Extant: The COVID-19 Employee Attendance System

Project report submitted in partial fulfilment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By

Ananya Shukla (171038)
Sushant Singh (171041)

UNDER THE GUIDANCE OF

Mr. Pradeep Garg

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

December 2020

TABLE OF CONTENTS

DECLARATION

We hereby declare that the work reported in the B.Tech Project Report entitled **“Extant: The COVID-19 Employee Attendance System”** submitted at **Jaypee University of Information Technology, Waknaghat, India** is an authentic record of our work carried out under the supervision of **Mr. Pradeep Garg.** We have not submitted this work elsewhere for any other degree or diploma.

Ananya	[illegible]
Ananya Shukla	Sushant Singh
(171038)	(171041)

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

[illegible]

Mr. Pradeep Garg

Department of Electronics and Communication

Dr. Rajiv Kumar

Head of Department

Date: 3rd December 2020

Acknowledgement

I'd want to take this opportunity to thank and express my heartfelt gratitude to Mr. Pradeep Garg, my mentor and project guide, for his unwavering support and invaluable direction, without which we would not have been able to get to this point in our final year project.

I am grateful for the excellent support provided by Prof. Dr. Rajiv Kumar, Head of Electronics & Communication Department. We would also like to thank all professors and other supporting members for their generous support in various ways to complete this work. The Department of Electronics and Communication. I am also grateful for your valuable support to all my faculty members in their respective fields which helped me achieve my project at this phase.

Date: 3rd December 2020

LIST OF FIGURES

ABSTRACT

Automation is the future of technology. Tedious work will soon be taken over by intelligent machines, which will be able to do the same task with precision and in lesser time.

Our project aims at improving the lives of business owners in the current pandemic situation by using current technologies which help in keeping their employees safe and healthy. We hope to provide an all-in-one solution for keeping the business premises safe from the COVID-19 virus.

Our project uses state of the art Neural Networks to distinguish whether an employee is wearing a mask or not. This information is then fed into a database, which can be checked by the employer, and can potentially raise an alert when an irregular entry happens.

Using a robust Convolutional Neural Network to determine if an employee is following the proper safety precautions is an efficient and feasible method as most computing machines are able to do mathematical calculations involving Neural Networks.

We plan to include facial detection as well as a User Interface so that it can become a standalone program with no dependencies on ID cards or Roll Numbers.

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

In the current times, nothing is predictable. The COVID-19 has impacted all of our lives in a major way, and it has dealt a huge blow to small to medium scale businesses.

In a country with a population of more than 133 crore people, it is not an easy task to manage a pandemic. Realistically, even with harsh measures like lockdowns and quarantines, the virus has spread rapidly throughout the whole country, and as the 2nd wave is on the brink, small businesses need a lot of help, especially in safety measures to protect their employees and their customers from COVID-19.

It is both potentially dangerous to personally check if everyone around you is following the correct safety precautions, not to mention the monotonous and stressful nature of the task.

To counter this problem, we decided to create a solution that does this work for us, in a systemized and tabular manner.

1.2 Objectives

Our objective with Extant is to take attendance systems to the next step and make them completely independent.

We aim to help local and small to medium scale business get back up on their feet without worrying about the pandemic affecting their place of business. The state-of-the-art Mask Detection algorithm ensures that no employee or customer walks into the building without a mask, eliminating the need for a person to constantly keep an eye out for defaulters.

The web interface dashboard can also serve as a performance indicator for the employees, which can contain extra information, for example, the clock in-clock out time can be used to calculate how many hours an employee worked, which can help in major decisions.

1.3 Methodology

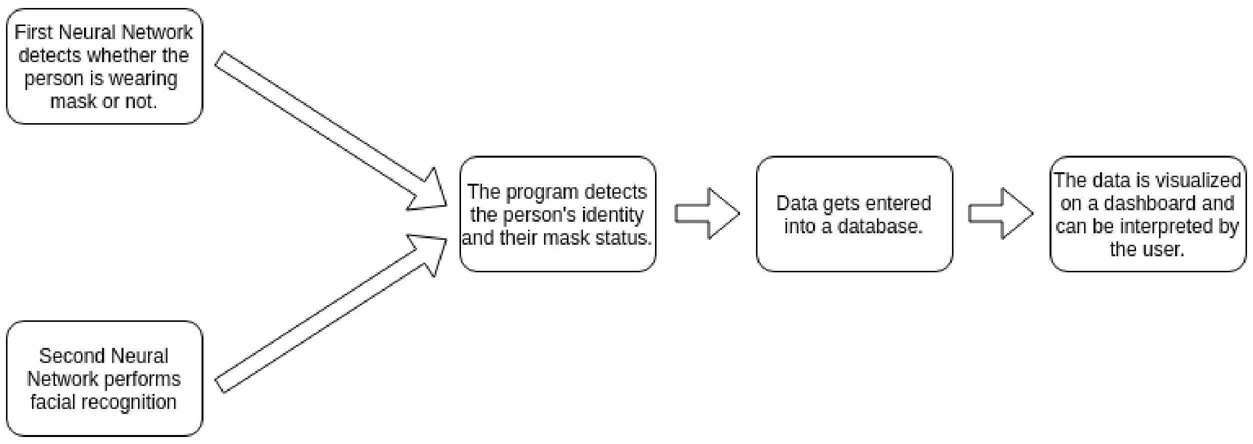

First Neural Network detects whether the person is wearing mask or not.
Second Neural Network performs facial recognition
The program detects the person's identity and their mask status.
Data gets entered into a database.
The data is visualized on a dashboard and can be interpreted by the user.

Figure 1.1: Flowchart of the project

CHAPTER 2

THE COVID-19 PANDEMIC

2.1 A Brief Summary

2.1.1 The Origin

A new strain of coronavirus, SARS-CoV-2, was initially found in December 2019 in Wuhan, a city in China's Hubei Province with a population of 11 million people, following an outbreak of pneumonia with no apparent reason. The virus has already spread to over 200 countries and territories around the world, prompting the World Health Organization (WHO) to declare it a pandemic on March 11, 2020.

It is a member of the coronaviridae family of single-stranded RNA viruses, a widespread type of virus that affects mammals, birds, and reptiles.

Where the virus has originated from is still unknown. The virus was initially understood to have originated in the Wuhan food market and eventually transmitted from animals to humans. The most common theory is that it was transmitted from a bat to a human.

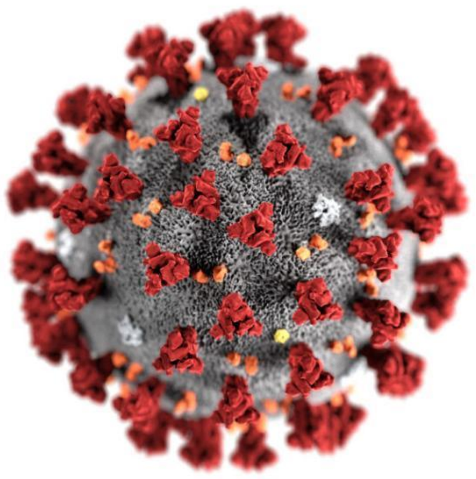


Figure 2.1: An image of the Novel Corona Virus SARS-CoV-2

2.1.2 The Symptoms

In humans, it causes small diseases similar to the common cold, and it accounts for 10-27 percent of upper respiratory tract infections in adults. Coronaviruses can cause gastrointestinal and neurologic illnesses, but more severe infections are rare. The incubation period for coronavirus varies, but it usually lasts two weeks.

2.1.3 The Spread

The numbers of reported diagnoses have shown the person-to-person dissemination of SARS-CoV-2 exists, particularly in healthcare professionals. The preliminary number of reproductions (i.e. the total number of cases caused by a single case over the duration of its infectious period) is currently expected to be between 1.4 and 2.5, indicating that between 1.4 and 2.5 individuals may be infected by each infected individual.

MERS and SARS are spread through respiratory droplets, which are created when an infected person sneezes or coughs, in the same way as other common respiratory tract illnesses are. Steps to safeguard against infection work with the current assumption that SARS-CoV-2 is transmitted in the same way.

2.2 Technological Advances to Battle COIVD-19

2.2.1 3D Printing Supplies

People have been producing all sorts of useful things since 3D printers became mainstream for hobbyists. 3D printers are proving particularly effective in the aftermath of the COVID-19 pandemic.

A Canadian boy scout named Quinn Callander learned that medical practitioners were experimenting for ways to alleviate ear pressure from wearing masks every day at a nearby hospital. He wanted to make an ear guard that sits behind the head and retains the strips of elastic that will typically go around the ears of a human.

The ear guard is a plain, flat plastic piece with four notches on each side, enabling doctors and nurses to change their mask's tightness and bypass their ears.

For physicians and nurses on the front lines, 3D printing firms are now stepping in to render PPE. Fuel, Prusa Science, and Formlabs 3D Systems, 3D printer manufacturers are manufacturing face shields at a rapid pace so far. Around 7,500 masks have already been produced and production is expected to increase dramatically.

To help out as far as possible, high school students even make face shields and even masks using 3D printers.

Although 3D printed face shields are legally not FDA-approved, these homemade face shields and masks are not recommended against being used. However they are advising individuals that 3D printed face shields do not have the same degree of protection for fluid barrier or air filtration that FDA-approved PPE offers.

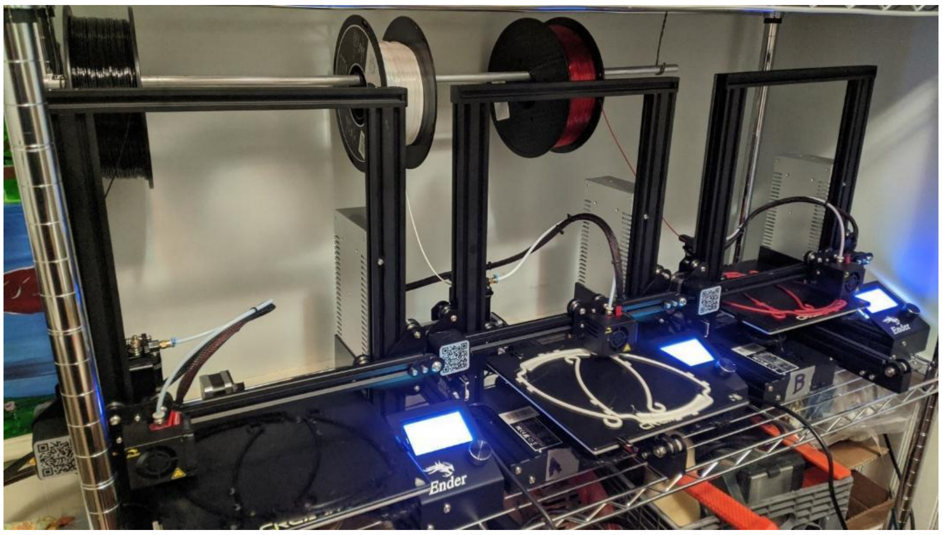


Figure 2.2: A 3D Printer Employed to print PPEs

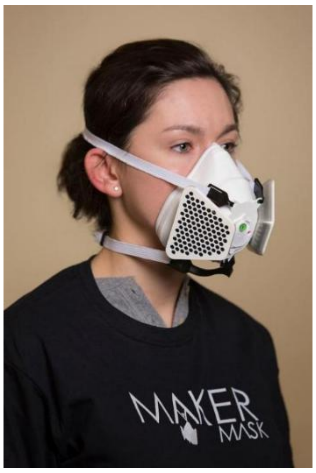


Figure 2.3: A 3D Printed PPE Mask

2.2.2 Google and DeepMind

In two big ways, Google is using technologies to battle COVID-19. First the organisation is partnering with the government of the United States to develop an educational platform on COVID-19 to host resources. The aim is to build a single place where without having to sift through confusion, people can find all the right stuff.

The firm also contributed its DeepMind software to its AlphaFold framework as part of Google's battle against COVID-19. Both systems are part of Google's artificial intelligence solutions, although the inclusion of DeepMind is intended to anticipate the SARS-CoV-2 virus-related protein structures that cause COVID-19 in order to build successful therapies.

An Alphabet-owned business called Verily is creating a wear patch for COVID-19 patients that senses a fever and connects with a phone app to capture an early COVID-19 or flu diagnosis.

Inspired by Google and our desire to help the world as an engineer, we decided to use Deep Learning to help out people in the pandemic in ways that keep them safe.

CHAPTER 3
ARTIFICIAL NEURAL NETWORKS

3.1 Introduction

Human minds comprehend the meaning of real-world events in a way that algorithms cannot. Neural networks were first developed in the 1950s to overcome this challenge. An artificial neural network is a computer programme that attempts to recreate the neuron network that makes up a human brain so that it can learn information and make decisions in a human-like manner. ANNs are created by programming conventional computers to act like interconnected brain cells.

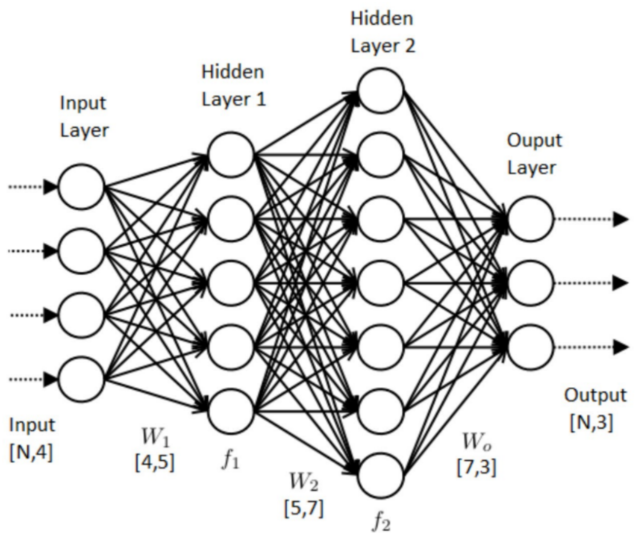


Figure 3.1: Pictorial Representation of an Artificial Neural Network.

3.2 The ‘Learning’ Process

To make sense of the data it is served, artificial neural networks use multiple levels of statistical computation. An artificial neural network usually has hundreds to millions of artificial neurons organised in a sequence of layers, called units. From the outside world, the input layer absorbs different types of information. This is the information that the network is attempting to comprehend or think about. One or more hidden units travel through the results from the input unit. The secret unit's job is to convert the input into something that the output unit can understand.

From one layer to another the bulk of neural networks are entirely connected. These relations are weighted; the higher the amount, equivalent to a human brain, the greater the impact one unit has on another. The network learns more about the data as the data goes into each unit. On the other side of the network are the output units, which are where the network responds to the data it has been given and analysed.

Since computer scientists tried the original artificial neural networks first, the human brain has been discovered by computer neuroscientists. One of the points they found is that it is the task of different sections of the brain to find multiple aspects of knowing and this is hierarchically arranged. Feedback therefore enters the brain, and every neuronal level provides information and the data are transferred to the next, higher level. This is exactly the process that ANNs try to reproduce.

They should receive an enormous amount of material, called a training package, to remind ANNs. In order to continue understanding the network, the instructions would have thousands of photographs on the dog as you are trying to show an ANN how to distinguish cat from dog. If the large volume of information is conditioned, it will try to identify potential data around the different units based on what it feels it sees (or hearing, depending on the data set). The computer's performance is compared with what should be observed during the training process in human terms.

3.3 Forward and Backward Propagation

Every complex structure can be simplistically absorbed or at least divided into its core abstract components. Uncertainty emerges through the aggregation of many basic layers. This article seeks to clarify how neural networks operate in the simplest abstraction. We'll try to distil the machine learning process in NN down to its most fundamental components. In opposition to other posts that describe neural networks and concentrate exclusively on high level abstract principles, we would like to use the smallest possible number of mathematical equations and codes.

The learning process takes the inputs and desired outputs and adjusts its internal state suitably so that the measured output approaches the required outcome as closely as feasible. The forecasting method requires an input and generates the best feasible result based on previous "training experience" with the internal state. That's why machine learning is also referred to as model fitting.

Following are the steps involved in building an Artificial Neural Network.

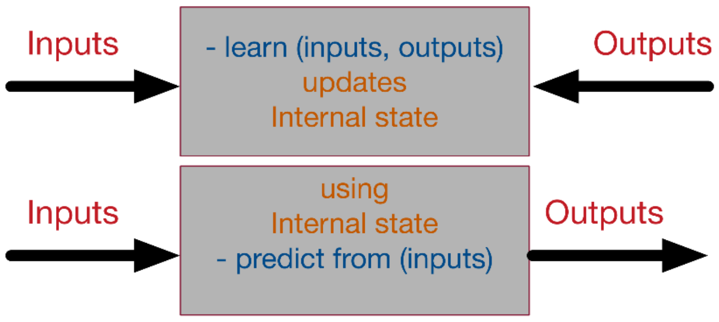


Figure 3.2: Block Diagram of Forward and Backward Propagation

3.3.1 Model Initialisation

The first level of learning is the original hypothesis, starting from everywhere. Neural networks, like in genetic algorithms and the theory of development, can start from anywhere. It is thus common practise to randomly initialise the model. The notion is that we can acquire the pseudo-ideal paradigm from any starting point if we are persistent and learn in an iterative fashion.

3.3.2 Forward Propagation

The next natural step upon haphazardly initialising the system is to evaluate its performance. We start with the data we have, send it via the network layer, and then calculate the model's real output in a straightforward manner.

This stage is named forward-propagation because the calculation stream is moving in the usual progressive manner from the input to the neural network to the output.

3.3.3 The Loss Function

At this moment, we have the genuine output of our haphazardly initialised neural network in one hand. We do, however, have a required outcome for the network to learn.

To be able to generalise to every query, we describe what we term a loss function. Essentially, it's a measure of how successfully the NN completes the goal of creating outputs that are as close to the goal values as possible.

Simply put, loss = the simplest loss function is (desired output: real output). This loss function however returns positive values when undermining the network and negative values when overdoing the network (pronouncing > desired output). If we want the loss feature to describe an absolute output mistake irrespective of whether it is overshooting or undershooting, we can define it as:

Loss = (desired — actual) Absolute value.

Nonetheless, a variety of circumstances might result in the same total number of errors: for example, a series of minor errors or a series of significant errors can add up to the same total number of errors.

It is more advantageous to provide a spread with multiple modest errors rather than a few significant ones if we want the forecast to work under all conditions.

The loss function can be defined as the sum of squares of absolute mistakes to allow the NN to converge in such a situation (which is the most famous loss function in NN). In this way, minor errors are considered much less than severe errors!

To conclude, the loss function is an error metric that indicates how much accuracy we lose when we substitute the real output with the actual output produced by our trained neural network model. It is for this reason that it is referred to as the *loss* function.

Types of loss functions are:

1. **Mean Square Error**: It's calculated by squaring the average variance between projections and actual measurements. It doesn't matter which path they take; all that matters is the average degree of mistake. Predictions that are distant from real values, in contrast to less deviating forecasts, are heavily penalised due to squaring. MSE has good mathematical qualities that make measuring gradients easier.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

Figure 3.3: MSE Formula

2. **Mean Absolute Error**: It is calculated the total number of absolute discrepancies between forecasts and actual observations. This, like the MSE, calculates the extent of mistake without taking into account the course of the error. Unlike MSE, MAE uses more sophisticated methods to measure gradients, such as linear programming. MAE is more resistant to outliers because it does not use the square.

$$MAE = \frac{\sum_{i=1}^{n} \mid y_i - \hat{y}_i \mid}{n}$$

Figure 3.4: MAE Formula

3. **Mean Bias Error**: This is much less prominent in the field of machine learning compared to its predecessor. This is the same as MSE, with the only exception that absolute values are not taken by us. Clearly, as positive and negative errors can cancel each other out, there is a need for caution. In reality, although less precise, it could decide if the model has positive bias or negative bias.

$$MBE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n}$$

Figure 3.5: MBE Formula

4. **SVM Loss**: In basic words, the score of the correct category by any safety margin should be higher than the sum of the scores of all incorrect categories (usually one). And hinge failure,

especially for support vector machines, is therefore used for maximum-margin classification. While not differentiated, it is a convex function that makes it simple to work with common convex optimizers used in the field of machine learning.

$$SVMLoss = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)$$

Figure 3.6: SVM Loss Formula

5. **Cross Entropy Loss**: This is the most prevalent situation in which classification issues arise. As the projected likelihood differs from the actual label, cross-entropy loss grows.

6.

$$CrossEntropyLoss = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))$$

Figure 3.7: CE Loss Formula

3.3.4 Differentiation

Obviously, any optimal solution that modifies the inner weights of neural networks can be utilised to minimise the previously mentioned total loss function. These strategies can be used in genetic algorithms, greedy search, or even a simple brute-force search:

In our basic numerical example, we will check from -1000.0 to +1000.0 step 0.001, where W has the smallest number of squares of errors over the dataset, with only one weight parameter to maximise W.

This might work if the model contains very little parameters and the accuracy is not very great. However, we can quickly hit millions-weight models to refine the NN via a range of 600x400 inputs, and the brute force cannot even be imagined because this is pure machine energy waste.

Fortunately for us, there is a useful theory in mathematics called differentiation that can teach us how to maximise the weights. The derivative of the loss function is essentially dealt with. In mathematics, the derivative of a function at a given position expresses the pace or intensity with which that function's values vary at that location.

We should ask ourselves the following question in order to see the influence of the derivative: how much the overall error can change if we change the inner weight of the neural network to a certain small value of δW. δW=0.0001 would be taken into account for the sake of convenience. It should be even smaller in fact.

By calculating the derivative of the loss function we could have guessed this rate. It is much faster and more accurate to calculate the benefit of utilising mathematical derivation.

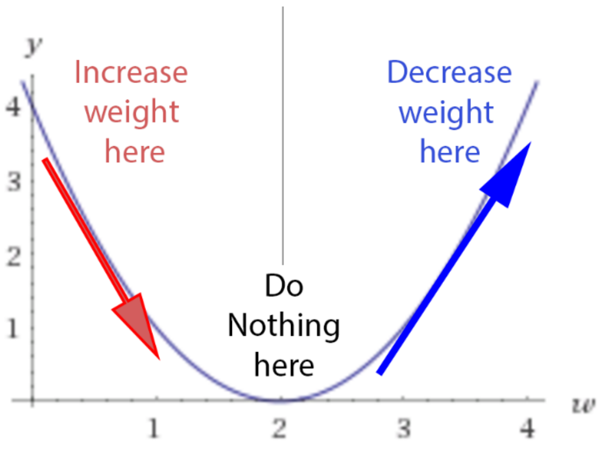


Figure 3.8: The function of derivatives.

Performing a derivative check:

If it's positive, which means that increasing the weights increases the mistake, we can lower the weight. If it's negative, which means that increasing the weights minimises the mistake, we can increase the weight.

We don't do anything if it's 0; we've reached our steady state. In a nutshell, we're creating a technique that mimics gravity. Regardless of where we haphazardly launch the ball on this error function path, there is a force field that forces the ball back to the lowest energy state of ground 0.

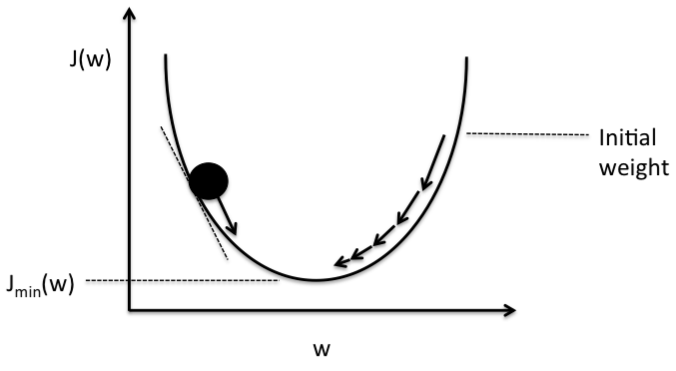


Figure 3.9: Gradient Descent.

3.3.5 Backward Propagation

Within the neural network, we just employed one layer between the inputs and outputs.In some cases, additional layers are required to achieve further changes in neural network control.

Over the entire layers of the network, we can only establish one complicated feature that describes the composition. For instance, if layer 1 does: 3.x to produce a secret output z, and layer 2 does: z^2 to generate the final output, $(3.x)^2 = 9.x^2$ would be achieved by the composed network. However the composition of the functions is very complicated in most situations. Moreover, the devoted derivative of the composition must be calculated for each composition.

Because the derivative is dispersible, it can be back-propagated, which helps us solve the problem.

We know how to measure the derivative of the loss function, and we can spread the error backwards from the end to the beginning if we know how to calculate the derivative of each function from its composition.

Let's look at a simple linear example: we multiply the input three times to produce a hidden layer, and then we multiply the hidden (middle layer) two times to produce the output.

When we create a library with distinct features or layers where we can propagate each function (through direct use of this function) and how to propagate (by determining the derivative of the function), we can make up a complicated neural network. To know how to reverse errors using derivatives of such functions, we only need to keep stacks of function calls and their parameters during the forward pass. It can be accomplished by removing calls via the feature. This method is called self-differentiation and only requires each function to implement its derivative.

Every layer can transmit its results in the neural network for many other layers, in which case we add the deltas from all target layers to backflow. This can make the linear calculation stack a complicated calculation diagram.

3.3.6 Updating the Weights

The derivative is just the rate at which the error changes as the weight varies. Any major change in weights will result in chaotic behaviour due to the high level of non-linearity. It's important to remember that the derivative is only local at the place where it's being calculated.

Hence, we update the weights using the Delta Rule:

*New weight = Old weight — Derivative *Learning Rate*

If the derivative rate is positive, it means that the error is increased by an increase in weight, so the new weight should be lower.

If the derivative rate is negative, then we must increase our weight and reduce the error by increasing our weight.

If 0 is the derivative, we are at a minimum secure. No update on weights -> we've got a stable state is therefore required.

3.3.7 Iteration

When we modify weights using a small delta step, we need to learn more than one iteration.

The same thing applies to genetic algorithms, when after each generation we apply a small mutation rate and the fittest survive.

The gradient-decreasing force modifies the weight of the neural network to a lower global loss function after every iteration.

The resemblance is that the delta rule works as a mutation operator and a fitness function reduces loss.

The change is that the mutations of genetic algorithms are blind. Some mutations are bad, some are decent, but the best ones can endure more. But NN is smarter to update weight, as it is guided by the reduction of the gradient over the error.

CHAPTER 4

CONVOLUTIONAL NEURAL NETWORKS

4.1 Introduction

Here we will try to explain what Convolutional Neural Networks are and how they work. Our current implementation is a mask detection program that employs a classification CNN.

4.2 The Program

The program is a convolutional neural network, built using Keras. Our dataset is images of faces, split into 2 classes, one being images of people wearing masks, and the other being the images of people not wearing masks. We will use the Keras Sequential API, where we define a CNN layer by layer, then compile and run our dataset through it.

4.2.1 Importing all required libraries

Importing necessary libraries is very important when working on a project. One can always go the long way and build the whole program on their own, but if the theory behind your project is clear, it is a time-saving technique to use preconfigured libraries for your specific use.

Functions and Classes are pre-built, hence the tedious job of writing functions and creating classes is not necessary anymore. One can simply write the code for the idea they have in mind try making it better in every iteration without consuming much time.

The libraries which we used in this build of our CNN are:

1. <u>NumPy</u>: If your project involves more than a basic level of mathematical computation, the NumPy library comes very handy. With a plethora of inbuilt functions of various mathematical functions, it does help in tasks where mathematics is a primary concern.

2. <u>Pandas</u>: Pandas comes in handy when you need to manipulate and analyse data. It is built for Python, on which we are building our whole CNN.

3. <u>SK-Learn</u>: The specific modules that we are importing the huge SK-Learn library are the label encoder and the standard scalar.

4. Keras: The previous libraries and sub-libraries were all for the preprocessing of data. But Keras is only for building our CNN. We import convert to categorical, image to array and image loader.

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

(The code for importing of libraries can be seen in Appendix A.1)

4.2.2 Data Preprocessing

Data preprocessing is one of the most important steps while dealing with data. Whatever program you want to feed, it will require the data to be in a certain way, shape or form, and that is where data preprocessing steps in.

In the CNN that we are using, we will need our images in a specific size and aspect ratio so that our CNN runs smoothly. The various steps taken in this segment are:

1. We read the dataset of the pre-extracted features and categorize them via their IDs and their labels.

2. We do the same for the testing data of the pre-extracted features.

3. Now, we resize the images to a specific aspect ratio. We do this by taking the larger dimension of the image and scaling the smaller one to the previous one, giving us scaled images, which can be read with greater ease.

4. Next, we take an input array with various image IDs as NumPy arrays so that the longest side of the image is the maximum dimension length.

5. Then we insert the image array into a matrix, with all images categorized in order.

6. We now split the training data of the pre-extracted features and the images into training data and cross-validation data.

7. Finally, we load this new training data into a final matrix, which we will input into our CNN.

(The code for Data Preprocessing can be found in Appendix A.2)

4.2.3 Data Augmentation

To make our CNN model more robust in predicting the kind of plant we introduce to the bot, we will employ a technique, called Data Augmentation. Through experience, one realises that a CNN needs a substantial amount of data, and a small number like 600 will not give us accurate outputs for our problem, we need at least 800 or so images per class.

To overcome this issue, we augment our image data. We take the 16 images we have, and we randomly rotate them or apply a zoom transform on them, then replicate this image, to have a bigger dataset of images that are slightly different from one another.

Steps taken to augment data are:

1. Number all images to be augmented.
2. Unlock multithreading so that all transformations can be done in parallel.
3. Set the range of zoom transformation.
4. Set horizontal and vertical flipping functions to True.
5. Parse all images through the data augmentation function

4.2.4 Building the CNN

Convolutional Neural Networks are made up of neurons that have variable weights and biases. Each neuron receives an input from the previous layer, performs a dot product and can follow this with a non-linear function, which is optional.

The whole network still expresses a single differentiable cost function: starting from the dataset's image pixels on one end to the accuracy on the other. CNNs have a loss function, just like regular artificial neural networks, like SVM or SoftMax. On the output layer, you get a final result, pertaining to the problem you're aiming to solve with the CNN.

CNNs take advantage of the fact that the input fed into it is made up of images. Hence, their architecture is designed in a more sensible way. If we compare it to a regular NN, where the neurons are just 1 or 2 dimensional, the architecture of CNNs allows them to have neurons placed in 3 different dimensions: length, breadth, height.

The basic working of a CNN can be summarised by saying that it takes 3-Dimensional input and converts it into a 3-Dimensional output usually through a differentiable function.

The name of CNNs is derived from the word 'convolution'. The convolution operator, used mainly in Signal Processing is a mathematical technique of combining 2 signals to make the 3rd signal. It gives a mathematical relation between the input signal, the output signal and the impulse response.

This immensely important mathematical operator has found its way into Deep Learning is well. What a Convolutional Neural Network does is that it takes an image, which is a combination of 3 matrices of pixel colour density, and the 3 parameters being colours: Red, Green and Blue i.e. RGB. Let us take one of these matrices, and take a 'filter' (A smaller predefined matrix) with which we want to perform convolution. The process of convolution involves selecting a smaller matrix within your original image matrix, which is the size of the filter, multiplying them element-wise and adding them up to form a single number. Adding all such values, moving left to right and top to bottom gives us a new matrix, which is the output of the convolution we just performed.

Similarly, we perform a convolution for all three matrices, or we could call it a n x n x 3 matrix, with a f x f x 3 filter, giving us a (n - f - 1) x (n - f - 1) x 1 matrix as an output. The next step would be running the same input matrix through multiple filters, and stacking up these output matrices to form a (n - f - 1) x (n - f - 1) x y matrix, where y is the number of filters.

This operation can if done iteratively, by giving correct values for the filters, can do tremendous predictions. This is the method we are going to use to detect the type of plant, by applying this whole procedure to a dataset of leaves.

The Architecture of the CNN includes how many hidden layers the CNN will have, what will be the functions of these layers, the cost function, etc. We need to pick the right values for all these parameters for the optimal functioning of our CNN.

We will define the layers as:

1. Defining the input layers, giving details of the input images.

2. The first layer is a 2D Convolution Layer, followed by an activation layer with the ReLU Activation Function.

3. The second layer is a 2D Maxpooling Layer.

4. Next, we have another set of a Colvolutional, an Activation and a Maxpooling layer.

5. Now, we flatten our 2D matrix into an array.

6. We add a fully connected layer and then get the final output.

7. Finally, we compile the model of our CNN.

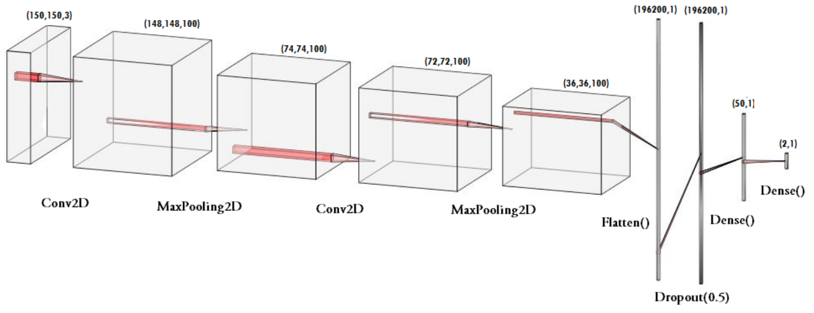


Figure 4.1: Representation of out CNN Model

Here, we can see that the image accurately represents our model. All the layers sequentially laid out one after the other, and finally compiled.

(The code for building the CNN can be found in Appendix A.3)

4.2.5 Training the Model

We will take the image augmenter generator, creating a minibatch. Feed these into the CNN, which will run indefinitely until we find the optimal solution.

(The code for training the CNN can be found in Appendix A.4)

4.2.6 Visualisation

This step is not vital to our project, but it does help us understand how the CNN learns and classifies according to its training. Here, we can see the graphs of Accuracy v. Epochs and Loss v. Epochs.

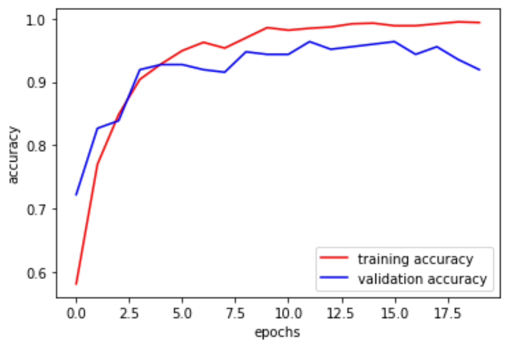


Figure 4.2: Accuracy v. Epochs

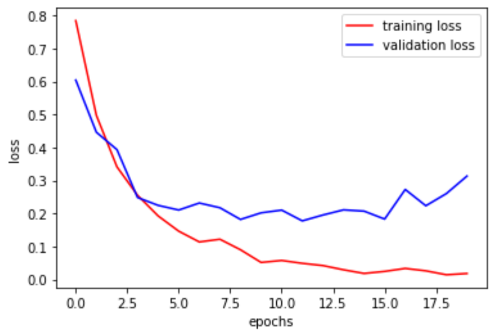
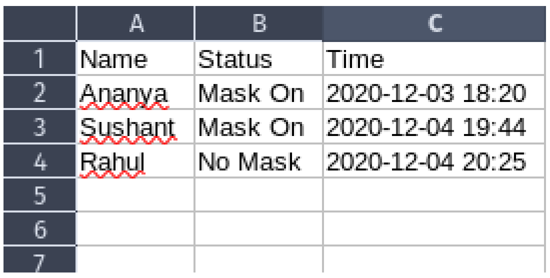


Figure 4.3: Loss v. Epochs

4.3 The Results

Our Mask Detection Algorithm got an accuracy of 96% approx., and to begin implementing future modules, we also wrote a small program that appends the data categorically in a CSV file which can then be converted into a full-fledged database.

(The code for running the Mask Detection Webcam Integration can be found in Appendix A.5)



	A	B	C
1	Name	Status	Time
2	Ananya	Mask On	2020-12-03 18:20
3	Sushant	Mask On	2020-12-04 19:44
4	Rahul	No Mask	2020-12-04 20:25
5			
6			
7			

Figure 4.4: Stored Data from the program.

CHAPTER 5

IMPLEMENTATION OF DASHBOARD

5.1 Introduction

We created a virtual dashboard to track the employee's attendance digitally. Data from the face detection and mask detection algorithms is collected and stored in the dashboard, which is built with ReactJS and Material UI.

5.2 Technologies Used

5.2.1 ReactJS

ReactJS is a library for JavaScript which allows you to build reusable UI components. React is a library for building modular user interfaces. It supports the development of reusable components of the interface, which show changes in data over time. In MVC reactors are frequently used as V. Reaction summarises the DOM from you, making your programming model easier and your performance improved. React can also use a node to render native applications on the server and react to native apps. React implements a reactive one-way data flow, which eliminates the boiler plate and makes it easier to reason than standard data binding.

Features include :

- **JSX** −JSX is an extension of JavaScript. JSX is not necessary, but strongly encouraged, for the reaction development.
- **Unidirectional data flow and Flux** - React implements an exclusive flow of data that makes your application reasonable. Flux is a model which helps maintain unidirectional data.
- Uses a virtual DOM-named JavaScript object. Because the virtual DOM JavaScript is faster than normal DOM, it increases the speed of the app.
- Patterns of components and data improve readability, making it easier to handle large apps.

5.2.2 Material UI

Material is a Google design framework that enables teams to create high-quality digital products for Android, iOS, Flutter, and the internet. The real world and its textures, as well as how they represent light and cast shadows, are sources of inspiration for Material Design. Paper and ink are reimagined as material surfaces.

Components :

Material Components are interactive building blocks that communicate attention, range, activation, error, hover, push, drag, and disabled states via a built-in states system. There are component libraries for Android, iOS, Flutter, and the internet.

Components address a variety of interface requirements, including :

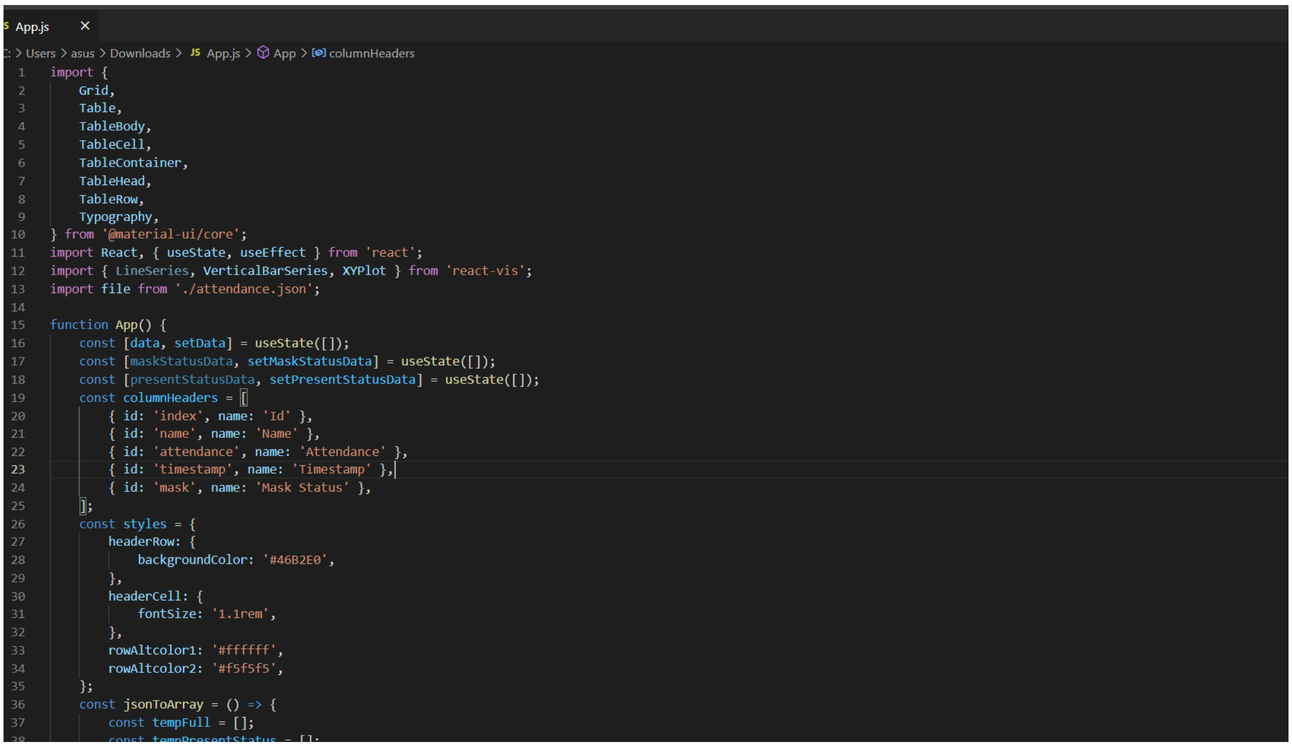
- **Display :** Using components such as cards, lists, and sheets to place and organise material.
- **Actions :** Using components like the floating action button, users can perform tasks.
- **Navigation :** Use components such as navigation drawers and tabs to allow users to navigate through the product.
- **Input :** Using components such as text fields, chips, and selection controls, users may enter information or make selections.
- **Communication :** Snackbars, banners, and dialogues are used to alert users to important information and notifications.

Theming :

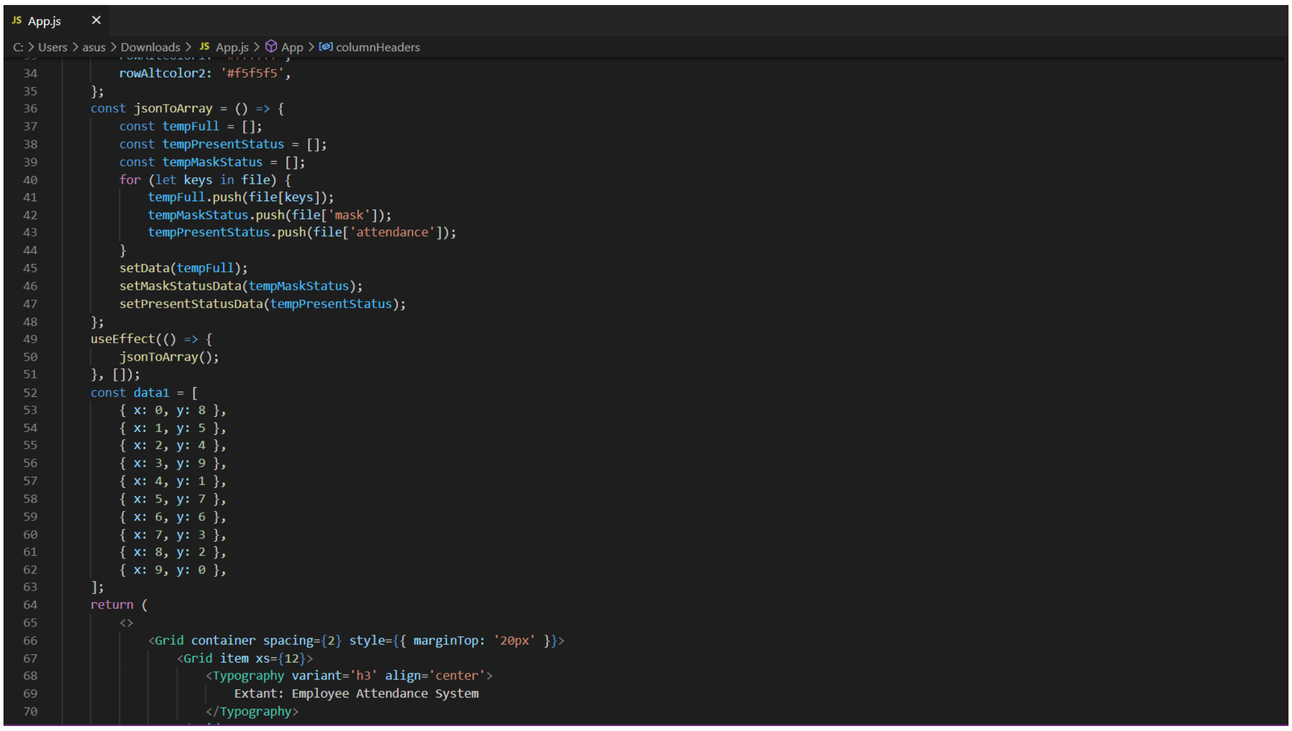
With built-in help and instructions for customising colours, typography styles, and corner shapes, Material Theming makes it simple to customise Material Design to fit the look and feel of your brand.

- **Color :** Material's colour scheme is a method for adding colour to a user interface that is well-organized. Main, secondary (brand colours), surface, context, and error are all global colour types with semantic names and defined use in components. To facilitate continuity and accessible contrast, each colour has a complementary colour that is used for elements that are put “on” top of it.
- **Shape :** Form types can be used to draw attention to specific components, convey their status, and articulate the brand. Based on their dimensions, all Material Components are categorised into shape groups (small, medium, large). These global styles allow you to easily adjust the shape of components that are identical in size. With the shape customization app, you can build your own shape types.
- **Typography :** From headlines to body text and captions, the Material Design type scale offers 13 typography types. Each style has a distinct meaning and function within the gui. The typeface, font weight, and letter case, for example, can all be changed to fit your brand and style.

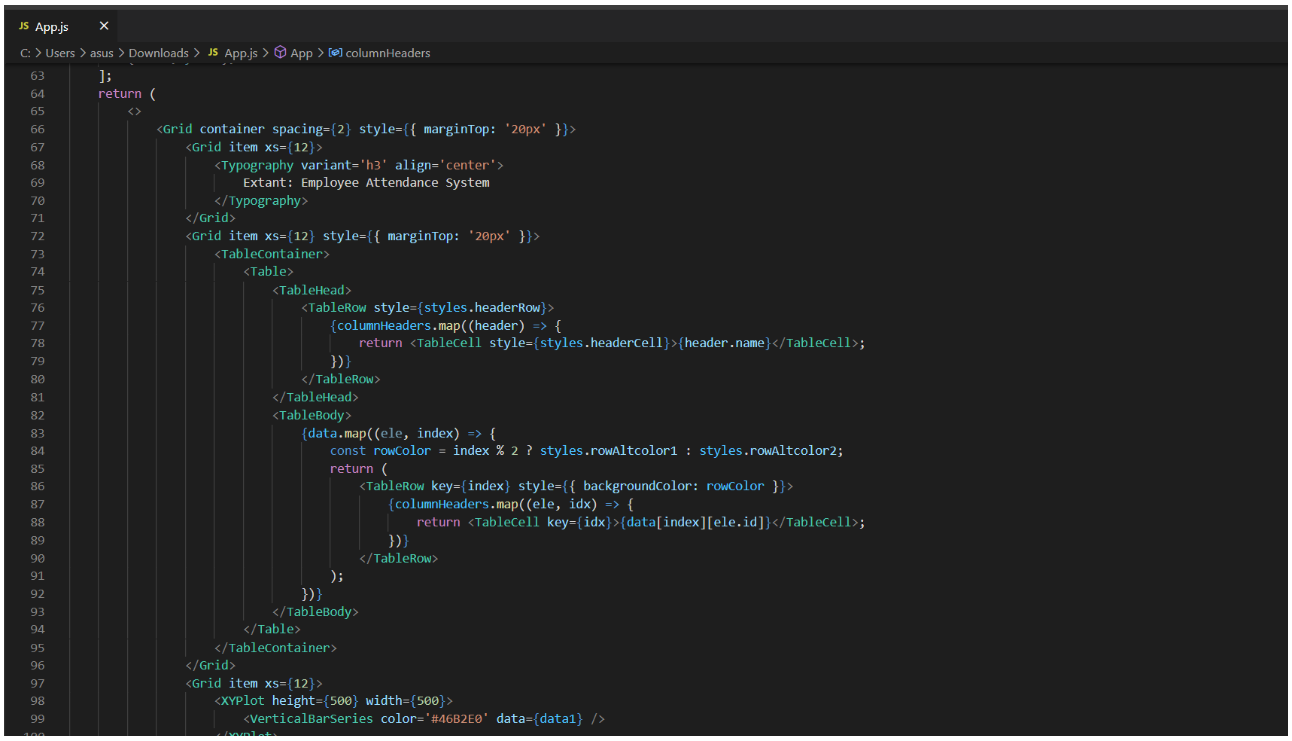
5.3 Implementation Code


App.js
C: > Users > asus > Downloads > App.js > App > columnHeaders

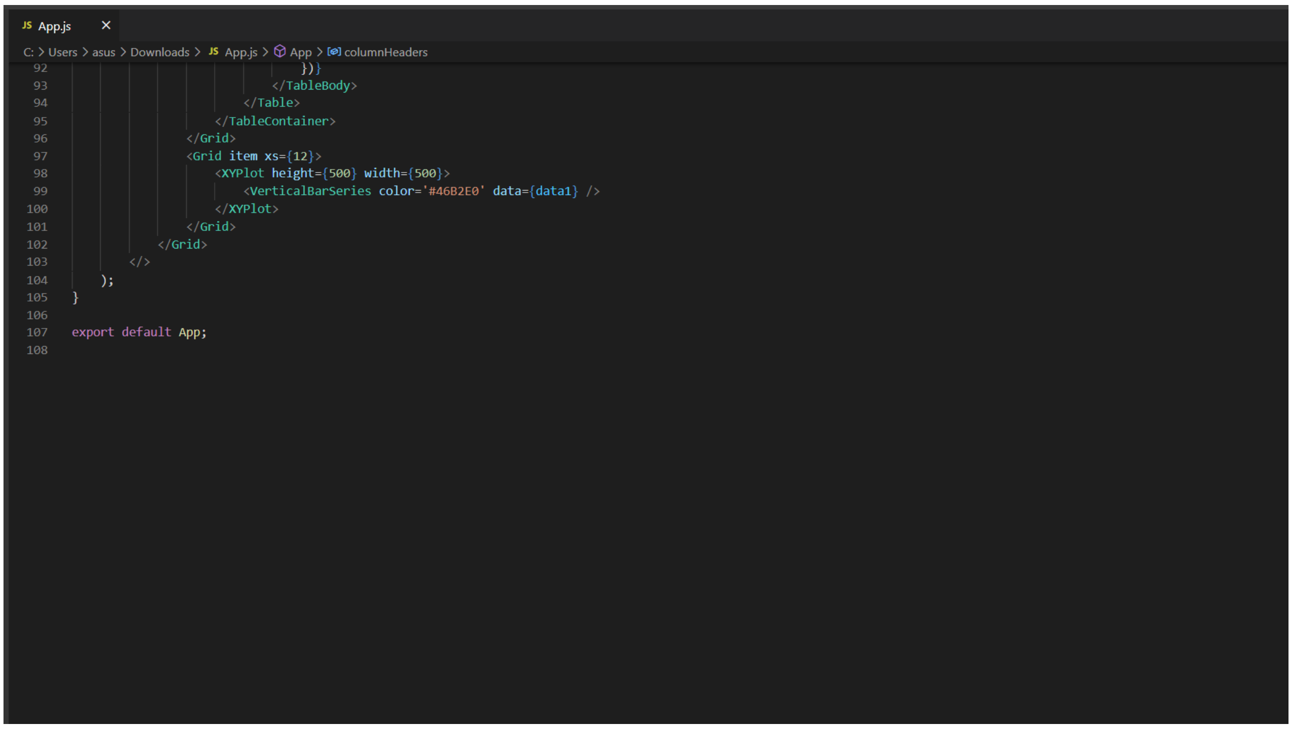
```
import {
    Grid,
    Table,
    TableBody,
    TableCell,
    TableContainer,
    TableHead,
    TableRow,
    Typography,
} from '@material-ui/core';
import React, { useState, useEffect } from 'react';
import { LineSeries, VerticalBarSeries, XYPlot } from 'react-vis';
import file from './attendance.json';

function App() {
    const [data, setData] = useState([]);
    const [maskStatusData, setMaskStatusData] = useState([]);
    const [presentStatusData, setPresentStatusData] = useState([]);
    const columnHeaders = [
        { id: 'index', name: 'Id' },
        { id: 'name', name: 'Name' },
        { id: 'attendance', name: 'Attendance' },
        { id: 'timestamp', name: 'Timestamp' },
        { id: 'mask', name: 'Mask Status' },
    ];
    const styles = {
        headerRow: {
            backgroundColor: '#46B2E0',
        },
        headerCell: {
            fontSize: '1.1rem',
        },
        rowAltcolor1: '#ffffff',
        rowAltcolor2: '#f5f5f5',
    };
    const jsonToArray = () => {
        const tempFull = [];
```


App.js
C: > Users > asus > Downloads > App.js > App > columnHeaders

```
        rowAltcolor2: '#f5f5f5',
    };
    const jsonToArray = () => {
        const tempFull = [];
        const tempPresentStatus = [];
        const tempMaskStatus = [];
        for (let keys in file) {
            tempFull.push(file[keys]);
            tempMaskStatus.push(file['mask']);
            tempPresentStatus.push(file['attendance']);
        }
        setData(tempFull);
        setMaskStatusData(tempMaskStatus);
        setPresentStatusData(tempPresentStatus);
    };
    useEffect(() => {
        jsonToArray();
    }, []);
    const data1 = [
        { x: 0, y: 8 },
        { x: 1, y: 5 },
        { x: 2, y: 4 },
        { x: 3, y: 9 },
        { x: 4, y: 1 },
        { x: 5, y: 7 },
        { x: 6, y: 6 },
        { x: 7, y: 3 },
        { x: 8, y: 2 },
        { x: 9, y: 0 },
    ];
    return (
        <>
            <Grid container spacing={2} style={{ marginTop: '20px' }}>
                <Grid item xs={12}>
                    <Typography variant='h3' align='center'>
                        Extant: Employee Attendance System
                    </Typography>
```

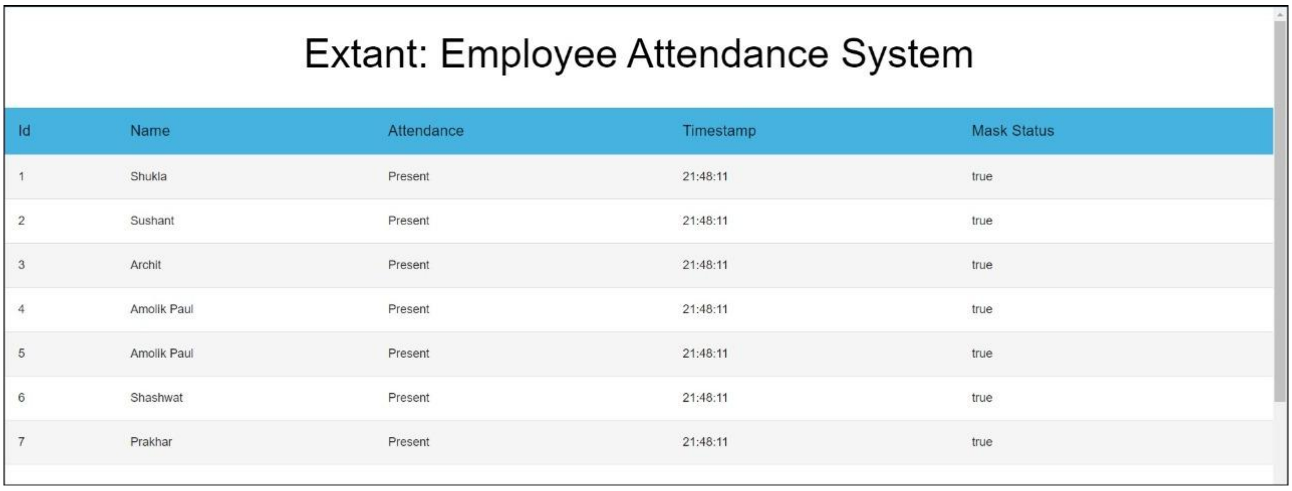

C: > Users > asus > Downloads > JS App.js > App > columnHeaders

```
    ];
    return (
        <>
            <Grid container spacing={2} style={{ marginTop: '20px' }}>
                <Grid item xs={12}>
                    <Typography variant='h3' align='center'>
                        Extant: Employee Attendance System
                    </Typography>
                </Grid>
                <Grid item xs={12} style={{ marginTop: '20px' }}>
                    <TableContainer>
                        <Table>
                            <TableHead>
                                <TableRow style={styles.headerRow}>
                                    {columnHeaders.map((header) => {
                                        return <TableCell style={styles.headerCell}>{header.name}</TableCell>;
                                    })}
                                </TableRow>
                            </TableHead>
                            <TableBody>
                                {data.map((ele, index) => {
                                    const rowColor = index % 2 ? styles.rowAltcolor1 : styles.rowAltcolor2;
                                    return (
                                        <TableRow key={index} style={{ backgroundColor: rowColor }}>
                                            {columnHeaders.map((ele, idx) => {
                                                return <TableCell key={idx}>{data[index][ele.id]}</TableCell>;
                                            })}
                                        </TableRow>
                                    );
                                })}
                            </TableBody>
                        </Table>
                    </TableContainer>
                </Grid>
                <Grid item xs={12}>
                    <XYPlot height={500} width={500}>
                        <VerticalBarSeries color='#46B2E0' data={data1} />
```


C: > Users > asus > Downloads > JS App.js > App > columnHeaders

```
                                })}
                            </TableBody>
                        </Table>
                    </TableContainer>
                </Grid>
                <Grid item xs={12}>
                    <XYPlot height={500} width={500}>
                        <VerticalBarSeries color='#46B2E0' data={data1} />
                    </XYPlot>
                </Grid>
            </Grid>
        </>
    );
}

export default App;
```

5.4 Result



Extant: Employee Attendance System

Id	Name	Attendance	Timestamp	Mask Status
1	Shukla	Present	21:48:11	true
2	Sushant	Present	21:48:11	true
3	Archit	Present	21:48:11	true
4	Amolk Paul	Present	21:48:11	true
5	Amolk Paul	Present	21:48:11	true
6	Shashwat	Present	21:48:11	true
7	Prakhar	Present	21:48:11	true

REFERENCES

The following papers were read while researching this project:

[1] J. J. Hopfield, "Artificial neural networks," in IEEE Circuits and Devices Magazine, vol. 4, no. 5, pp. 3-10, Sept. 1988, doi: 10.1109/101.8118.

[2] M. Asadullah and A. Raza, "An overview of home automation systems," *2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI)*, Rawalpindi, 2016, pp. 27-31.

[3] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), Antalya, 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

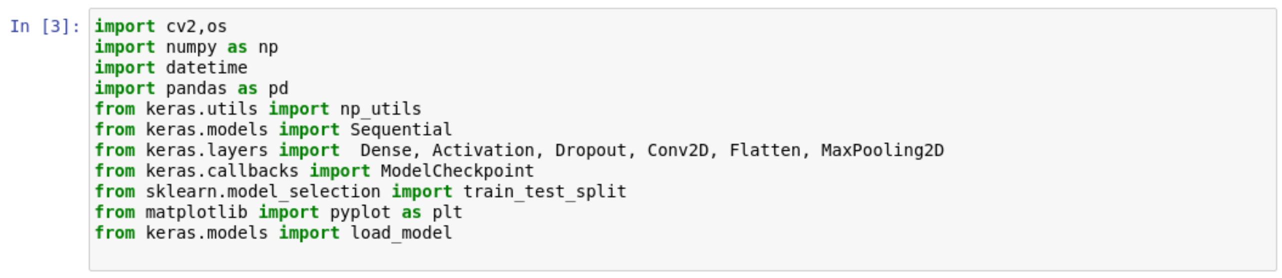
[4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Kauai, HI, USA, 2001, pp. I-I.

[5] C. Tang, Y. Feng, X. Yang, C. Zheng and Y. Zhou, "The Object Detection Based on Deep Learning," *2017 4th International Conference on Information Science and Control Engineering (ICISCE)*, Changsha, 2017, pp. 723-728.

[6] L. Cuimei, Q. Zhiliang, J. Nan and W. Jianhua, "Human face detection algorithm via Haar cascade classifier combined with three additional classifiers," *2017 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)*, Yangzhou, 2017, pp. 483-487.

APPENDIX A

A.1 Code for importing libraries



```
In [3]: import cv2,os
        import numpy as np
        import datetime
        import pandas as pd
        from keras.utils import np_utils
        from keras.models import Sequential
        from keras.layers import  Dense, Activation, Dropout, Conv2D, Flatten, MaxPooling2D
        from keras.callbacks import ModelCheckpoint
        from sklearn.model_selection import train_test_split
        from matplotlib import pyplot as plt
        from keras.models import load_model
```

A.2 Code for data pre-processing

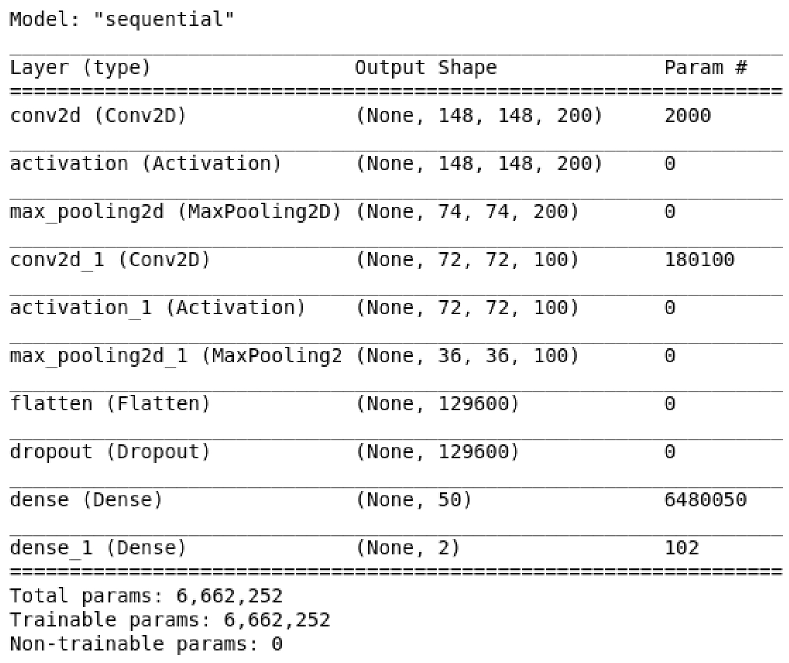
```
In [4]: data_path = r'/home/gryffindorito/observations/experiements/data'
        categories = os.listdir(data_path)
        labels = [i for i in range(len(categories))]
        label_dict = dict(zip(categories,labels))
        print(label_dict)
        print(categories)
        print(labels)

        {'without_mask': 0, 'with_mask': 1}
        ['without_mask', 'with_mask']
        [0, 1]
```

```
In [5]: img_size = 150
        data = []
        target = []
        for category in categories:
            folder_path = os.path.join(data_path,category)
            img_names = os.listdir(folder_path)

            for img_name in img_names:
                img_path = os.path.join(folder_path,img_name)
                img = cv2.imread(img_path)
                try:
                    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
                    resized = cv2.resize(gray,(img_size,img_size))
                    data.append(resized)
                    target.append(label_dict[category])

                except Exception as e:
                    print("Exception: ",e)
```

```
In [6]: data = np.array(data)/255.0
        #data values are normalized
        #reshaping of data
        data = np.reshape(data,(data.shape[0],img_size,img_size,1))
        target = np.array(target)
        new_target = np_utils.to_categorical(target)
        #saving the files
        np.save('data',data)
        np.save('target',new_target)
```

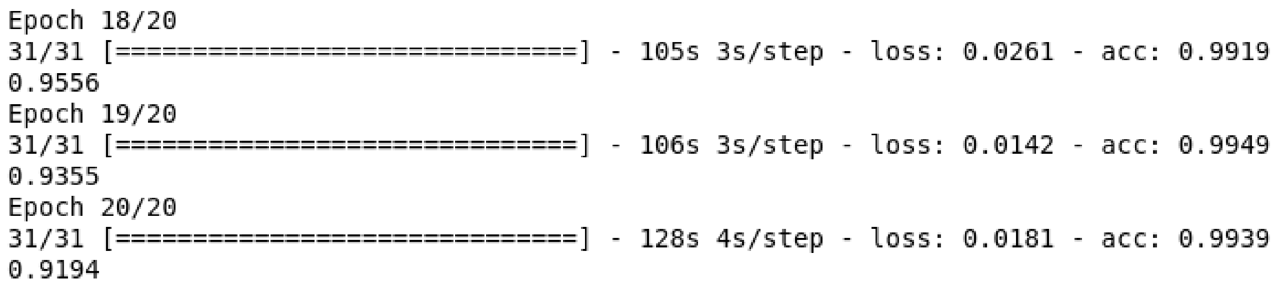
A.3 Building the CNN

```
In [11]:
         data = np.load('data.npy')
         target = np.load('target.npy')
         model = Sequential()
         model.add(Conv2D(200,(3,3),input_shape=data.shape[1:]))
         model.add(Activation('relu'))
         model.add(MaxPooling2D(pool_size=(2,2)))
         model.add(Conv2D(100,(3,3)))
         model.add(Activation('relu'))
         model.add(MaxPooling2D(pool_size=(2,2)))
         model.add(Flatten())
         model.add(Dropout(0.5))
         model.add(Dense(50,activation='relu'))
         model.add(Dense(2,activation='softmax'))
         model.compile(optimizer='adam', loss='categorical_crossentropy', metrics = ['acc'])
         model.summary()
```



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 148, 148, 200)     2000      
_________________________________________________________________
activation (Activation)      (None, 148, 148, 200)     0         
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 200)       0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 100)       180100    
_________________________________________________________________
activation_1 (Activation)    (None, 72, 72, 100)       0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 100)       0         
_________________________________________________________________
flatten (Flatten)            (None, 129600)            0         
_________________________________________________________________
dropout (Dropout)            (None, 129600)            0         
_________________________________________________________________
dense (Dense)                (None, 50)                6480050   
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 102       
=================================================================
Total params: 6,662,252
Trainable params: 6,662,252
Non-trainable params: 0
```

A.4 Training the CNN



```
Epoch 18/20
31/31 [==============================] - 105s 3s/step - loss: 0.0261 - acc: 0.9919
0.9556
Epoch 19/20
31/31 [==============================] - 106s 3s/step - loss: 0.0142 - acc: 0.9949
0.9355
Epoch 20/20
31/31 [==============================] - 128s 4s/step - loss: 0.0181 - acc: 0.9939
0.9194
```

A.5 Running the Webcam Interface

In [8]:

```python
name =input("Enter your name : ")
final_list = []
temp_list = []
f = 0
base_path = "/home/gryffindorito/Documents/ProjectCSV/"

while(True):
    ret,frame = video_capture.read()
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray,1.3,5)



    for x,y,w,h in faces:
        face_img = gray[y:y+w,x:x+h]
        resized = cv2.resize(face_img,(img_size,img_size))
        normalized = resized/255.0
        reshaped = np.reshape(normalized,(1,img_size,img_size,1))
        result = model.predict(reshaped)
        acc = list(result[0])


        if acc[1] > 0.87 and f == 0:
            temp_list.append(name)
            temp_list.append("Mask On")
            a = datetime.datetime.now()
            time = str(a)
            x = time.rindex(':')
            temp_list.append(time[0:x])
            final_list.append(temp_list)
            f = 1
```

```python
        print(final_list)
        label = np.argmax(result,axis=1)[0]
        cv2.rectangle(frame,(x,y),(x+w,y+h),color_dict[label],2)
        cv2.putText(frame,labels_dict[label],(x,y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)


    cv2.imshow('Video',frame)
    key=cv2.waitKey(1)

    if(key==27):
        break;

cv2.destroyAllWindows()
video_capture.release()
if f != 1:
    temp_list.append(name)
    temp_list.append("No Mask")
    b = datetime.datetime.now()
    time = str(b)
    x = time.rindex(':')
    temp_list.append(time[0:x])
    final_list.append(temp_list)


#df_save_path = os.path.join(base_path, "Extant", "MaskCheck.csv")
df = pd.DataFrame(final_list)
df.to_csv("MaskCheck.csv", mode = 'a', index=False, header=False)
```