Face Mask Detection

-: Team Members :-

1. Aayush Ajaybhai Vaishnav - 0805055

2. Juned Aiyub Saleh - 0805065

3. Dharmkumar Chandubhai Kakadiya - 0785327

4. Deev Vijaybhai Viradiya – 0805400

5. Parth Manoj Joshi – 0805440

-:Final Project report:-

Introduction and Related Work

-:Roles of group members :-

1. Aayush Ajaybhai Vaishnav – Documentation , Report building and analysis.
2. Juned Aiyub Saleh – Frontend Development

3. Dharmkumar Chandubhai Kakadiya – Project code review and Github code management
4. Deev Vijaybhai Viradiya – Backend Development
5. Parth Manoj Joshi – System testing python code and Dataset collection

1. Aayush Ajaybhai Vaishnav – It is my responsibility to gather and record any pertinent information needed for the project. I ran into problems while creating the documentation, such as the fact that there is a tonne of information online on machine learning models and other projects that are similar to our one. Finding the appropriate word at first that fit the project concept and the mindset of each participant was a difficult assignment for me. Nevertheless, by the time the project's documentation and information gathering components needed my help, I had learnt and contributed.

2. Juned Aiyub Saleh – Since the project was primarily specified and created in Python code, my main challenge was fitting the exact requirements into the frontend development. So, for any developer, integrating any front-end framework with Python is a challenging undertaking.

Each contributor was initially working on their own research for the tasks that were assigned to them, as was the entire project scaffold. But for me, the front end was something that gave them all a totally different edge. However, following extensive conversation, I created front-end code and tested it locally. It is now functioning as expected and adhering to the project's specifications.

3. Dharmkumar Chandubhai Kakadiya – More difficult than writing code is reading and comprehending someone else's. My problem was that I needed to go over all of the potential failures that could happen while testing the code, which was developed by juned or deev. What if written code is not integrated with front-end code, for example. What if upgraded versions of any machine-leaning library contain problems caused by written code? because no one can predict when a library upgrade will take place. Because libraries are the portions that update themselves periodically

4. Deev Vijaybhai Viradiya – Python development is simple. Even when machine leaning is included, this is still simple. However, that model training was the biggest obstacle for me. Because of the provided dataset, I must train the model repeatedly in order to verify whether the written code is fitted accurately into the model or not. As a result, a local system must spend a lot of time on model training. But by the time we got the accuracy we expected, we were already doing a lot of caching to make the system run quickly.

5. Parth Manoj Joshi – Finding the dataset for a machine learning project is extremely difficult before the project even begins. Although there are millions of datasets available online, finding one that is perfectly

suited to a project is a difficult challenge. For instance, there are numerous cloud providers that offer face datasets, however for this project we need a dataset that offers both faces with and without masks. Moreover, I also need to learn about some of the most unique pictures. Using a hand instead of a mask, for example.

AIM:

to create a Face Mask Detection system using OpenCV, Keras/TensorFlow, and Deep Learning to find faces hidden in both still pictures and moving video.

OBJECTIVES:

- To develop a specific deep learning model for determining whether or not someone is wearing a mask.
- To create an artificial dataset of masked photos with machine vision assistance.
- There will be two types for datasets: "with mask" and "without mask."
- to use Keras and TensorFlow to train the face mask detector on the specific dataset.
- to put into practise the trained model for identifying masks in both real-time movies and static input photos.

Research:

→What objectives we wanted to achieve :

The objective of this study is to develop a Face Mask Detection system using OpenCV, Keras/TensorFlow, and Deep Learning in order to detect face masks in static images as well as live video streams.

→ABSTRACT (Why are we doing this project) :

In recent times, where Covid-19 has impacted a domino effect on manufacturing, travel, tourism, hospitality, crippling the global economy. In addition to it, is the growing curve of human deaths across the globe due to the pandemic, this project which relies on computer vision and deep learning, intends to make an impact and solve the real-world problem of safety measures at some significant level.

This project can be used at airports, offices, hospitals and many more public places to ensure that the safety standards are maintained, and people are abiding by the rules and regulations to wear protective masks at public places. If the detection system classifies as 'No Mask', reminders can be given as well as actions can be taken against such individuals.

RELATED WORK:

1. A thermal camera called FebriEye has extra analytics like a face mask and a social distance monitoring system that sends out an alert in the event of any infractions. Vehant Technologies is creating it, and the Telangana government will deploy it.
2. In nations like the US, Uber has acknowledged to CNN Business that it requires both drivers and passengers to wear face masks or other similar coverings, and it is working on technologies to monitor whether or not drivers are complying with these requirements.

3. The Face Mask Alert app, which LeewayHertz software solutions is now working on. The users receive a notification ordering them to put on masks.

Main purpose:

A face mask detector was created utilising TensorFlow/Keras, TensorFlow/OpenCV, and computer vision and deep learning techniques.

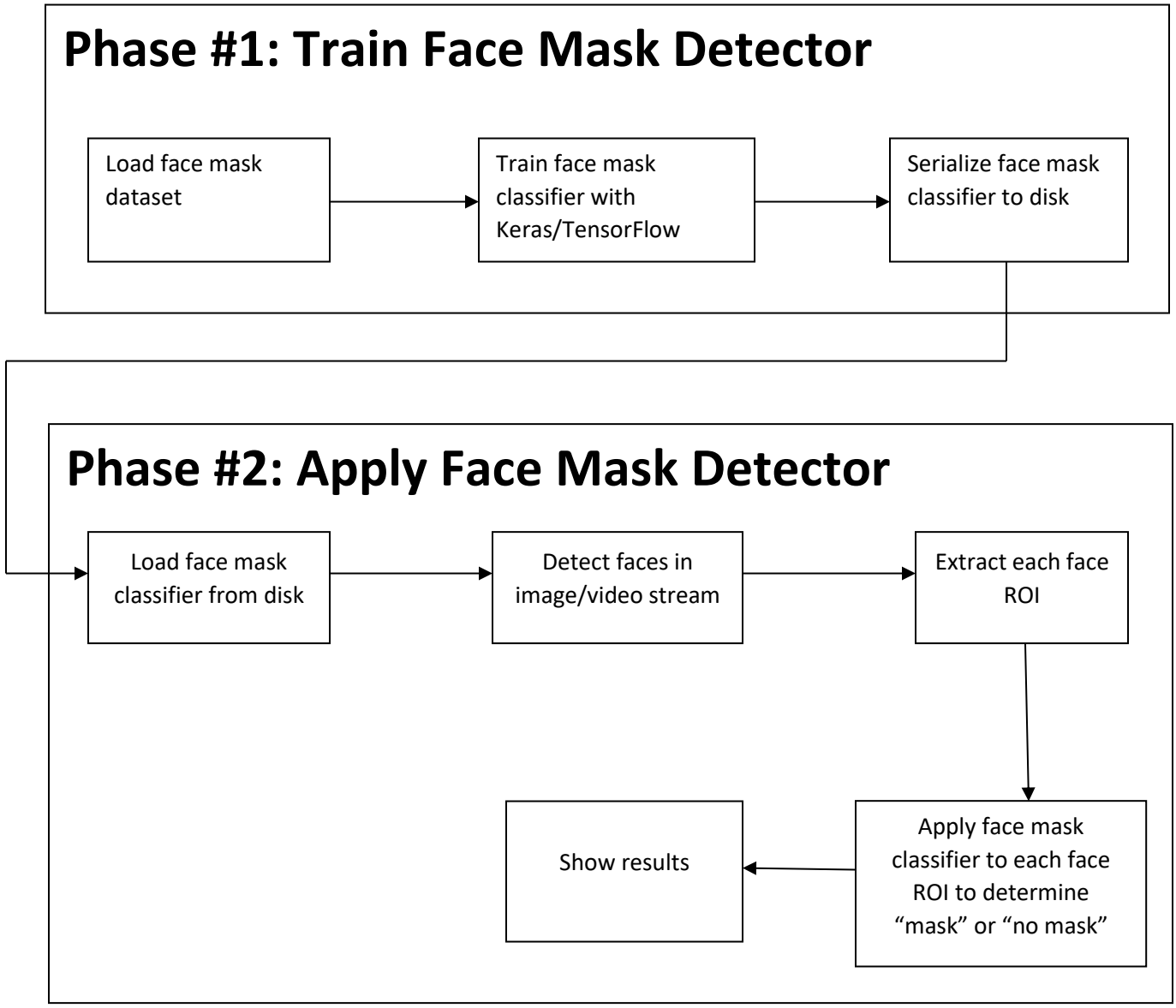


Figure 1: Phases and individual steps for building a COVID-19 face mask detector with computer vision and deep learning using Python, OpenCV, and TensorFlow/Keras.

OpenCV, and TensorFlow/Keras.

1. Training:

A. Loading the face mask detection dataset from disk

B. Training a model using Keras/TensorFlow on this loaded dataset

C. Serializing the face mask detector back to disk

2. Deployment:

A. Loading the face mask detector

B. Performing face detection

C. Classifying each face as “Mask” or “No Mask”

OUTCOMES EXPECTATION:

I. If a person is wearing a mask or not, the system will be able to identify them appropriately.

II. A customised face masking computer vision Python script that transforms ordinary photos into a fictitious (but still useful in the real world) dataset called "with mask."

III. A Python face mask detector training script that loads and pre-processes the photos, labels them using TensorFlow.keras and sklearn, and accepts the input dataset.

IV. Additionally, using pre-trained ImageNet weights, it would refine the model using the MobileNetV2 classifier.

V. Correctly recognise faces in static photographs stored in folders.

VI. The method created successfully detects face masks on every frame of the real-time video stream using your webcam.

VII. Returns whether the static images and live video streams under examination have a "Mask" or "No Mask" in them.

DATA COLLECTION

The data will be collected from the below mentioned sources:

✓ Real-World Masked Face Dataset (RMFD)

✓ Kaggle Datasets

✓ Bing Search API

More than 1800 images in both the "with mask" and "without mask" classes are to be gathered.

For the Bing Search API, a Python script will be used to find photos while searching for terms like "covid mask," "facial mask," and "N95 mask."

Real photographs of faces wearing and not wearing protective face masks make up the project's unique dataset.

This dataset consists of 3835 images belonging to two classes:

- with_mask: 1916 images
- without_mask: 1919 images

A total of more than 5000 photographs were gathered, however only a small number of them were accepted since they were distorted, blurry, or unproductive. They were eliminated, and as a result, data pruning was carried out.

With the aid of the Sklearn library, the dataset will be divided into 80% training and 20% testing data. Approximately 3068 photos make up the training set, whereas 767 images make up the testing set.

OUR CODE:

```
from requests import exceptions
import argparse
import requests
import cv2
import os
ap = argparse.ArgumentParser()
ap.add_argument("-q", "--query", required=True,
    help="search query to search Bing Image API for")
ap.add_argument("-o", "--output", required=True,
    help="path to output directory of images")
args = vars(ap.parse_args())
API_KEY = "d8982f9e69a4437fa6e10715d1ed691d"
MAX_RESULTS = 500
GROUP_SIZE = 50
URL = "https://api.cognitive.microsoft.com/bing/v7.0/images/search"
EXCEPTIONS = set([IOError, FileNotFoundError,
```

```
    exceptions.RequestException, exceptions.HTTPError,
    exceptions.ConnectionError, exceptions.Timeout])
term = args["query"]
headers = {"Ocp-Apim-Subscription-Key" : API_KEY}
params = {"q": term, "offset": 0, "count": GROUP_SIZE}
print("[INFO] searching Bing API for '{}'".format(term))
search = requests.get(URL, headers=headers, params=params)
search.raise_for_status()
results = search.json()
estNumResults = min(results["totalEstimatedMatches"], MAX_RESULTS)
print("[INFO] {} total results for '{}'".format(estNumResults,
    term))
total = 0
for offset in range(0, estNumResults, GROUP_SIZE):
    print("[INFO] making request for group {}-{} of {}...".format(
        offset, offset + GROUP_SIZE, estNumResults))
    params["offset"] = offset
    search = requests.get(URL, headers=headers, params=params)
    search.raise_for_status()
    results = search.json()
    print("[INFO] saving images for group {}-{} of {}...".format(
        offset, offset + GROUP_SIZE, estNumResults))
    for v in results["value"]:
        try:
            print("[INFO] fetching: {}".format(v["contentUrl"]))
            r = requests.get(v["contentUrl"], timeout=30)
            ext = v["contentUrl"][v["contentUrl"].rfind("."):]
            p = os.path.sep.join([args["output"], "{}{}".format(
                str(total).zfill(8), ext)])
            f = open(p, "wb")
            f.write(r.content)
            f.close()
        except Exception as e:
            if type(e) in EXCEPTIONS:
                print("[INFO] skipping: {}".format(v["contentUrl"]))
                continue
        image = cv2.imread(p)
        if image is None:
            print("[INFO] deleting: {}".format(p))
            os.remove(p)
            continue
        total += 1
```

SEQUENCE DIAGRAM:

Sequence Diagrams are interaction diagrams that describe the steps used to complete an operation. They depict how items interact within the framework of a cooperation. By using the vertical axis of the diagram to represent time and the messages that are transmitted and when, sequence diagrams, which have a time focus, can visually depict the order of an interaction.

Sequence Diagrams captures:

• the conversation that occurs during a collaboration that either executes an operation or a use case (instance diagrams or generic diagrams)

• High-level communications between the system and its user, with other systems, or with its subsystems (sometimes known as system sequence diagrams)

Purpose of Sequence Diagram

• Model the interaction between object instances within a collaboration that realises a use case

• Model the interaction between objects within a collaboration that realises an operation

• Either model generic interactions (showing all possible paths through the interaction) or specific instances of an interaction.

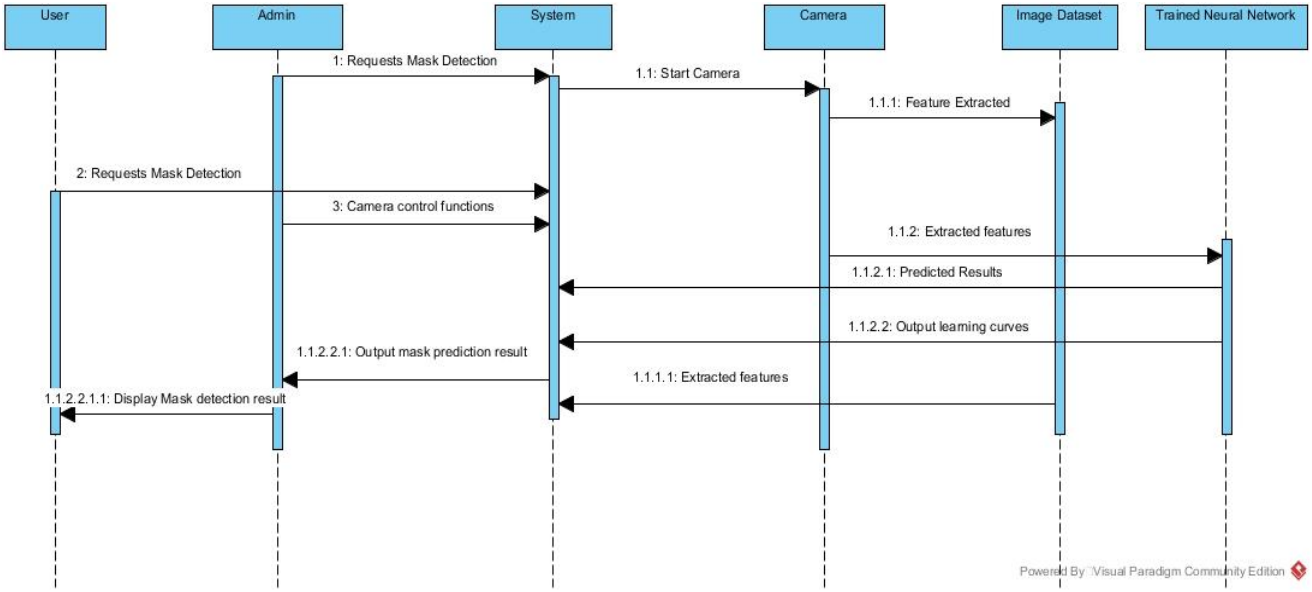
• Model high-level interaction between active objects in a system (showing just one path through the interaction)

Sequence Diagram at a Glance

Sequence Diagrams are arranged in accordance with object (horizontally) and time to depict elements as they interact across time (vertically)

Sequence diagrams can include annotations and constraints just like any other diagram.

SEQUENCE DIAGRAM FOR FACE MASK DETECTION SYSTEM



USE CASE DIAGRAM:

The main representation of system/software requirements for a new, developing software programme is a use case diagram. Use cases describe the desired behaviour (what), not the precise means of achieving it (how).

Use case modeling's ability to assist in system design from the standpoint of the end user is a crucial idea. By describing all externally observable system behaviour, it is a useful tool for explaining system behaviour to users.

Use Case Diagram captures:

• Only a few of the connections between use cases, actors, and systems are summarised.

• It omits displaying the sequence in which each use case's objectives are accomplished.

Purpose of Use Case Diagram:

• Describe the system's context.

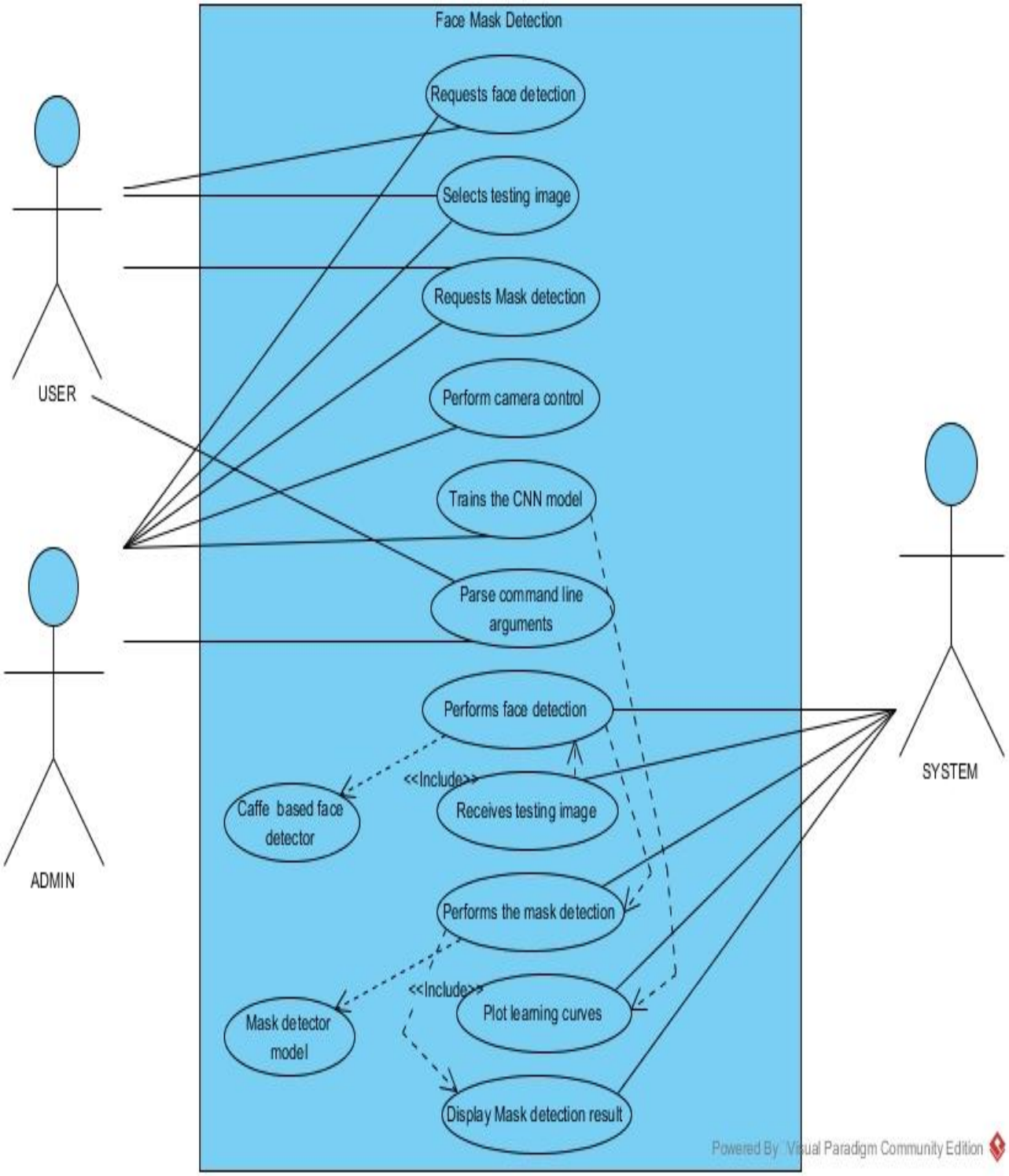
• Record a system's requirements.

• Confirm a system architecture

• Lead implementation efforts and produce test cases

• Created by analysts and subject-matter experts

USE CASE DIAGRAM FOR FACE MASK DETECTION SYSTEM



ACTIVITY DIAGRAM:

Activity diagrams show how multiple levels of abstraction of activities are coordinated to produce a service. Typically, an event must be accomplished by some operations, especially when the operation is meant to accomplish several different things that call for coordination. Another common requirement is how the events in a single use case relate to one another, especially in use cases where activities may overlap and require coordination.

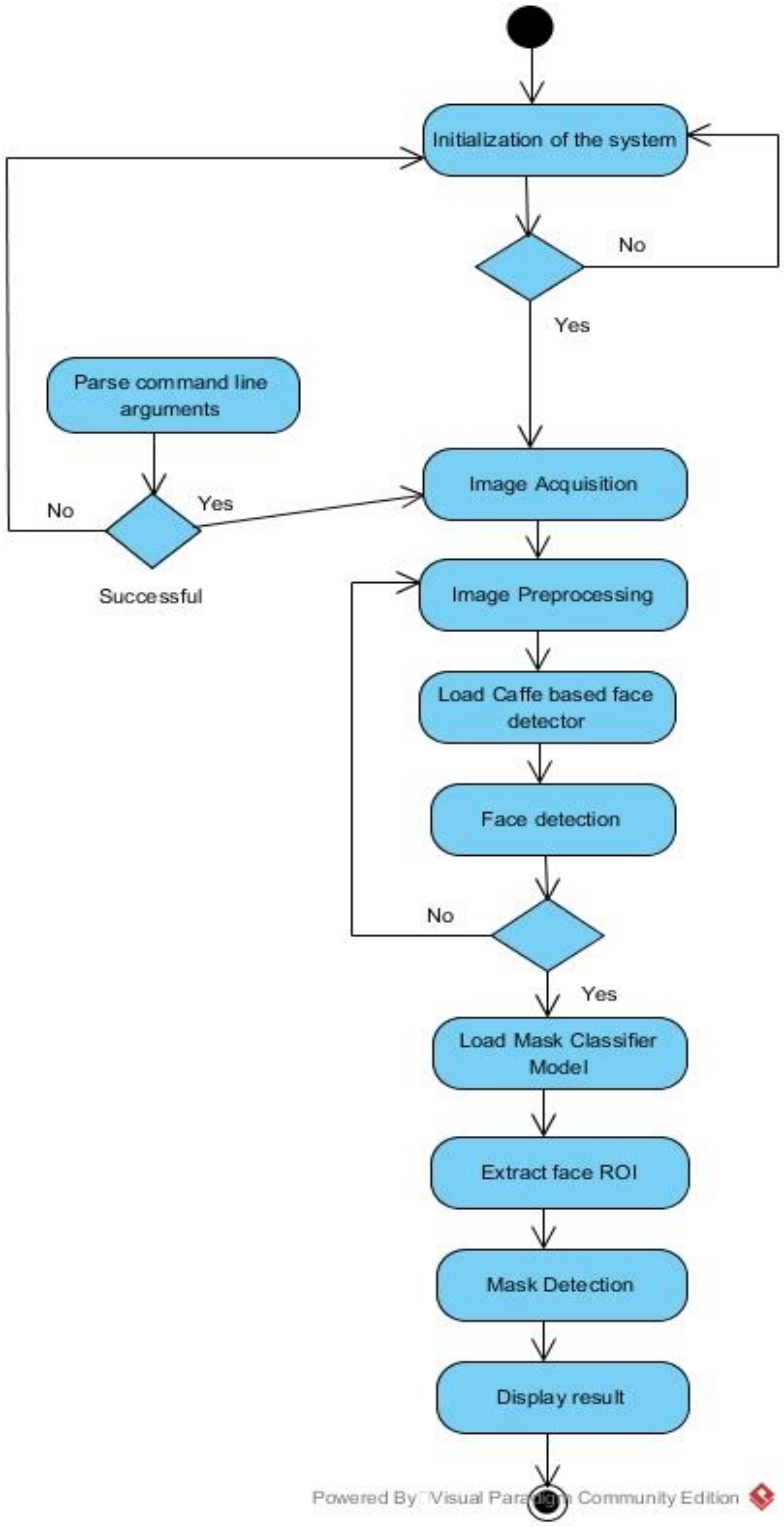
Purpose of Activity Diagram:

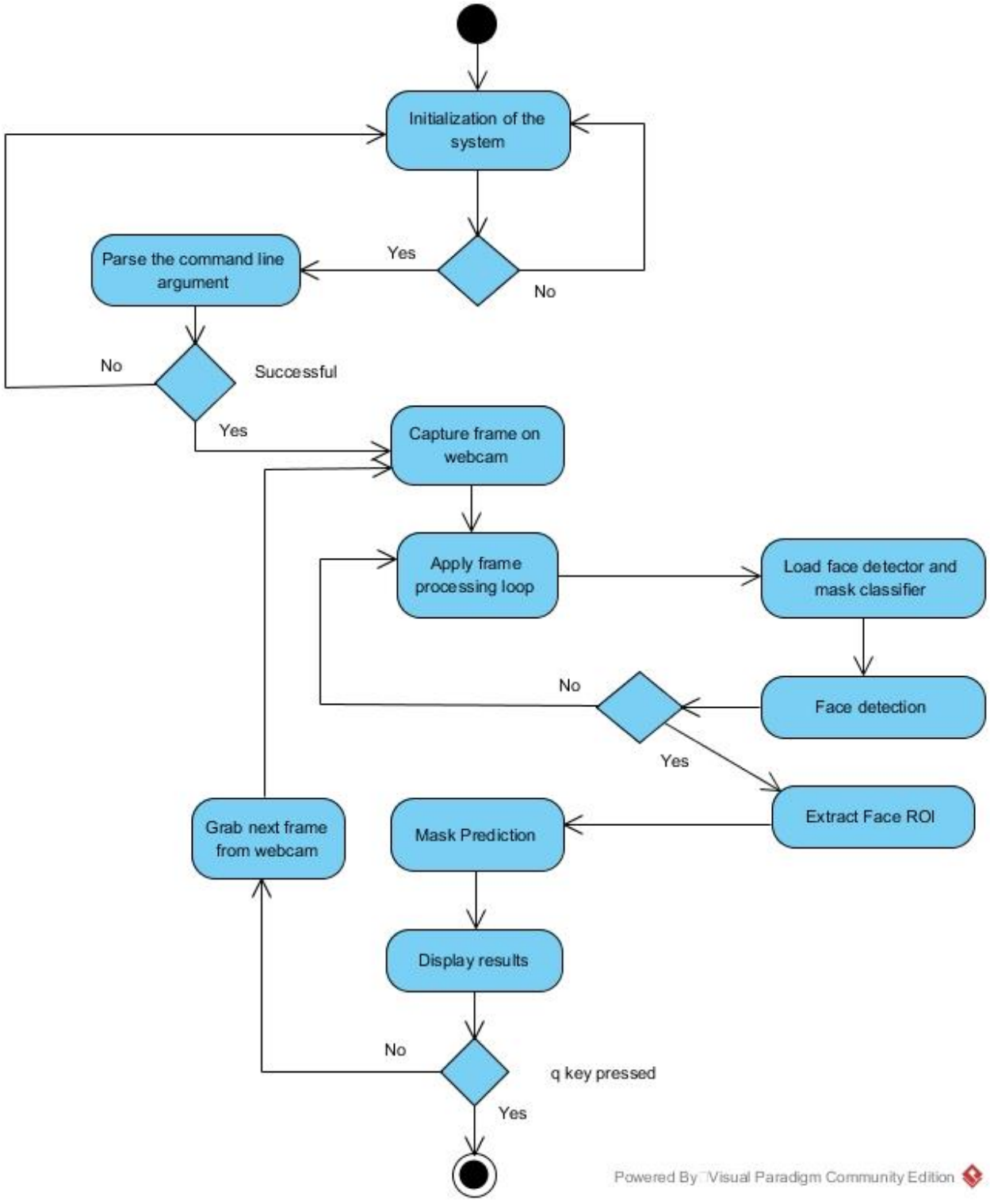
- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

Activity diagram can be used for:

- Modeling work flow by using activities.
- Modeling business requirements.
- High level understanding of the system's functionalities.
- Investigating business requirements at a later stage.

ACTIVITY DIAGRAM OF FACE MASK DETECTION SYSTEM




Initialization of the system
Yes
No
Parse the command line argument
No
Successful
Yes
Capture frame on webcam
Apply frame processing loop
Load face detector and mask classifier
No
Face detection
Yes
Extract Face ROI
Grab next frame from webcam
Mask Prediction
Display results
No
q key pressed
Yes
Powered By Visual Paradigm Community Edition

CLASS DIAGRAM:

The fundamental units of all object-oriented approaches are class diagrams. The classes, relationships, interface, affiliation, and collaboration can all be displayed using the class diagram.

Class diagrams are a type of static structure diagram that illustrates the classes, attributes, operations (or methods), and relationships between objects in a system to describe the structure of the system.

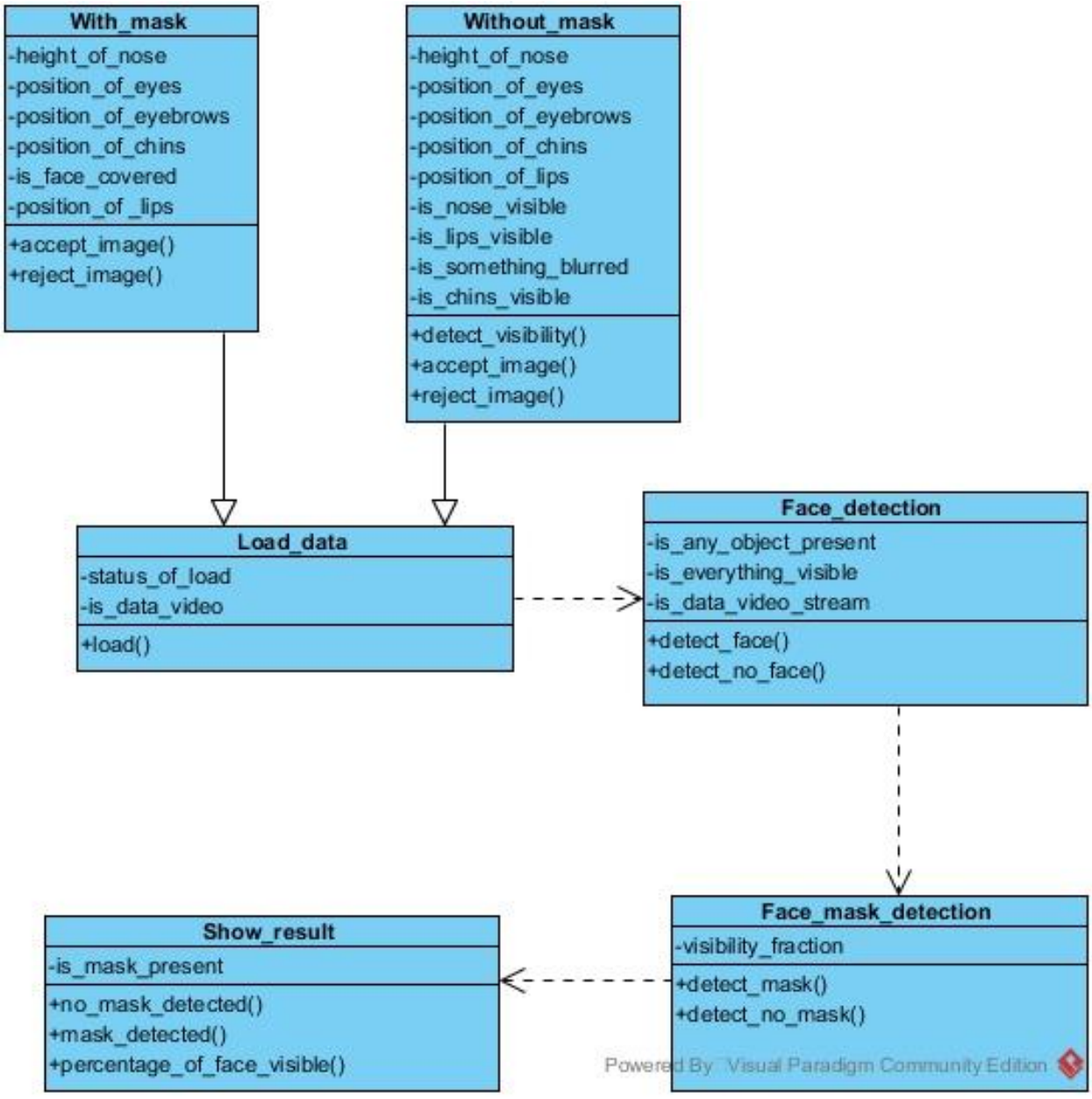
Purpose of Class Diagram:

- Displays a system's classifiers' static structure
- Diagram offers a fundamental notation for additional UML-required structure diagrams.
- Beneficial to both developers and the rest of the team
- Business analysts can model systems from a business perspective using class diagrams.

A UML class diagram is made up of:

- A set of classes and
- A set of relationships between classes

CLASS DIAGRAM OF FACE MASK DETECTION SYSTEM



METHODOLOGIES:

1. Real world mask face dataset (RMFD)
2. Bing search API

RMFD Dataset | Papers With Code

➔ Finding suitable machine learning algorithm

1. Transfer learning algorithm
2. Convolution neural netwok (CNN/deep learning)

→"Deep neural networks" (dnn) module of OpenCV The extremely effective dnn module is part of OpenCV (3.3 or later), and it is supported by several deep learning frameworks, including Caffe, TensorFlow, and Torch/PyTorch. The face detector in this module is based on Caffe and is more precise. The following files are required because we will be using Caffe to train our deep learning model in this project.

→Using MobileNetV2 architecture and the Single Shot Multibox Detector (SSD) framework, OpenCV's face detector: We must use object detection to determine the bounding box (x, y) coordinates for an object (mask in this case).

→Image pre-processing is made easier by OpenCV's blobFromImage and blobFromImages functions. The dnn module of OpenCV's cv2.dnn.blobFromImage and cv2.dnn.blobFromImages functions makes it easier to prepare images for deep learning categorization. These two activities accomplish:

→Keras ImageDataGenerator: Using real-time data augmentation, create batches of tensor image data. Iterative looping of the data (in batches).

References:

[1] https://keras.io/api/applications/mobilenet/#mobilenetv2-function

[2] https://arxiv.org/abs/1704.04861

[3] https://www.pyimagesearch.com/2019/06/03/fine-tuning-with-keras-and-deep-learning/

[4] https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset

[5] https://github.com/AIZOOTech/FaceMaskDetection

[6] https://www.pyimagesearch.com/2018/04/09/how-to-quickly-build-a-deep-learningimage-dataset/

[7] https://gogul.dev/software/flower-recognition-deep-learning

[8] https://www.tensorflow.org/tutorials/images/transfer_learning

[9] https://towardsdatascience.com/detecting-faces-with-python-and-opencv-facedetection-neural-network-f72890ae531c

[10] https://www.indulgexpress.com/msociety/2020/may/04/hyderabad-companylaunches-ai-based-surveillance-tech-to-detect-face-mask-and-social-distancing-viol24656.html

[11] https://www.engadget.com/uber-face-mask-detection-technology-184137514.html

[12] https://www.leewayhertz.com/face-mask-detection-system/

[13] https://www.thehansindia.com/telangana/hyderabad-cameras-to-detect-face-maskstest-temp-soon-619505

[14] https://www.pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-dataaugmentation/