**MINI PROJECT**

**on**

**FACE MASK DETECTION USING REACT**

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfilment of the requirement for the award of the degree of

**BACHELOR OF TECHNOLOGY**

**in**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**By**

|  |
| --- |
|  |
| **TADAKA RAHUL** | | **187Y1A0492** |
| **KOTHA BHARDWAJ** | | **187Y1A0471** |

**Under the Guidance of**

Mrs. T. Tanuja, ASSISTANT ROFESSOR



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**



*i*

****

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

Date: 27-oct-2021

**CERTIFICATE**

This is to certify that the project work entitled “**FACE MASK DETECTION USING REACT**” work done by  **T. RAHUL(187Y1A0492) and K. BHARDWAJ (187Y1A0471)** student(s) of Department of Electronics and Communication Engineering, is a record of bonafide work carried out by the member during a period from April, 2021 to October, 2021 under the supervision of **T. Tanuja**. This project is done as a fulfilment of obtaining Bachelor of Technology Degree to be awarded by Jawaharlal Nehru Technological University Hyderabad, Hyderabad.

The matter embodied in this project report has not been submitted by us to any other university for the award of any other degree.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| T.RAHUL | K.BHARDWAJ |  |

This is to certify that the above statement made by the candidate(s) is correct to the best of my knowledge.

Date: ( **T.Tanuja )**

The Viva-Voce Examination of above student(s), has been held on………………………

|  |  |
| --- | --- |
| Head of the Department | External Examiner |
| Principal  ***ii*** |  |
|  | |
|  | |

**Text

Description automatically generated**

**ACKNOWLEDGEMENTS**

we would like to express my sincere gratitude to my guide **Mrs. Tanuja Assistance Professor**, Department of Electronics and Communication Engineering, for his excellent guidance and invaluable support, which helped me accomplish the B.Tech (ECE) degree and prepared me to achieve more life goals in the future. His total support of my dissertation and countless contributions to my technical and professional development made for a truly enjoyable and fruitful experience. Special thanks are dedicated for the discussions we had on almost every working day during my project period and for reviewing my dissertation.

We are very much grateful to my Project Coordinator**, Dr. G. Amarnath Associate Professor,** Department of ECE, MLRITM, Dundigal, Hyderabad, who has not only shown utmost patience, but was fertile in suggestions, vigilant in directions of error and has been infinitely helpful. we are extremely grateful to **Dr. Srinivas Bachu**, **Professor & HOD-ECE**, MLRITM, Dundigal, Hyderabad, for the moral support and encouragement given in completing my project work. we wish to express deepest gratitude and thanks to **Dr. K. Venkateswara Reddy, Principal,** MLRITM, for his constant support and encouragement in providing all the facilities in the college to do the project work.

we would also like to thank all our faculties, administrative staff and management of MLRITM, who helped me to completing the mini project. On a more personal note, I thank my **beloved parents and friends** for their moral support during the course of our project

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | **Page No.** |
| *Certificate* | | | | *2* |
| *Acknowledgements* | | | | *3* |
| *Table of Contents* | | | | *6* |
| *List of Figures* | | | | *8* |
| *List of Abbreviations* | | | | *9* |
| *List of Tables* | | | | *10* |
| *Abstract* | | | | *11* |
| **Chapter 1: Introduction** | | | |  |
|  | 1.1 | In Detail | | 12 |
|  | 1.2 | The implementation | | 14 |
|  |  | 1.2.1 | TensorFlow.js | 14 |
|  |  | 1.2.2 | Reactjs | 15 |
|  | 1.3 | Image size | | 19 |
|  |  | 1.3.1 | Additional Resources | 21 |
|  |  | 1.3.2 | Dataset | 21 |
|  | 1.4 | 1.4 | Possible approaches to run neural network models in browser: | 22 |
|  |  | 1.4.2 | How to Increase Accuracy, Decrease False Positives, Run Access Control API for Stable Faces Only | 25 |
|  |  | 1.4.3 | Performance Results: Getting Frames | 26 |
| **Chapter 2: Literature Survey** | | | |  |
|  | 2.1 |  | Literature Survey | 28 |
| **Chapter 3: Code** | | | |  |
|  | 3.1 |  | Code testing | 29 |
|  | 3.2 | Command | | |
|  |  | 3.2.1 | npm start | 29 |
|  |  | 3.2.2  3.2.3 | npm test  npm run build | 30 |
|  |  | 3.2.4 | npm run eject | 31 |
|  | 3.3.2 | The code | | 33 |
| ***Chapter 4: Result*** | | | | 37 |
| ***Chapter 5 : conclusion*** | | | | 39 |
| ***References*** | | | | 44 |
| ***Publications (if any)*** | | | |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Name of the Figure** | **Page No.** |
| Figure 1 | Example for facemask detection | 13 |
| Figure 2 | React modal | 16 |
| Figure 3 | Mask detection in various devices | 20 |
| Figure 4 | Run neural network | 22 |
| Figure 5 | Mapping of mask | 23 |
| Figure 6 | Web Worker usage to run models in different context and have parallelism in the browser | 25 |
| Figure 7 | Output 1 | 37 |
| Figure 8 | Output 2 | 38 |
| Figure 9 | Appropriate scheme describes data | 40 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| CNN | Neural Network |
| ANN | Artificial neural network |
| WASW | WebAssembly |
| WebGL | Web Graphics Library |
| CPU | Central processing unit |
| HTML | HyperText Markup Language |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Name of the Table** | **Page No.** |
| Table 1 | Color scheme: < 6 fps red, 7-12 fps orange, 13-18 fps yellow, 19+ fps green |  |
| Table 2 | Here are some measurements for WW processing time on different devices |  |
| Table 3 | Single mode it is max performance of getting frame. Sync mode it's how often we can get frame with face detection. |  |

**Text

Description automatically generated**

**ABSTRACT**

The present scenario of COVID-19 demands an efficient face mask detection application. The main goal of the project is to implement this system at entrances of colleges, airports, hospitals and offices where chances of spread of COVID-19 through contagion are relatively higher.  Reports indicate that wearing face masks while at work clearly reduces the risk of transmission.  It is an object detection and classification problem with two different classes (Mask and Without Mask). A hybrid model using deep and classical machine learning for detecting face mask will be presented. A dataset is used to build this face mask detector using CNN (Convolutional Neural Network). We have already trained by using dataset in CNN and deployed in REACT (Tensor flow JS). The code in face mask detection is java script. When a person entering in the surrounding of the device it identifies the human and then it detects the person is with mask or person is without mask . If anyone is found to be without a face mask, then it shows that he doesn’t wear the mask.

As all the workplaces and education institutes are opening. The number of cases of COVID-19 are still getting registered throughout the Country. We hope that everyone follows the safety measures and precautions , then it can come to an end. Hence to ensure that people wear masks .

**CHAPTER 1**

INTRODUCTION

A new strain of virus was identified in humans, known as novel coronavirus, which was never previously been identified in humans. Coronaviruses are a wide group of viruses which cause illness that range from basic colds to infections like Middle East Respiratory Syndrome and Severe Acute Respiratory Syndrome (SARS). The first infected patient of coronavirus was found in December 2019. The habit of wearing face masks while stepping out is rising due to the COVID- 19 corona virus epidemic. Before Covid-19, masks were worn by people to protect their health from air pollution. Scientists have concluded that wearing face masks work on decreasing COVID-19 transmission. In 2020, the rapid spread of COVID-19 led the World Health Organization to declare COVID- 19 as a global pandemic. The virus spreads through close contact of humans in crowded/overcrowded places. Cleaning hands repeatedly while meeting others, maintaining a safe distance, wearing a mask, refraining from touching eyes, covering nose and mouth are the main, where wearing a mask is the simplest one. Unfortunately, people are not following these rules properly which is resulting in speeding the spread of this virus. we have a solution which will reduce the number of people affected by COVID. detect the people not wearing mask and informing their authorities. the face  mask detection is a technique to find out whether the person is wearing a mask or not. In medical  applications Deep learning techniques are highly used as it

* The main goal is an attempt to detect face and mask in the browser, instead of Python implementation at back-end. This application is a simple webapp / SPA application which contains JS code only and can send some data to a backend for next processing. But initial face and mask detection is done on the browser side, no Python backend implementation is needed.
* At the current moment, the app works only in the Google Chrome browser.
* In future articles, I will describe more technical details, the implementation of all our investigation results.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_DEPARTMENT OF ECE, MLRITM PAGE | 1

Graphical user interface, application, icon

Description automatically generated

Figure 1 :-Example for facemask detection example

The implementation

There are 2 approaches of how this can be done with browser implementation

1. TensorFlowJS
2. ONNXJS  
    **And we mainly discuss about TensorFlow js**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_DEPARTMENT OF ECE, MLRITM PAGE | 1

**Both runtimes support WASM, WebGL, and CPU backends. But we will compare only WASM and WebGL, because CPU performance is very low and it can't be used in production.**

TensorFlow.js

At the official website, TensorFlow.js proposes some pre-trained and ready to be used models which includes the appropriate JS postprocessing. For real-time face detection, it is specified in the BlazeFace model, for which an online demo is available. The appropriate links to run the app with different models can be found below:

WASM (face detection image size: 160x120; mask detection image size: 64x64px With the help of TensorFlow, developers can create dataflow graphs which are structures that show how data passes through the graph, or a series of nodes. Think of each node as a mathematical operation and each edge representing a multidimensional data array or a tensor. This can beeasily implemented in python where these nodes and tensors act as objects. However, the mathematical operations are performed in C++ binaries which shows an optimal performance. Python takes care of directing the traffic and combines them to work together as a unit. TensorFlow can be run on multiple platforms such as in a cloud, a local machine, CPUs or GPUs, iOS, and Android devices. It can also be run on Google’s custom TensorFlow Processing Unit (TPUs). The trained models can be run on any system for predicting results. TensorFlow 2.0 which was released in October 2019 made many significant changes from user feedback. It works more efficiently and is more convenient with simple Keras API for training models and better performance. With the help of TensorFlow Lite, it is possible to train models on a wide variety of devices.

**React**

It is a free and open-source front-end JavaScript libraryfor building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality

The Greeter function is a React component that accepts a property greeting. The variable App is an instance of the Greeter component where the greeting property is set to 'Hello, World!'. The ReactDOM.render method then renders our Greeter component inside the DOM element with id myReactApp. When displayed in a web browser the result will be

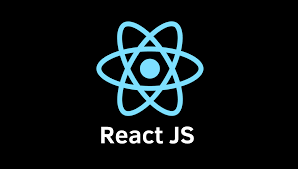


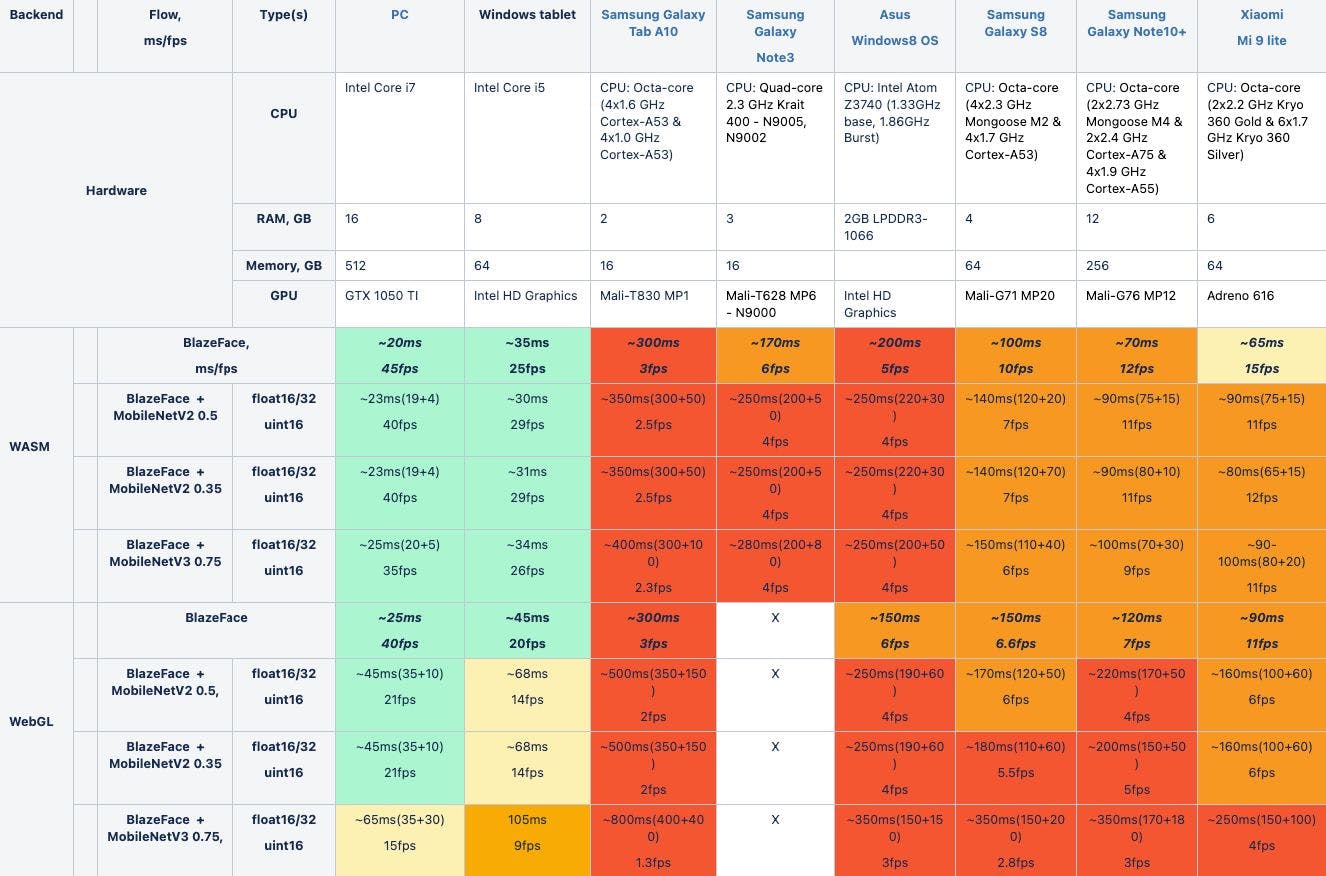
Figure 2 : React

**Proposed workflow**

We decided to build a very simple and basic Convolutional Neural Network (CNN) model using  TensorFlow with code js to detect if you are wearing a face mask to protect  yourself. All the aspects of our work are described below.

**Deep learning architecture**

The deep learning architecture learns various important nonlinear features from the given  samples. Then, this learned architecture is used to predict previously unseen samples.

 Table 1 : Color scheme: < 6 fps red, 7-12 fps orange, 13-18 fps yellow, 19+ fps gree

We excluded timing metrics from the application start. It is obvious that during the application startup and first runs of the model it will consume more resources/time. When the app is in a warm mode it is worth getting performance metrics only. Warm mode in our case it is just let the app do work 5 - 10 sec and then get performance metrics.

So, there are two model types:

* Front: Input size 128 x 128px, faster but lower accuracy.
* Back: Input size 256 x 256px, higher accuracy but slower.

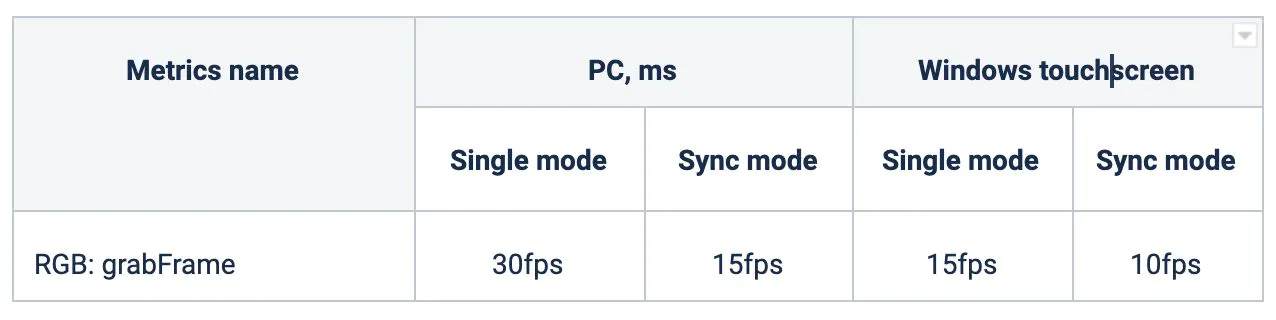


Table 2 : Single mode it is max performance of getting frame. Sync mode it's how often we can get frame with face detection.

This means that the preparation of a detector model with three classes (clear face, face with mask, background) can be a time consuming task.

Images Size

The original image can be of any size according to the camera settings and business needs. But when we process the frames for face and mask detection the original frame is resized to the appropriate size depending on the model.

Image which is used by Blaze Face for face detection is specified with the size 128 x 128px. The original frame is resized to this size considering its proportion. Image which is used by mask detection is specified with the size 64 x 64px.

We chose the minimal resolutions for both images taking into account performance requirements and results. Such minimal images demonstrated the best performance results on PC and mobile devices. We use 64 x 64px for thumbnails to detect the mask because size 32 x 32px is not enough for mask detection with sufficient accuracy.

In this step, we split our data into the training set which will contain the images on which the CNN model will be trained and the test set with the images on which our model will be tested.

In this, we take split\_size =0.8, which means that 80% of the total images will go to the training set and the remaining 20% of the images will go to the test set.

In this step, we split our data into the training set which will contain the images on which the CNN model will be trained and the test set with the images on which our model will be tested.

In this, we take split\_size =0.8, which means that 80% of the total images will go to the training set and the remaining 20% of the images will go to the test set

After building our model, let us create the ‘train\_generator’ and ‘validation\_generator’ to fit them to our model in the next step. We see that there are a total of 2200 images in the training set and 551 images in the test set.

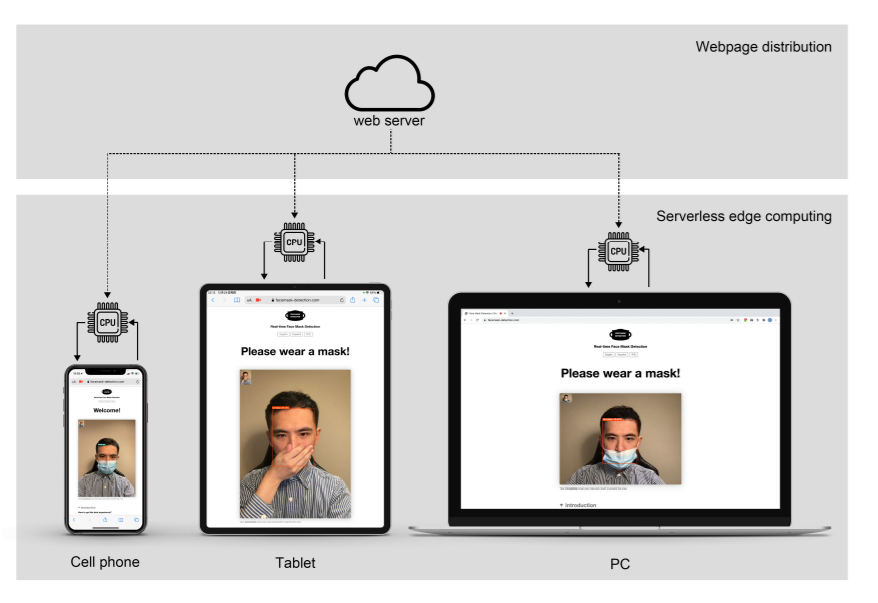
Cascade Classifiers for detecting the features of the face.

Figure 3: Mask detection in various devices

Additional Resources

1. WASM JS back-end
2. OpenCV JS
3. Tensorflow.js

If it is used by web workers and OpenCV.js is used in the worker only it will significantly reduce the size of the main app with code JS

Dataset

Collection To train our deep learning architecture, we collected images. The architecture of the learning technique highly depends on CNN. Data from source is collected for training and testing the model. Dataset contains images of faces only. It consists of about 1,315 images in which 658 images containing people with face masks and 657 images containing people without face masks. For training purposes, 80% images of each class are used and the rest of the images are utilized for testing purposes.

Possible approaches to run neural network models in browser:

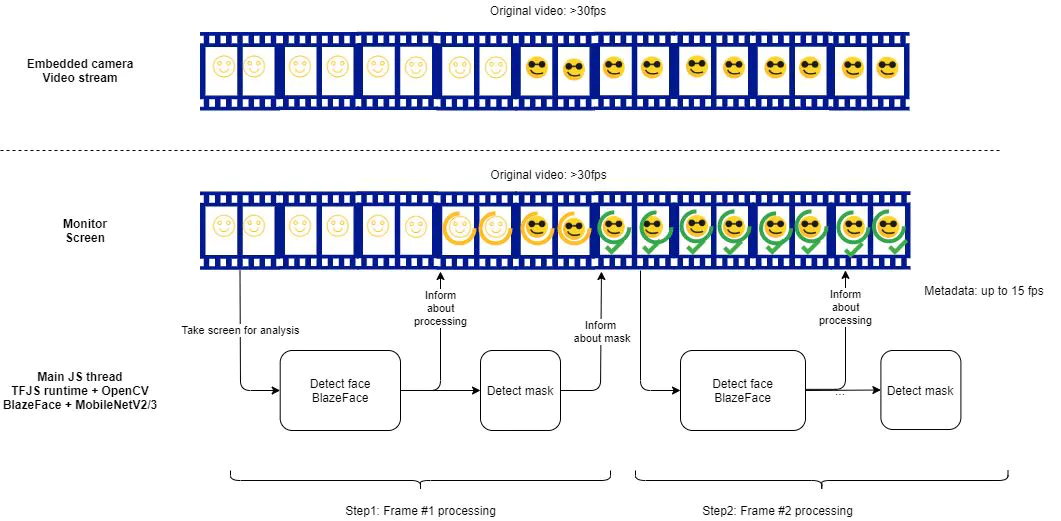


Figure 4: Run neural network

It is a default implementation for browsers. We run both models for face and mask detections in one thread. A crucial point is to provide a good performance for both models to be run with any issues in one single thread. It is the current approach which was used to take the performance metrics above.

This approach has some limitations. If we would like to add additional models to be run in the pipeline they will be run in the same thread in async way but in sequential flow. It will decrease total performance metrics regarding frames processing.

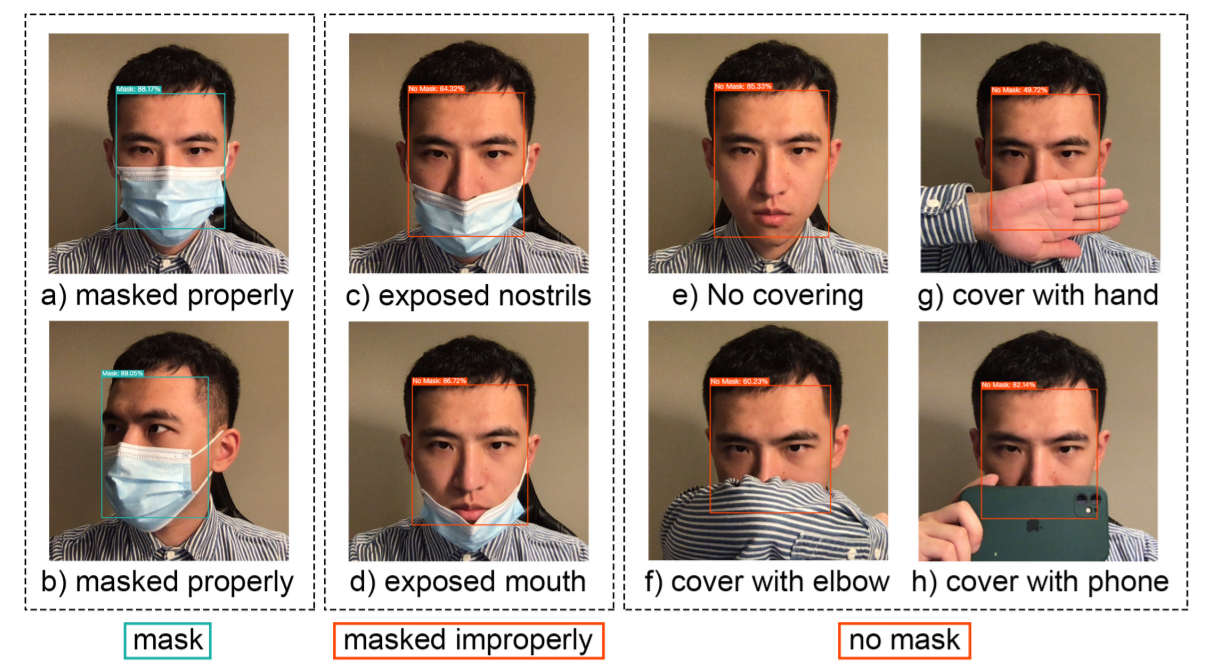


Figure 5: Mapping of mask

In this project a convolutional neural network is used to detect face masks. The neural network takes in the input image as a frame from the video, processes it and classifies it under two categories: mask and no mask. The model was trained using 3800 images, 1900 images each for “with mask” and “without mask” categories.

Web Worker usage to run models in different context and have parallelism in the browser:

It is used in the main JS thread to run model for face detection, but mask detection is running in a separate thread via web worker. With such implementation we can separate both models running and introduce parallelism for processing in the browser. It will have a positive influence on general UX feeling with the application. Web workers will load TFJS and OpenCV libraries, main JS thread - TF.JS only.

It means that the main app will start much faster and by doing it we can significantly reduce TTI time in the browser. Face detection will start more often, it will increase the FPS of the face detection process. As a result mask detection process will be run more often as well and FPS of this process will be increased as well

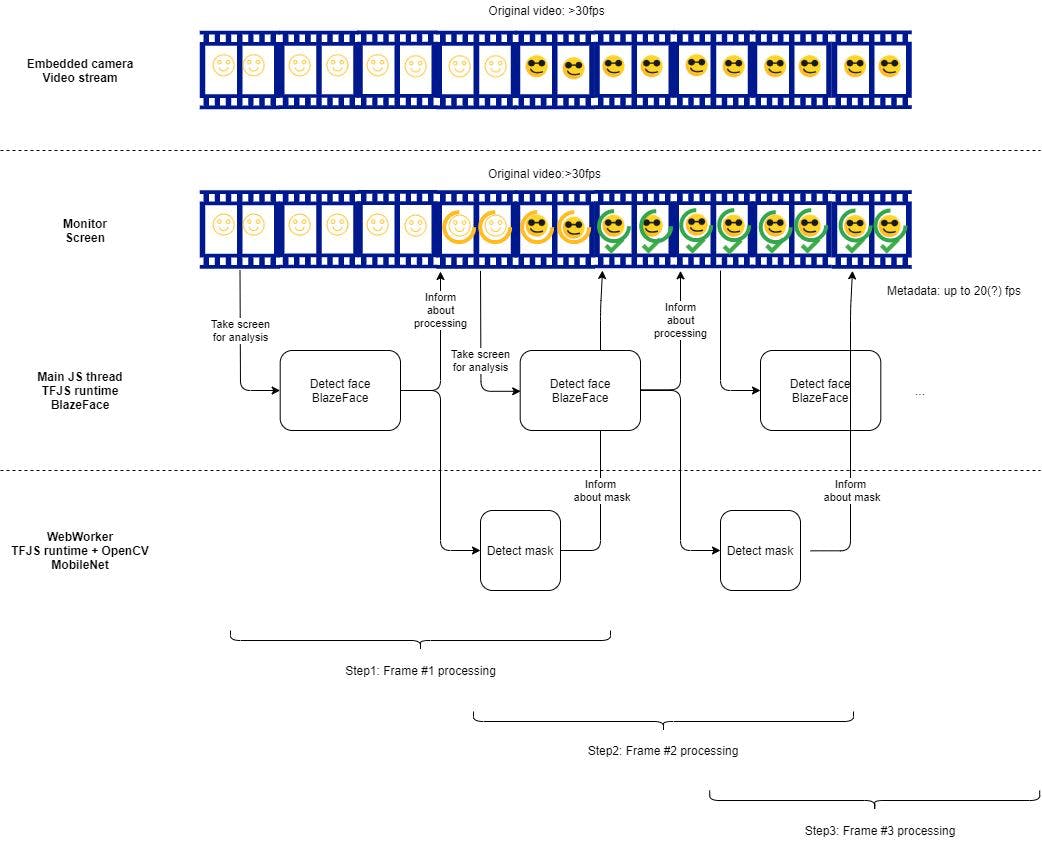


Figure 6: Web Worker usage to run models in different context and have parallelism in the browser

How to Increase Accuracy, Decrease False Positives, Run Access Control API for Stable Faces Only

We need to have more "context" to take the appropriate decision to inform to face detection and removing mask . Such context is the prev processed frames and their state (is face or not, is face or not, is mask or not). We implemented a "flying" segment to manage such additional context. The length of segment is configurable and depends on the current FPS, but in any case it should not be more than 200-300ms to not introduce visible delays.

Performance Results: Getting Frames

We can get frames via the appropriate HTML API features. But this process frames consumes time as well. So we need to understand how much time we will spend on such activities. There are the appropriate timing metrics below.

Our goal is to detect faces as fast as possible so we should get each frame of the live stream and process. For this issue, we can use requestAnimationFrame which is called every 16.6ms or each frame.

Using grabFrame() method of the ImageCapture allows us to take a snapshot of the live video in a MediaStreamTrack and return a promise that resolves with an ImageBitmap containing the snapshot.

All governments around the world are struggling against COVID-19, which causes serious health crises. Therefore, the use of face masks regulatory can slow dow

the high spread of this virus. In this context, Dey et al. proposed in [6] a deep learning-based model for detecting face mask. This model named “MobileNet Mask” is multiphase. A pretrained model of the ResNet-10 architecture is utilized to find faces in video stream. Also, numerous steps are used such as charging the classifier (MobileNet), building the FC layer, and testing phase. All the experimental cases are supervised on Google Colab that runs in the cloud and is provided with over 12 GB of RAM. Different performance metrics (accuracy, F1-score, precision, and recall) are used to judge the performance of the proposed model.

Two distinct face mask datasets are used to train and test the model. The first dataset, named IDS1, consists of 3835 images, divided into two classes: 1916 images of faces with masks and 1919 images without masks. Kaggle dataset, RMFD, and Bing Search API are the source of the typical images of this dataset. The second dataset, named IDS2, consists of 1376 images, divided into two classes: 690 images of faces with masks and 686 images without masks. The sample images of this dataset are gathered from the SMFD.

For all experiments, 80% of the datasets are dedicated for training and 20% for testing. When testing the model, it achieved an accuracy of 93% in IDS1, but almost 100% in IDS2. Comparing the results of their model with those of state-of-art models available in the literature, Dey et al. found that the accuracy

theirs is higher. The main advantage is that their model can be implemented on light-weight embedded computing devices.

**Chapter 2:-**

**Literature Survey**

There are several approaches are used for facial masks detection. For instance, used electromagnetic and radiometry techniques for facial masks detection. employed deep neural networks (ANN) using machine learning techniques in Facial Masks detection. Also comparison was made between ELM ANN and BP ANN based on performance measurements . Neural Networks are used to exacted information from ultrasound to classify the abnormal lesions. An island based model for classification of face mask and distinguishing between various classes of face feature detection using artificial neural network. That artificial neural network to detect the abnormality masks lesions based on edge. characteristics, shape and darkness of a lesion. Ultrasound imaging system in order to reduce the dependency of the operator. Linear Discriminant Analysis to classify the informal face mask feature detection using texture and morph metric parameters.

**Chapter 3 :-**

**Code**

Code Testing

# Getting Started with Create React App

This project was bootstrapped with [Create React App](<https://github.com/facebook/create-react-app>)

## Available Scripts

In the project directory, you can run:

### `npm start`

Runs the app in the development mode.\

Open [http://localhost:3000](http://localhost:3000) to view it in the browser.

The page will reload if you make edits.\

You will also see any lint errors in the console.

### `npm test`

Launches the test runner in the interactive watch mode.\

See the section about [running tests](https://facebook.github.io/create-react-app/docs/running-tests) for more information.

### `npm run build`

Builds the app for production to the `build` folder.\

It correctly bundles React in production mode and optimizes the build for the best performance.

The build is minified and the filenames include the hashes.\

Your app is ready to be deployed!

See the section about [deployment](https://facebook.github.io/create-react-app/docs/deployment) for more information.

### `npm run eject`

\*\*Note: this is a one-way operation. Once you `eject`, you can’t go back!\*\*

If you aren’t satisfied with the build tool and configuration choices, you can `eject` at any time. This command will remove the single build dependency from your project.

Instead, it will copy all the configuration files and the transitive dependencies (webpack, Babel, ESLint, etc) right into your project so you have full control over them. All the commands except `eject` will still work, but they will point to the copied scripts so you can tweak them. At this point you’re on your own.

You don’t have to ever use `eject`. The curated feature set is suitable for small and middle deployments, and you shouldn’t feel obligated to use this feature. However we understand that this tool wouldn’t be useful if you couldn’t customize it when you are ready for it.

## Learn More

You can learn more in the [Create React App documentation](https://facebook.github.io/create-react-app/docs/getting-started).

To learn React, check out the [React documentation](<https://reactjs.org/>).

### Code Splitting

This section has moved here: [https://facebook.github.io/create-react-app/docs/code-splitting](<https://facebook.github.io/create-react-app/docs/code-splitting>)

### Analysing the Bundle Size

This section has moved here: [https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size](<https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size>)

### Making a Progressive Web App

This section has moved here: [https://facebook.github.io/create-react-app/docs/making-a-progressive-web-app](<https://facebook.github.io/create-react-app/docs/making-a-progressive-web-app>)

### Advanced Configuration

This section has moved here: [https://facebook.github.io/create-react-app/docs/advanced-

configuration](<https://facebook.github.io/create-react-app/docs/advanced-configuration>)

### Deployment

This section has moved here: [https://facebook.github.io/create-react-app/docs/deployment](<https://facebook.github.io/create-react-app/docs/deployment>)

### `npm run build` fails to minify

This section has moved here: [https://facebook.github.io/create-react-app/docs/troubleshooting#npm-run-build-fails-to-minify

Code :

import React, { useEffect, useRef, useState } from 'react';

import './App.css';

import \* as tfjs from "@tensorflow/tfjs";

import Webcam from 'react-webcam';

import { Button } from '@material-ui/core';

function App() {

  const videoConstraints = {

    width: 260,

    height: 200,

    facingMode: "user"

  };

  const [prediction,setprediction]=useState("");

  const webcamRef=useRef(null);

  const [loading,setloading]=useState(false);

  const clas=['with mask','without mask']

  const [image,setimage]=useState(null);

  const [cmode,setcmode]=useState(true);

  var Model;

  var imgtensor;

  const captureimg = React.useCallback(

    () => {

      const imageSrc = webcamRef.current.getScreenshot();

      setimage(imageSrc);

      setcmode(false);

    },

    [webcamRef]

  );

  const stoi=(a)=>{

    var index=parseInt(a[0]);

      setprediction(clas[index]);

      setloading(false);

      return clas[index];

  }

  const pred=async()=>{

    if(image==null)

    {

      setprediction("capture the image");

      return;

    }

    setloading(true);

    Model=await tfjs.loadLayersModel("/mymodeljs/model.json");

    var img=new Image();

    img.crossOrigin='anonymous';

    img.src=image;

    img.width=150;

    img.height=150;

    imgtensor=tfjs.browser.fromPixels(img).expandDims(0);

    console.log("predict"+imgtensor);

    //await sleep(7000);

    try{

    const res= await Model.predict(imgtensor);

    const s=res.toString().substring(14,res.toString().length-3).split(", ");

    console.log(s);

    console.log(stoi(s));

    }

    catch(err){

      console.log(imgtensor);

      console.log(err);

    }

  }

  return (

    <div className="App" >

      <pre className="titleborder">

     <h1 className="title">

      Mask Detection</h1></pre>

    {

      cmode?

    <div className="cam"

    style={{

      display:"block"

    }}

    >

    <Webcam

        audio={false}

        height={400}

        ref={webcamRef}

        screenshotFormat="image/jpeg"

        width={460}

        videoConstraints={videoConstraints}

        style={{display:"block",

              marginLeft:"auto",

              marginRight:"auto",

      }}

    />

    <Button

    variant="contained"

    color="primary"

    style={{

      display:"block",

      marginLeft:"auto",

      marginRight:"auto",

      fontFamily:"sans-serif"

    }}

    onClick={(e)=>{e.preventDefault();captureimg();}}

    > <i class="fas fa-camera"></i> capture</Button>

    </div>

    :

    <div className="capturedmode">

    <img className="capturedimg" src={image} width="260" height="280"

    style={{

      display:"block",

      marginLeft:"auto",

      marginRight:"auto"

    }}

    ></img>

    <div className="buttons"

    style={

      {

        marginTop:"2%",

        display:"flex",

        justifyContent:"center",

      }

    }

    >

    <Button

    variant="contained"

    color="primary"

    style={{

        marginRight:"6%",

    }}

    onClick={()=>setcmode(true)}

    ><i class="fas fa-redo"> Retake</i></Button>

    <Button

    variant="contained"

    color="secondary"

    style={{

    }}

    onClick={()=>pred()}

    >Predict</Button>

    </div>

    {

      loading?

      <div>

      <img src={"https://acegif.com/wp-content/uploads/loading-42.gif"}

      style={

        {

          width:"15%",

          height:"10%",

          display:"block",

          marginLeft:"auto",

          marginRight:"auto"

        }

      }

      ></img>

      </div>

      :(prediction=="with mask" )?

      <h2 style={{

        marginTop:"2%",

        color:"darkblue",

        textAlign:"center",

        animation:"shrink 5s linear infinite"

        }}>{prediction}</h2>

      :

      <h2 style={{

        marginTop:"2%",

        color:"darkblue",

        textAlign:"center",

        animation:"shrink 5s linear infinite"

      }}>{prediction}</h2>

    }

    </div>

    }

    </div>

  );

}

export default App;

Chapter 4:-

RESULT

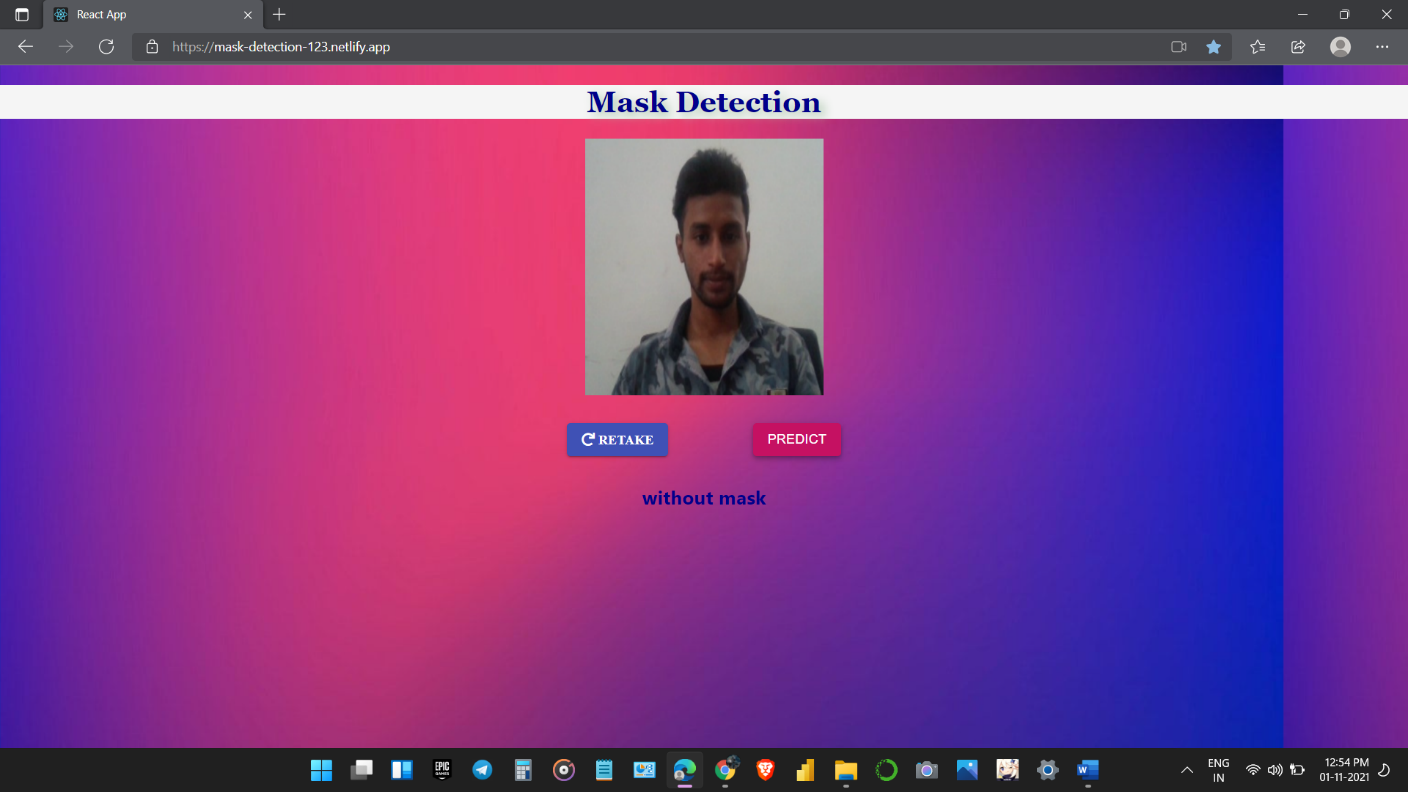


Figure 7 : Outputs1

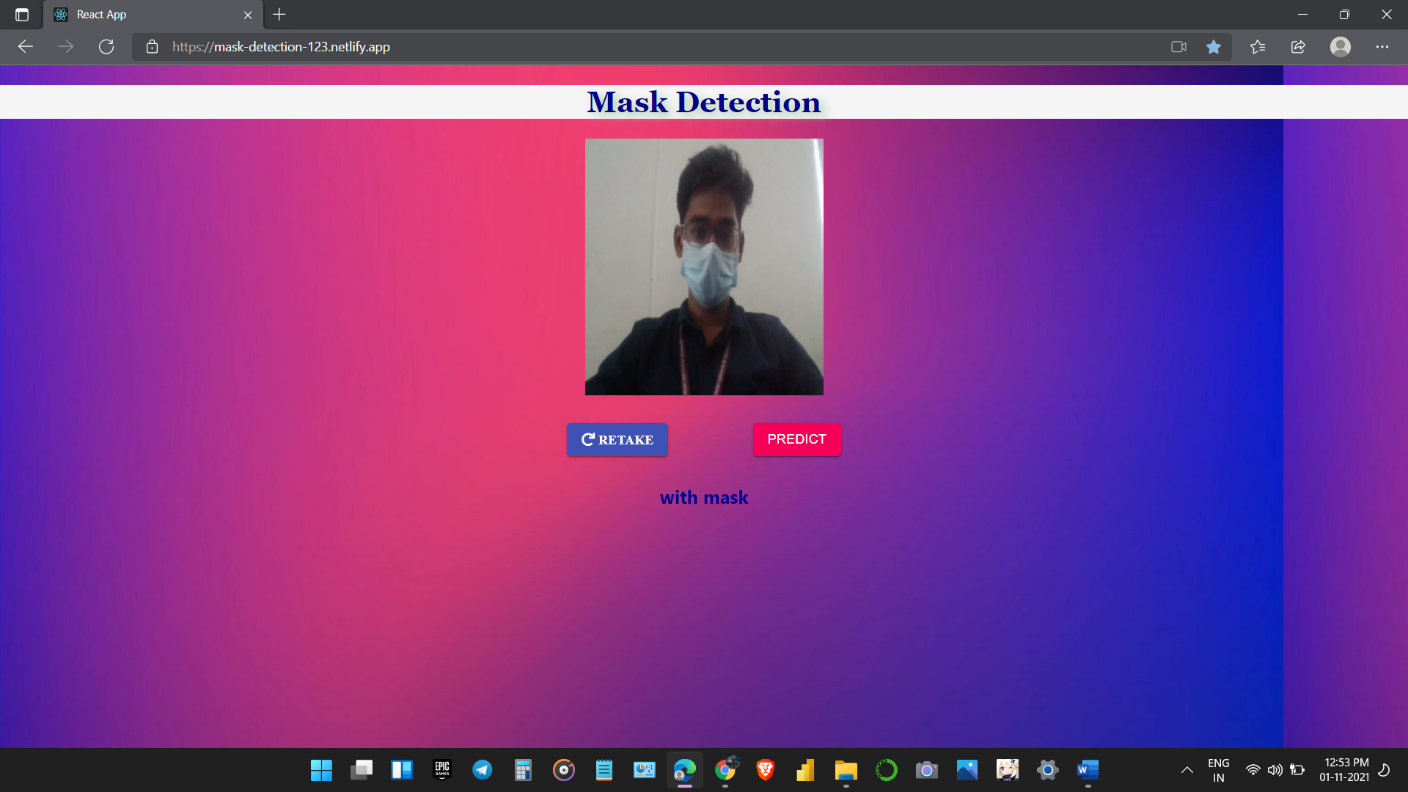


Figure 8 : output 2

***Chapter 5 :***

**Conclusions:**

As stronger device we have as better performance results regarding taking metrics it is demonstrated The proposed system to classify face mask detection using COVID-19 precaution both in images and videos using convolution neural network. Extensive experimentation on the datasets and the performance evaluation of the proposed methods are exhibited. Further, we made a successful attempt to preserve inter and intra class variations of face mask detection using symbolic approach. We studied the different classifiers like Support Vector Machine and Symbolic Classifier. The project is developed as a prototype to monitor temperature measurement and to detect mask for the people. The work is designed to provide a safety system for the people in order to avoid COVID-19. We proposed continuous monitoring of the people conditions and store the people’s data in the server using the Deep learning concept. In order to investigate the performance, the proposed method an extensive experimentation is conducted on 50 various Image datasets. We conducted experimentation under varying number of training and testing percentage for 10 random trails. From the results we could observe that, the results obtained for symbolic approach is better than the conventional approach.

LIMITATIONS AND FURTHER WORKS

The developed system can detect the live video streams but does not keep a record. Unlike the CCTV camera footage the admin can not rewind, play or pause it. As whenever a strict system is imposed people always try to break it. Hence when a person is detected with no mask, the head of the organization can be notified via mail that so and so person entered without mask. The proposed system can be integrated with databases of respective organizations to keep a record of the person who entered without mask. With more complex functions a screenshot of the person’s face can also be attached to keep it as a proof.

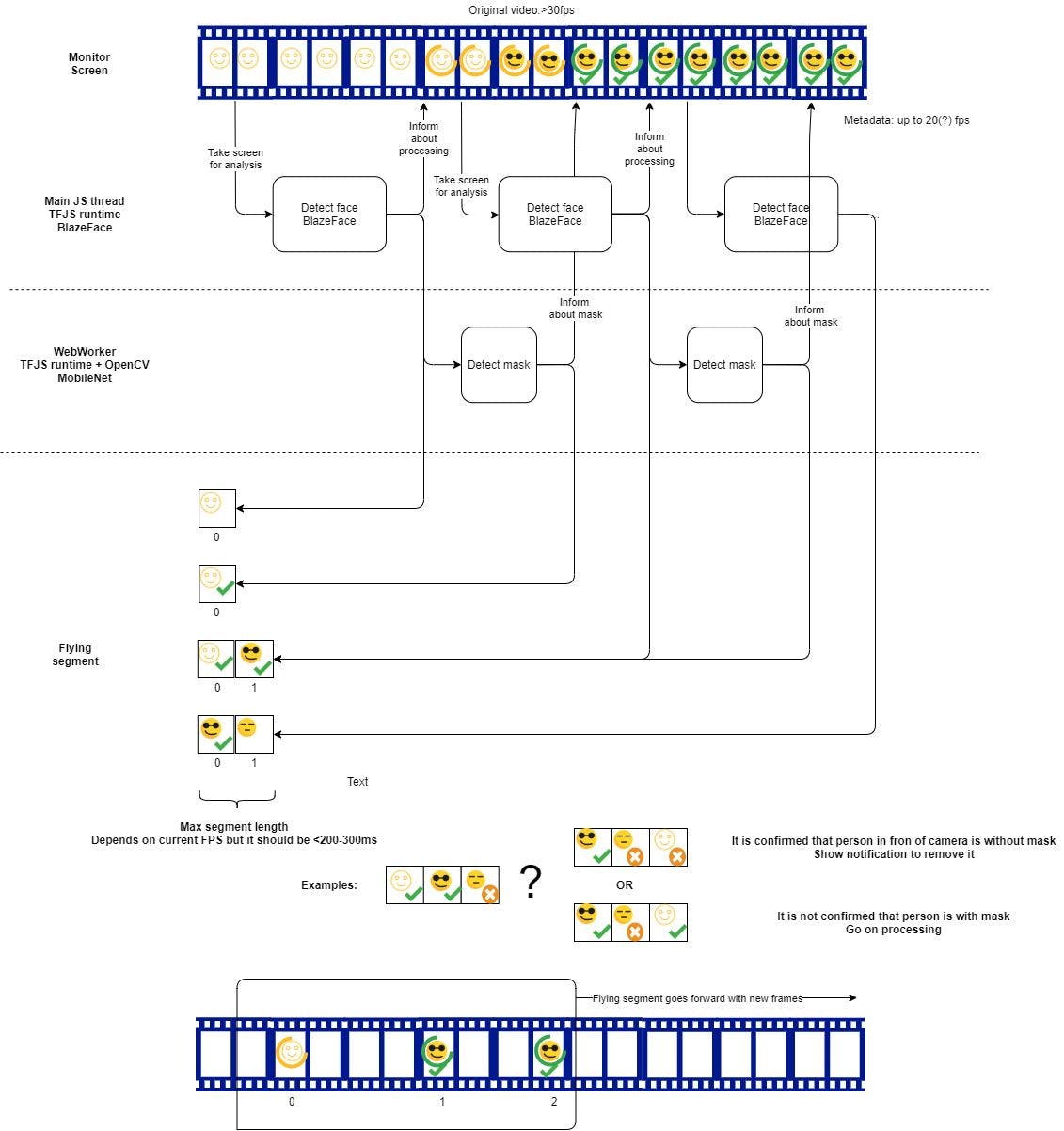


Figure 9: appropriate scheme which describes the idea

**Future Work**

The future work is as follows: -

* Perform the classification efficiently
* Using multiple datasets which could attain the optimum prediction.
* Database creation and addition of people in that database who are frequent defaulters
* Improve the overall time complexity of the entire workflow.
* Integrate the Person identification model and face mask detection model into a single detection algorithm.

For PC we could get the following metrics:

* Face detection only: >30fps
* Face detection + mask detection: up yo 45fps

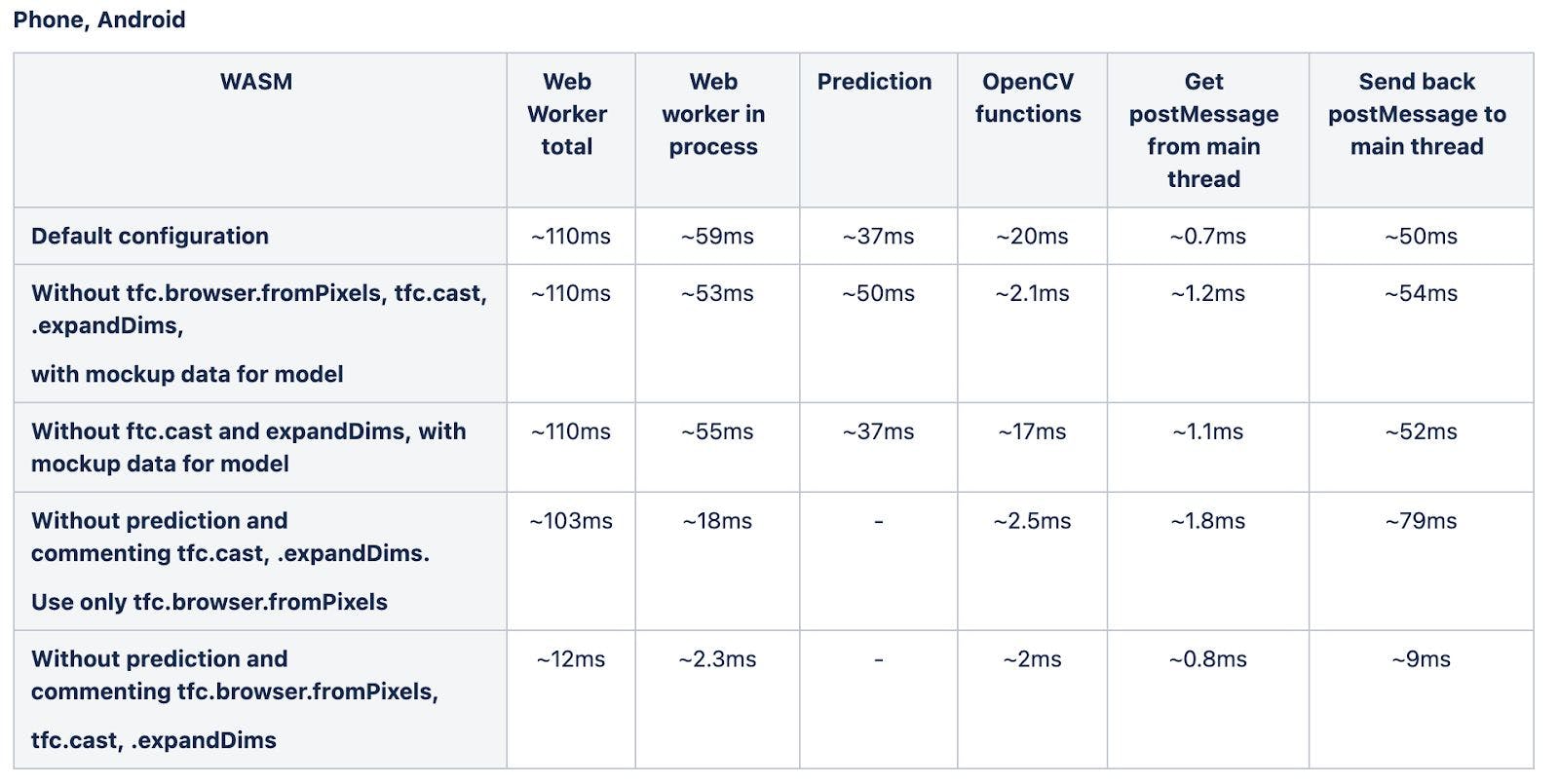


Table 3 : Here are some measurements for WW processing time on different devices:

For mobile devices we could get the following metrics:

* Face detection only: from 2.5fps to 12-15fps depends on mobile devices
* Face detection + mask detection: from 2 to 12fps depends on mobile devices

1. Pay attention that video is always real-time and its performance depends on device itself, but it always will be from 30fps.
2. For mask detection model for the most cases the best results are demonstrated by MobileNetV2 0.35 model, types don't influence the performance metrics with visible effect.
3. Size of the mask detection model depends on types. Since types don't influence performance metrics it is recommended to use uint16 or float16 models to have less model size in the browser and as a result faster TTI.
4. WASM runtime demonstrates better results in comparison with WebGL for BlazeFace model. It is correlated with the official TensorFlowJS documentation regarding performance of small models (<3Mb)
5. For most models, the WebGL backend will still outperform the WASM backend, however WASM can be faster for ultra-lite models (less than 3MB and 60M multiply-adds). In this scenario, the benefits of GPU parallelization are outweighed by the fixed overhead costs of executing WebGL shaders.
6. TTI time is always better for WASM models in comparison with webgl with the same models configuration.
7. More info is here. It is still an experimental feature which is not delivered with the runtime by default. Also after the release it will be available in the runtime by default or via additional configuration options.
8. On the client side the expected size of the app is ~3.5Mb of encoded JS with all dependencies, 466Kb of BlazeFace model, from 1.1Mb to 5.6Mb of mask detection model. The expected TTI time for the application is >10sec for mobile devices in cold mode; in warm mode - ~5sec.
9. With web workers usage OpenCV.js can be loaded into web workers only, it significantly reduce TTI for the main app.
10. During our testing we noticed that the devices were very hot. Since it is expected 24/7 availability of the solution this fact can have a big influence on the final decision which devices to use. You need to take this information into account.
11. It is also was noticed that it is better if devices are fully charged and they are connected to the charger during working. In this case the performance of devices according to our observations is better. Charging from USB doesn't allow to support battery on the same level in a long perspective, it should be used for electric charging. This information should be taken into account.
12. In our view even 4-5 fps are enough for the current solution to provide good UX user perception. But it is important to not show bounding boxes or landmarks on the screen, but operate the whole video/screen size

**References :**

1. World Health Organization et al. Coronavirus disease 2019 (covid-19): situation report, 96. 2020. - Google Search. (n.d.). https://www.who.int/docs/default-source/coronaviruse/situation-reports/20200816-covid-19-sitrep-209.pdf?sfvrsn=5dde1ca2\_2.

2. Social distancing, surveillance, and stronger health systems as keys to controlling COVID-19 Pandemic, PAHO Director says - PAHO/WHO | Pan American Health Organization. (n.d.). https://www.paho.org/en/news/2-6-2020-social-distancing-surveillance-and-stronger-health-systems-keys-controlling-covid-19.

3. Garcia Godoy L.R. Facial protection for healthcare workers during pandemics: a scoping review, BMJ. Glob. Heal. 2020;5(5) doi: 10.1136/bmjgh-2020-002553. [PMC free article] [PubMed] [CrossRef] [Google Scholar]

4. Eikenberry S.E. To mask or not to mask: Modeling the potential for face mask use by the general public to curtail the COVID-19 pandemic. Infect. Dis. Model. 2020;5:293–308. doi: 10.1016/j.idm.2020.04.001. [PMC free article] [PubMed] [CrossRef] [Google Scholar]

5. Wearing surgical masks in public could help slow COVID-19 pandemic’s advance: Masks may limit the spread diseases including influenza, rhinoviruses and coronaviruses -- ScienceDaily. (n.d.). https://www.sciencedaily.com/releases/2020/04/200403132345.htm.

6. Nanni L., Ghidoni S., Brahnam S. Handcrafted vs. non-handcrafted features for computer vision classification. Pattern Recogn. 2017;71:158–172. doi: 10.1016/j.patcog.2017.05.025. [CrossRef] [Google Scholar]

7. Y. Jia et al., Caffe: Convolutional architecture for fast feature embedding, in: MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia, 2014, doi: 10.1145/2647868.2654889.

8. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun, OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, 2014.

9. Erhan D., Szegedy C., Toshev A., Anguelov D. Proceedings of the IEEE conference on computer vision and pattern recognition. 2014. Scalable Object Detection using Deep Neural Networks; pp. 2147–2154. [CrossRef] [Google Scholar]

10. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, vol. 2016-Decem, pp. 779–788, doi: 10.1109/CVPR.2016.91.

for Small Object Detection. J. Electr. Comput. Eng. 2020;2020 doi: 10.1155/2020/3189691. [CrossRef] [Google Scholar]