip(locs, preds):**

**# unpack the bounding box and predictions**

**(startX, startY, endX, endY) = box**

**(mask, withoutMask) = pred**

**# determine the class label and color we'll use to draw**

**# the bounding box and text**

**label = "Mask" if mask > withoutMask else "No Mask"**

**color = (0, 255, 0) if label == "Mask" else (0, 0, 255)**

**# include the probability in the label**

**label = "{}: {:.2f}%".format(label, max(mask, withoutMask) \* 100)**

**# display the label and bounding box rectangle on the output**

**# frame**

**cv2.putText(frame, label, (startX, startY - 10),**

**cv2.FONT\_HERSHEY\_SIMPLEX, 0.45, color, 2)**

**cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)**

**# show the output frame**

**cv2.imshow("Frame", frame)**

**key = cv2.waitKey(1) & 0xFF**

**# if the `q` key was pressed, break from the loop**

**if key == ord("q"):**

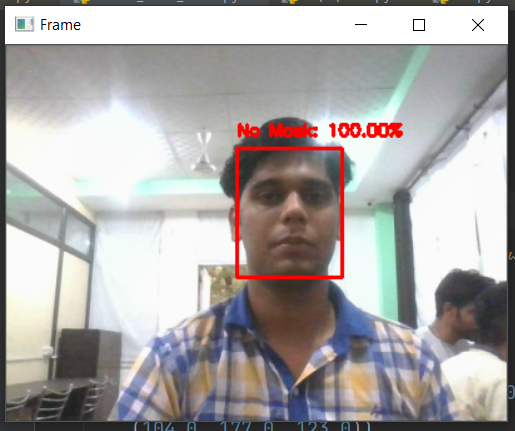
**break**

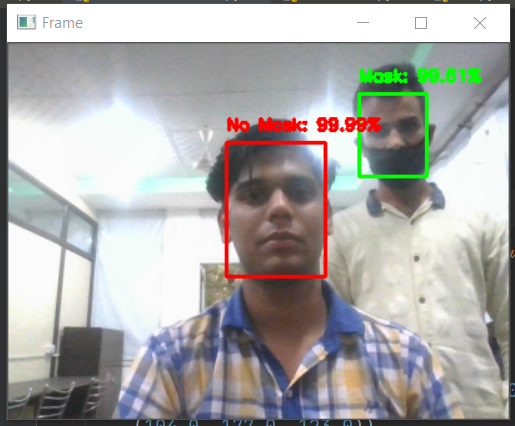
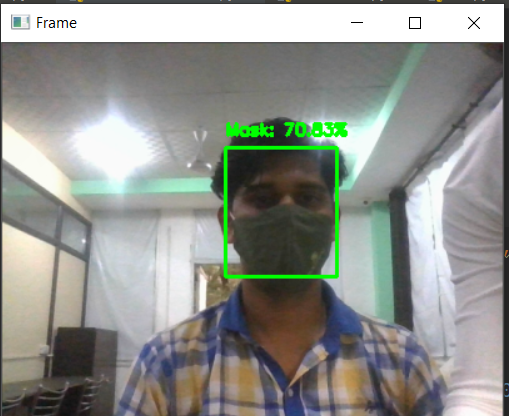
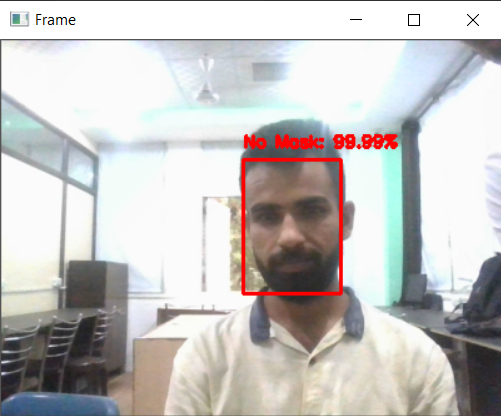
**# do a bit of cleanup**

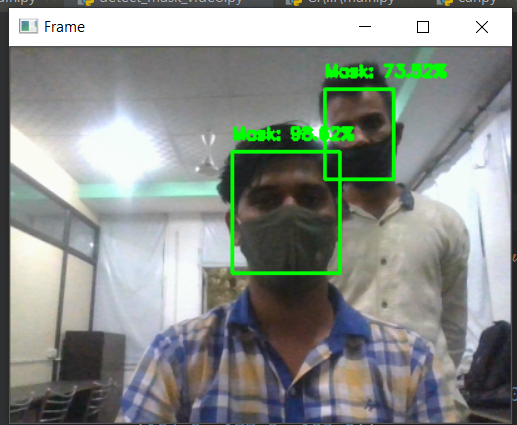
**cv2.destroyAllWindows()**

**vs.stop()**

**Screenshots:**







## Conclusion

This project presents a system for a smart city to reduce the spread of coronavirus by informing the authority about the person who is not wearing a facial mask that is a precautionary measure of COVID-19. The motive of the work comes from the people disobeying the rules that are mandatory to stop the spread of coronavirus. The system contains a face mask detection architecture where a deep learning algorithm is used to detect the mask on the face. To train the model, labeled image data are used where the images were facial images with masks and without a mask. The proposed system detects a face mask with an accuracy of 98.7%. The decision of the classification network is transferred to the corresponding authority. The system proposed in this study will act as a valuable tool to strictly impose the use of a facial mask in public places for all people.

**FUTURE SCOPE**

The developed system faces difficulties in classifying faces covered by hands since it almost looks like the person wearing a mask. While any person without a face mask is traveling on any vehicle, the system cannot locate that person correctly. For a very densely populated area, distinguishing the face of each person is very difficult. For this type of scenario, identifying people without face mask would be very difficult for our proposed system. In order to get the best result out of this system, the city must have a large number of CCTV cameras to monitor the whole city as well as dedicated manpower to enforce proper laws on the violators. Since the information about the violator is sent via SMS, the system fails when there is a problem in the network.

The proposed system mainly detects the face mask and informs the corresponding authority with the location of a person not wearing a mask. Based on this, the authority has to send their personnel to find out the person and take necessary actions. But this manual scenario can be automated by using drones and robot technology [22], [23] to take action instantly. Furthermore, people near to the person not wearing a mask may be alerted by an alarm signal on that location, and displaying the violators face in a LED screen to maintain a safe distance from the person would be a further study.