

REAL TIME DRIVER DROWSINESS DETECTION USING FACIAL FEATURES

**Submitted in partial fulfillment of the requirements for the award of Bachelor of science
Degree in computer science**

By

BHUVANESWARI.S(Reg. No. 39290012)

HARITHA.DR(Reg. No. 39290032)



**DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMIL NADU**

MARCH – 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY) Accredited with Grade “A”
by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119 www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of BHUVANESWARI.S(Reg. No. 39290012) and HARITHA.DR(Reg. No. 39290032) who carried out the project entitled

REAL TIME DRIVER DROWSINESS DETECTION USING FACIAL FEATURES

under my supervision from November 2020 to April 2022.

Internal Guide DR. MALINI DEEPIKA

Head of the Department DR. L LAKSHMANAN M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

BHUVANESWARI.S (Reg. No. 39290012) and HARITHA.DR(Reg. No. 39290032) hereby declare that the Project Report entitled

"REAL TIME DRIVER DROWSINESS DETECTION USING FACIAL FEATURES" done by me under the guidance of **Dr. MALINIDEEPIKA** is submitted in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D.**, Dean, School of Computing and **Dr. S. VIGNESHWARI, M.E., Ph.D.**, and **Dr. L. LAKSHMANAN, M.E., Ph.D.**, Head of the Department, Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. MALINIDEEPIKA** for his valuable guidance, suggestions and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways For the Completion of the project.

ABSTRACT

The advancement in computer vision has assisted drivers in the form of automatic self-driving cars etc. The misadventures are caused by driver's fatigue and drowsiness about 20%. It poses a serious problem for which several approaches were proposed. However, they are not suitable for real-time processing.

The major challenges faced by these methods are robustness to handle variation in human face and lightning conditions. We aim to implement an intelligent processing system that can reduce road accidents drastically.

This approach enables us to identify driver's face characteristics like eye closure percentage, eye-mouth aspect ratios, blink rate, yawning, head movement, etc. In this system, the driver is continuously monitored by using a webcam.

The driver's face and the eye are detected using cascade classifiers. Eye images are extracted and fed to Custom designed Convolutional Neural Network for classifying whether both left and right eye are closed.

Based on the classification, the eye closure score is calculated. If the driver is found to be drowsy, an alarm will be triggered.

TABLE OF CONTENTS

CHAPTER No	TITLE	PAGE
	LIST OF FIGURES	Vi
1	INTRODUCTION	
1.1	About the Project	
2	SYSTEM ANALYSIS	
2.1	Existing system	
2.2	Proposed system	
3	REQUIREMENTS SPECIFICATION	
3.1	Introduction	
3.2	Hardware and Software specification	
3.3	Technologies Used	
4.1	Design and Implementation Constraints	
4.2	Other Nonfunctional Requirements	
5	Architecture Diagram	
5.1	SYSTEM DESIGN – DETAILED	

6	Modules	
6.1	Module explanation	
7	CODING AND TESTING	
7.1	Coding	
7.2	Coding standards	
7.3	Test procedure	
7.4	Test data and output	
	SOURCE CODE	
	SNAP SHOTS	
	REFERENCES	

LIST OF FIGURES

5.1 Sequence Diagram

5.2 Use Case Diagram

5.3 Activity Diagram

5.4 Collaboration Diagram:

CHAPTER 1

INTRODUCTION

Aim:

This paper aim to detect Real time driver's fatigue state using Convolutional Neural Network (CNN)

Abstract:

Accidents are more due to the driver's drowsiness; it has been recorded that more than 40% of chances that accidents occur while the driver is in drowsiness state. It's very important that the driver must be alert while driving the car. Few methods are intrusive and distract the driver, some require expensive sensors and data handling. Therefore, in Existing study, a low cost, real time driver's drowsiness detection system is developed with acceptable accuracy. Facial landmarks on the detected face are pointed and subsequently the eye aspect ratio and mouth opening ratio are computed and depending on their values, drowsiness is detected based on developed adaptive thresholding. In the proposed system, a webcam records the video and the driver's face is detected in each frame employing image processing techniques. A novel system for evaluating the driver's level of fatigue based on face tracking and facial key point detection. In order to track the driver's face using CNN (Convolution Neural Network) and then the facial regions of detection based on facial key points. Then the eyes and mouth will be detected if the eye is closed the alert system will be displayed.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The driver facial features are identified by Support Vector Machine. But it is not accuracy and some major problems are occurred. The Detect faces are pointed and this point gives eye aspect ratio. But the ratio is not working properly. So we will move to the proposed system.

2.2 PROPOSED SYSTEM

In modern days, we see how car accidents are increasing due to many reasons like drowsy driving or drunk driving or speeding and many more reasons. Hence we develop a modern solution, where my system will alert the driver if the driver is sleeping. In the proposed system, a webcam records the video and the driver's face is detected in each frame for image processing techniques. A novel system for evaluating the driver's level of fatigue based on face tracking and facial key point detection. In order to track the driver's face using CNN (Convolution Neural Network) and then the facial regions of detection based on facial key points. Then the eyes and mouth will be detected if the eye is closed the alert system will be displayed.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

3.1 INTRODUCTION

Many safety connected driving supporter schemes decreased the danger of four-wheeler accidents, and investigations depicted weariness to be a major reason for four wheeler accidents. A car organization announced an idea that whole deadly accidents (17%) would be attributed to weary drivers. Many revisions showed by Volkswagen AG specify that 5-25% of all accidents are produced by the sleeping of the driver. The lack of concentration damages steering actions and decreases response period, and revisions illustrate that sleepiness raises the threat of crashes demand for a dependable intelligent driver sleepiness sensing system. The aim is to create an intelligent processing scheme to avoid road accidents. This can be done by monitoring the drowsiness and warning drivers of inattention to prevent accidents. Based on the literature survey, the driver's drowsiness can be detected based on three factors such as physiological, behavioral, and vehicle-based measurements. But these approaches pose some disadvantages in certain realtime scenarios. So, we aim to apply Deep Learning algorithms to this problem statement. Our methodology is to use a Convolutional Neural Network (CNN). CNN offers a computerized and effective way to categorize the driver as drowsy or non-drowsy correctly.

3.2 HARDWARE AND SOFTWARE SPECIFICATION

3.2.1 HARDWARE REQUIREMENTS

Hard Disk	:	500GB and Above
RAM	:	4GB and Above
Processor	:	I3 and Above
—Webcam - 1		

3.2.2 SOFTWARE REQUIREMENTS

✓ Operating System	:	Windows 7 , 8, 10 (64 bit)
✓ Software	:	Python

✓ Tools : Python 3.7 IDLE, jupyter NoteBook and Spyder

3.3 TECHNOLOGIES USED

CNN

3.3.1 Introduction to Python

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- System scripting

Introduction

Convolutional neural networks (CNN) sound like a weird combination of biology and math with a little CS sprinkled in, but these networks have been some of the influential

innovations in the field of computer vision. 2012 was the first year that neural nets grew to prominence as Alex Krizhevsky used them to win that year's ImageNet competition (basically, the annual Olympics of computer vision), dropping the classification error record from 26% to

15%, an astounding improvement at the time. Ever since then, a host of companies have been using deep learning at the core of their services. Facebook uses neural nets for their automatic tagging algorithms, Google for their photo search, Amazon for their product recommendations, Pinterest for their home feed personalization, and Instagram for their search infrastructure.

However, the classic, and arguably most popular, use case of these networks is for image processing. Within image processing, let's take a look at how to use these CNNs for image classification.

The Problem Space

Image classification is the task of taking an input image and outputting a class (a cat, dog, etc) or a probability of classes that best describes the image. For humans, this task of recognition is one of the first skills we learn from the moment we are born and is one that comes naturally and effortlessly as adults. Without even thinking twice, we're able to quickly and seamlessly identify the environment we are in as well as the objects that surround us. When we see an image or just when we look at the world around us, most of the time we are able to immediately characterize the scene and give each object a label, all without even consciously noticing. These skills of being able to quickly recognize patterns, generalize from prior knowledge, and adapt to different image environments are ones that we do not share with our fellow machines.



What We See

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 43 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

What Computers See

Inputs and Outputs

When a computer sees an image (takes an image as input), it will see an array of pixel values. Depending on the resolution and size of the image, it will see a 32 x 32 x 3 array of numbers (The 3 refers to RGB values). Just to drive home the point, let's say we have a color image in JPG form and its size is 480 x 480. The representative array will be 480 x 480 x 3. Each of these numbers is given a value from 0 to 255 which describe the pixel intensity at that point.

These numbers, while meaningless to us when we perform image classification, are the only inputs available to the computer. The idea is that you give the computer this array of numbers and it will output numbers that describe the probability of the image being a certain class (.80 for cat, .15 for dog, .05 for bird, etc).

What We Want the Computer to do

Now that we know the problem as well as the inputs and outputs, let's think about how to approach this. What we want the computer to do is to be able to differentiate between all the images it's given and figure out the unique features that make a dog a dog or that make a cat a cat. This is the process that goes on in our minds subconsciously as well. When we look at a picture of a dog, we can classify it as such if the picture has identifiable features such as paws or 4 legs. In a similar way, the computer is able to perform image classification by looking for low level features such as edges and curves, and then building up to more abstract concepts through a series of convolutional layers. This is a general overview of what CNN does. Let's get into the specifics.

Biological Connection

But first, a little background. When you first heard of the term convolutional neural networks, you may have thought of something related to neuroscience or biology, and you would be right. Sort of. CNNs do take biological inspiration from the visual cortex. The visual cortex has small regions of cells that are sensitive to specific regions of the visual field. This idea was expanded upon by a fascinating experiment by Hubel and Wiesel in 1962 ([Video](#)) where they showed that some individual neuronal cells in the brain responded (or fired) only in the presence of edges of a certain orientation. For example, some neurons fired when exposed to vertical edges and some when shown horizontal or diagonal edges. Hubel and Wiesel found out that all of these neurons were organized in a columnar architecture and that together, they were able to produce visual perception. This idea of specialized components inside of a system having specific tasks (the neuronal cells in the visual cortex looking for specific characteristics) is one that machines use as well, and is the basis behind CNNs.

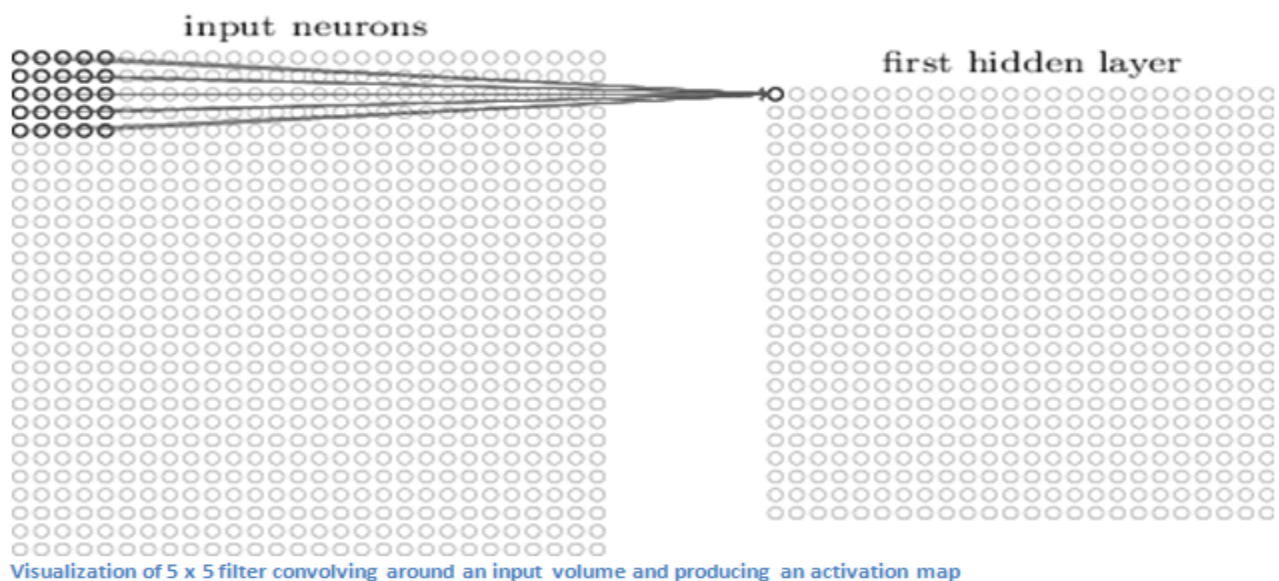
Structure

Back to the specifics. A more detailed overview of what CNNs do would be that you take the image, pass it through a series of convolutional, nonlinear, pooling (down sampling), and fully connected layers, and get an output. As we said earlier, the output can be a single class or a probability of classes that best describes the image. Now, the hard part understands what each of these layers do. So let's get into the most important one.

First Layer – Math Part

The first layer in a CNN is always a Convolutional Layer. First thing to make sure you remember is what the input to this conv (I'll be using that abbreviation a lot) layer is. Like we mentioned before, the input is a 32 x 32 x 3 array of pixel values. Now, the best way to explain a

conv layer is to imagine a flashlight that is shining over the top left of the image. Let's say that the light this flashlight shines covers a 5×5 area. And now, let's imagine this flashlight sliding across all the areas of the input image. In machine learning terms, this flashlight is called a filter (or sometimes referred to as a neuron or a kernel) and the region that it is shining over is called the receptive field. Now this filter is also an array of numbers (the numbers are called weights or parameters). A very important note is that the depth of this filter has to be the same as the depth of the input (this makes sure that the math works out), so the dimension of this filter is $5 \times 5 \times 3$. Now, let's take the first position the filter is in for example. It would be the top left corner. As the filter is sliding, or convolving, around the input image, it is multiplying the values in the filter with the original pixel values of the image (aka computing element wise multiplications). These multiplications are all summed up (mathematically speaking, this would be 75 multiplications in total). So now you have a single number. Remember, this number is just representative of when the filter is at the top left of the image. Now, we repeat this process for every location on the input volume. (Next step would be moving the filter to the right by 1 unit, then right again by 1, and so on). Every unique location on the input volume produces a number. After sliding the filter over all the locations, you will find out that what you're left with is a $28 \times 28 \times 1$ array of numbers, which we call an activation map or feature map. The reason you get a 28×28 array is that there are 784 different locations that a 5×5 filter can fit on a 32×32 input image. These 784 numbers are mapped to a 28×28 array.



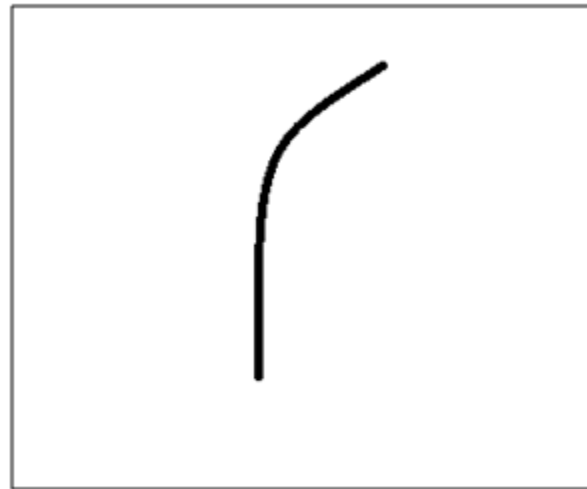
Let's say now we use two $5 \times 5 \times 3$ filters instead of one. Then our output volume would be $28 \times 28 \times 2$. By using more filters, we are able to preserve the spatial dimensions better. Mathematically, this is what's going on in a convolutional layer.

First Layer – High Level Perspective

However, let's talk about what this convolution is actually doing from a high level. Each of these filters can be thought of as **feature identifiers**. When I say features, I'm talking about things like straight edges, simple colors, and curves. Think about the simplest characteristics that all images have in common with each other. Let's say our first filter is $7 \times 7 \times 3$ and is going to be a curve detector. (In this section, let's ignore the fact that the filter is 3 units deep and only consider the top depth slice of the filter and the image, for simplicity.) As a curve detector, the filter will have a pixel structure in which there will be higher numerical values along the area that is a shape of a curve (Remember, these filters that we're talking about as just numbers!).

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

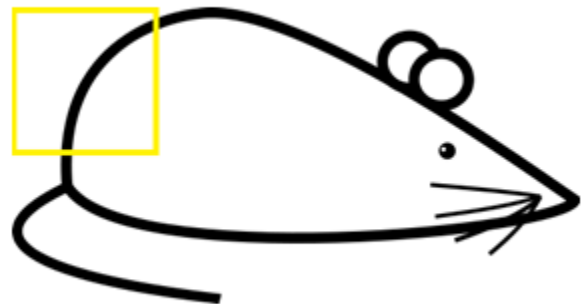


Visualization of a curve detector filter

Now, let's go back to visualizing this mathematically. When we have this filter at the top left corner of the input volume, it is computing multiplications between the filter and pixel values at that region. Now let's take an example of an image that we want to classify, and let's put our filter at the top left corner.

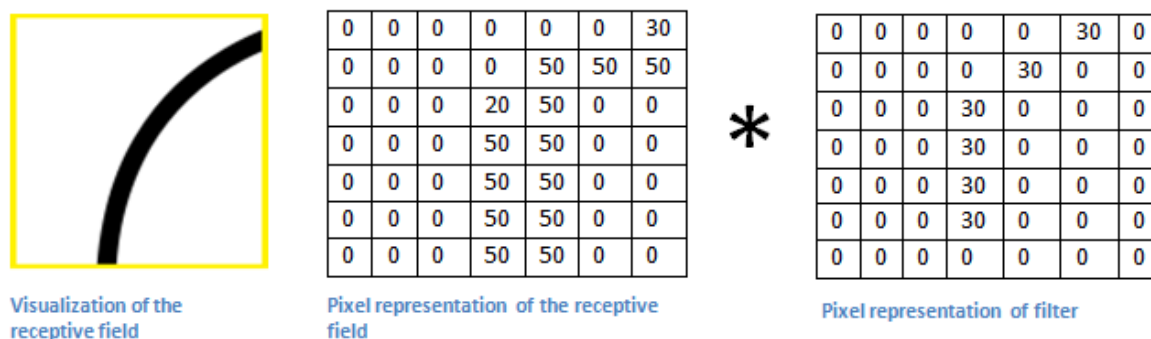


Original image



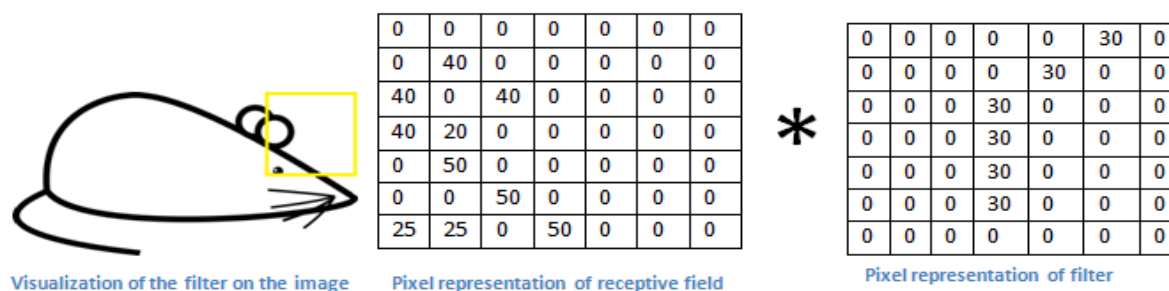
Visualization of the filter on the image

Remember, what we have to do is multiply the values in the filter with the original pixel values of the image.



Multiplication and Summation = $(50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600$ (A large number!)

Basically, in the input image, if there is a shape that generally resembles the curve that this filter is representing, then all of the multiplications summed together will result in a large value! Now let's see what happens when we move our filter.



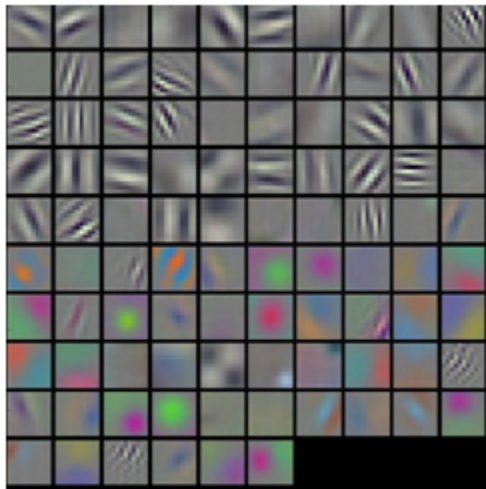
Multiplication and Summation = 0

The value is much lower! This is because there wasn't anything in the image section that responded to the curve detector filter. Remember, the output of this conv layer is an activation map. So, in the simple case of a one filter convolution (and if that filter is a curve detector), the activation map will show the areas in which there are mostly likely to be curves in the picture. In this example, the top left value of our 26 x 26 x 1 activation map (26 because of the 7x7 filter instead of 5x5) will be 6600. This high value means that it is likely that there is some sort of curve in the input volume that caused the filter to activate. The top right value in our activation map will be 0 because there wasn't anything in the input volume that caused the filter to activate (or more simply said, there wasn't a curve in that region of the original image). Remember, this is just for one filter. This is just a filter that is going to detect lines that curve outward and to the right. We can have other filters for lines that curve to the left or for straight edges. The more

filters, the greater the depth of the activation map, and the more information we have about the input volume.

Disclaimer:

The filter I described in this section was simplistic for the main purpose of describing the math that goes on during a convolution. In the picture below, you'll see some examples of actual visualizations of the filters of the first conv layer of a trained network. Nonetheless, the main argument remains the same. The filters on the first layer convolve around the input image and "activate" (or compute high values) when the specific feature it is looking for is in the input volume.



Visualizations of filters

Training (AKA: What Makes this Stuff Work)

Now, this is the one aspect of neural networks that I purposely haven't mentioned yet and it is probably the most important part. There may be a lot of questions you had while reading. How do the filters in the first conv layer know to look for edges and curves? How does the fully connected layer know what activation maps to look at? How do the filters in each layer know what values to have? The way the computer is able to adjust its filter values (or weights) is through a training process called **backpropagation**.

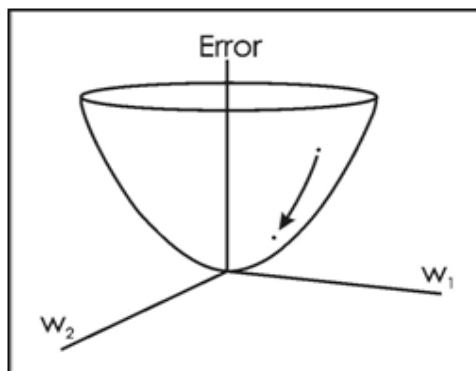
Before we get into backpropagation, we must first take a step back and talk about what a neural network needs in order to work. At the moment we all were born, our minds were fresh. We didn't know what a cat or dog or bird was. In a similar sort of way, before the CNN starts, the weights or filter values are randomized. The filters don't know to look for edges and curves. The filters in the higher layers don't know to look for paws and beaks. As we grew older however,

our parents and teachers showed us different pictures and images and gave us a corresponding label. This idea of being given an image and a label is the training process that CNNs go through. Before getting too into it, let's just say that we have a training set that has thousands of images of dogs, cats, and birds and each of the images has a label of what animal that picture is. Back to backprop.

So backpropagation can be separated into 4 distinct sections, the forward pass, the loss function, the backward pass, and the weight update. During the **forward pass**, you take a training image which as we remember is a 32 x 32 x 3 array of numbers and pass it through the whole network. On our first training example, since all of the weights or filter values were randomly initialized, the output will probably be something like [.1 .1 .1 .1 .1 .1 .1 .1 .1 .1], basically an output that doesn't give preference to any number in particular. The network, with its current weights, isn't able to look for those low level features or thus isn't able to make any reasonable conclusion about what the classification might be. This goes to the **loss function** part of backpropagation. Remember that what we are using right now is training data. This data has both an image and a label. Let's say for example that the first training image inputted was a 3. The label for the image would be [0 0 0 1 0 0 0 0 0]. A loss function can be defined in many different ways but a common one is MSE (mean squared error), which is $\frac{1}{2}$ times (actual - predicted) squared.

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

Let's say the variable L is equal to that value. As you can imagine, the loss will be extremely high for the first couple of training images. Now, let's just think about this intuitively. We want to get to a point where the predicted label (output of the ConvNet) is the same as the training label (This means that our network got its prediction right). In order to get there, we want to minimize the amount of loss we have. Visualizing this as just an optimization problem in calculus, we want to find out which inputs (weights in our case) most directly contributed to the loss (or error) of the network.



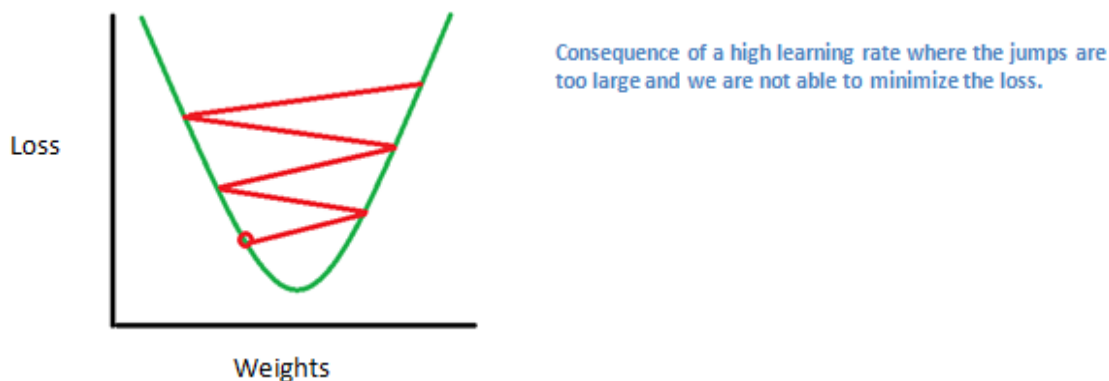
One way of visualizing this idea of minimizing the loss is to consider a 3-D graph where the weights of the neural net (there are obviously more than 2 weights, but let's go for simplicity) are the independent variables and the dependent variable is the loss. The task of minimizing the loss involves trying to adjust the weights so that the loss decreases. In visual terms, we want to get to the lowest point in our bowl shaped object. To do this, we have to take a derivative of the loss (visual terms: calculate the slope in every direction) with respect to the weights.

This is the mathematical equivalent of a dL/dW where W are the weights at a particular layer. Now, what we want to do is perform a **backward pass** through the network, which is determining which weights contributed most to the loss and finding ways to adjust them so that the loss decreases. Once we compute this derivative, we then go to the last step which is the **weight update**. This is where we take all the weights of the filters and update them so that they change in the opposite direction of the gradient.

$$W = W_i - \eta \frac{dL}{dW}$$

w = Weight
w_i = Initial Weight
η = Learning Rate

The **learning rate** is a parameter that is chosen by the programmer. A high learning rate means that bigger steps are taken in the weight updates and thus, it may take less time for the model to converge on an optimal set of weights. However, a learning rate that is too high could result in jumps that are too large and not precise enough to reach the optimal point.



The process of forward pass, loss function, backward pass, and parameter update is one training iteration. The program will repeat this process for a fixed number of iterations for each set of training images (commonly called a batch). Once you finish the parameter update on the last training example, hopefully the network should be trained well enough so that the weights of the layers are tuned correctly.

Testing

Finally, to see whether or not our CNN works, we have a different set of images and labels (can't double dip between training and test!) and pass the images through the CNN. We compare the outputs to the ground truth and see if our network works!

How Companies Use CNNs

Data, data, data. The companies that have lots of this magic 4 letter word are the ones that have an inherent advantage over the rest of the competition. The more training data that you can give to a network, the more training iterations you can make, the more weight updates you can make, and the better tuned to the network is when it goes to production. Facebook (and Instagram) can use all the photos of the billion users it currently has, Pinterest can use information of the 50 billion pins that are on its site, Google can use search data, and Amazon can use data from the millions of products that are bought every day. And now you know the magic behind how they use it.

Disclaimer

While this post should be a good start to understanding CNNs, it is by no means a comprehensive overview. Things not discussed in this post include the nonlinear and pooling layers as well as hyperparameters of the network such as filter sizes, stride, and padding. Topics like network architecture, batch normalization, vanishing gradients, dropout, initialization techniques, non-convex optimization, biases, and choices of loss functions, data augmentation, regularization methods, computational considerations, modifications of backpropagation, and more were also not discussed (yet).

CHAPTER 4

4.1 Design and Implementation Constraints

4.1.1 Constraints in Analysis

- ◆ Constraints as Informal Text
- ◆ Constraints as Operational Restrictions
- ◆ Constraints Integrated in Existing Model Concepts
- ◆ Constraints as a Separate Concept
- ◆ Constraints Implied by the Model Structure

4.1.2 Constraints in Design

- ◆ Determination of the Involved Classes
- ◆ Determination of the Involved Objects
- ◆ Determination of the Involved Actions

- ◆ Determination of the Require Clauses
- ◆ Global actions and Constraint Realization

4.1.3 Constraints in Implementation

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

4.2 Other Nonfunctional Requirements

4.2.1 Performance Requirements

The application at this side controls and communicates with the following three main general components.

embedded browser in charge of the navigation and accessing to the web service;

Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

4.2.2 Safety Requirements

1. The software may be safety-critical. If so, there are issues associated with its integrity level

2. The software may not be safety-critical although it forms part of a safety-critical system.
For example, software may simply log transactions.
3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.
4. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.
5. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.
6. Systems with different requirements for safety levels must be separated.
7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

CHAPTER 5

5.1 Architecture Diagram:

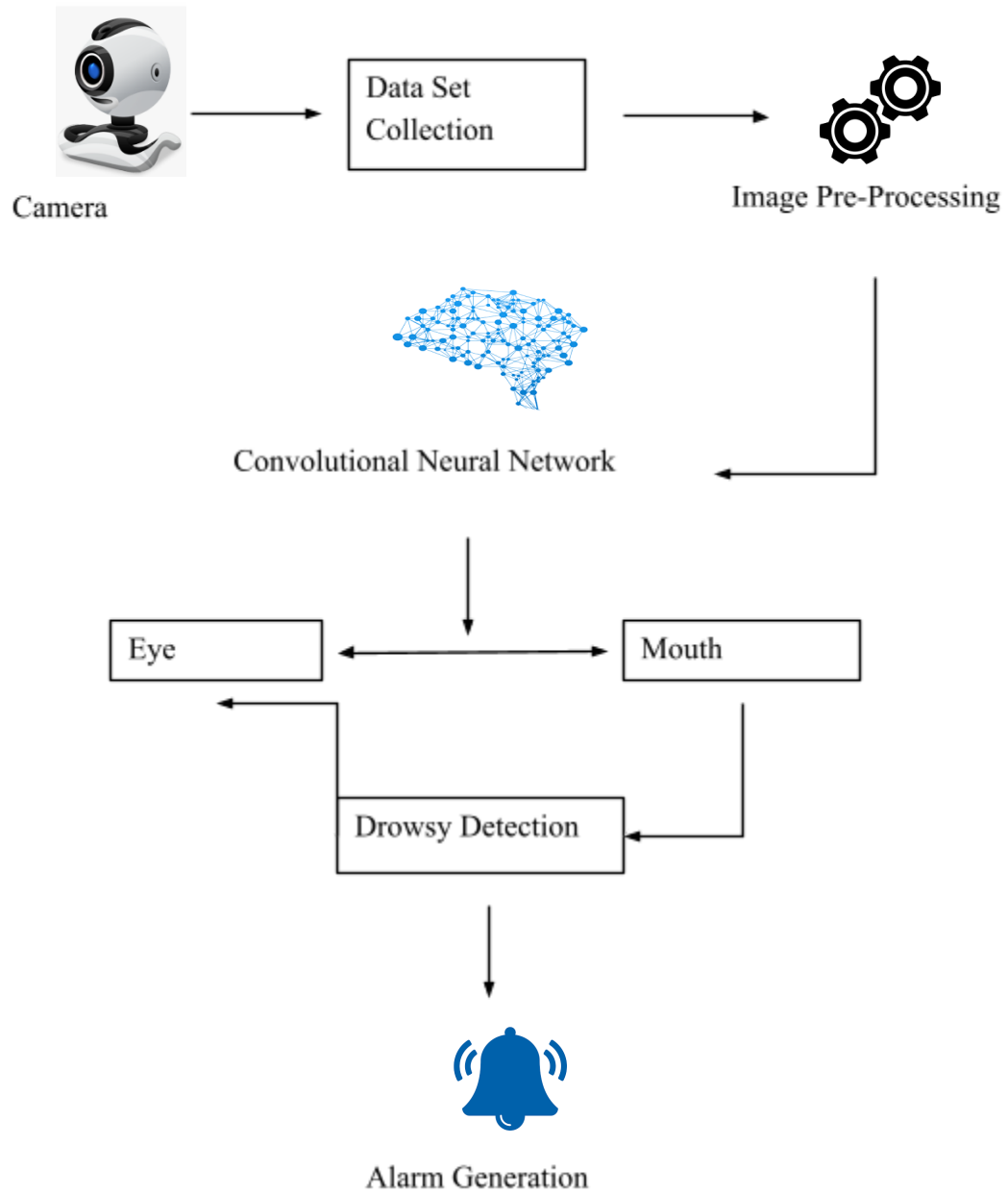
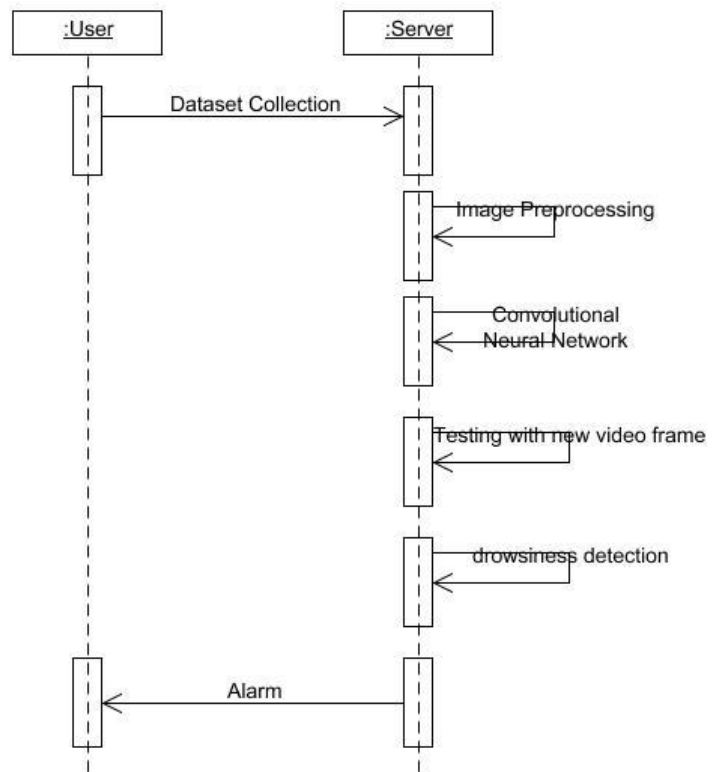


Fig: 5.1

5.2 Sequence Diagram:

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.



5.3 Use Case Diagram:

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

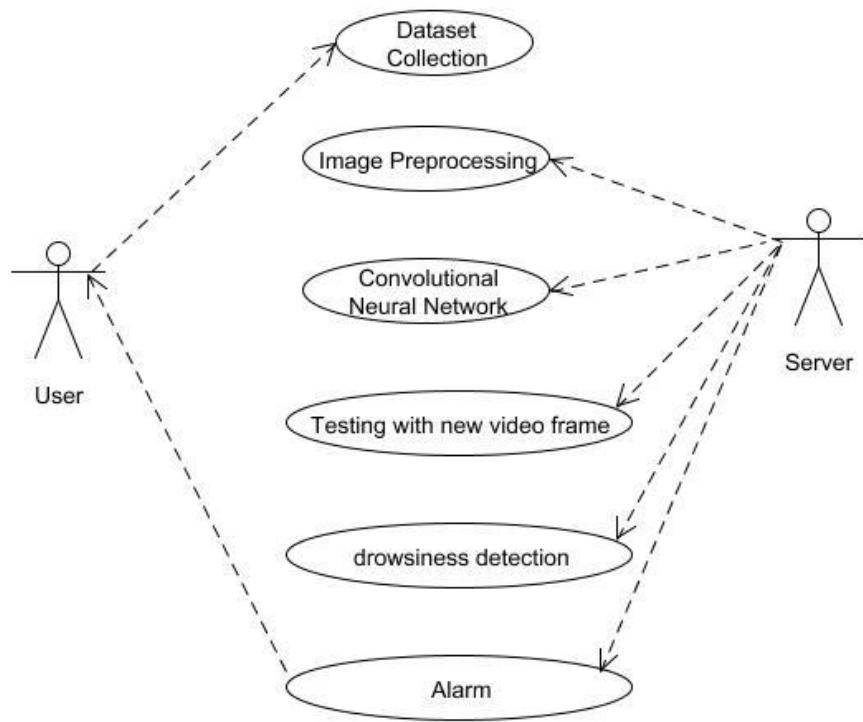
5.3.1. USECASE DIAGRAM

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

Use case: A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

Actor: An actor is a person, organization or external system that plays a role in one or more interaction with the system.

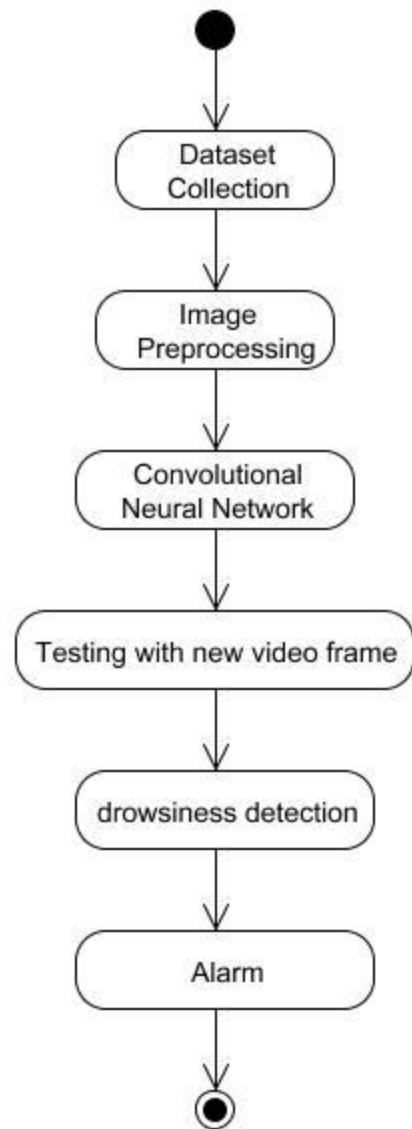


5.4 Activity Diagram:

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

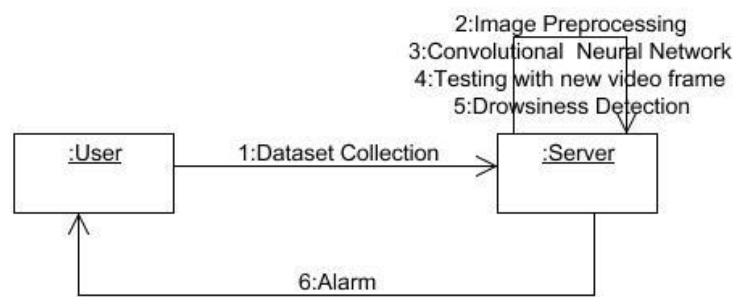
The most important shape types:

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.
- A black circle represents the start of the workflow.
- An encircled circle represents the end of the workflow.



5.5 Collaboration Diagram:

UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.



CHAPTER 6

6.1 MODULES

Video streaming using webcam

Determination of eyes and mouth regions

Evaluation of the driver's fatigue state

6.2 MODULE EXPLANATION:

6.2.1 Determination of eyes and mouth regions:

We recognize and track the driver's face in each video frame.

Evaluation of the driver's fatigue state:

We use the angle of the eye to compensate for the CNN's limitations regarding eye closure recognition. After the CNN validates that the driver's eye is open, we use the angle of the eye to validate the result. A blink is the process of the eye closing and opening

CHAPTER 7

CODING AND TESTING

7.1 CODING

Once the design aspect of the system is finalized the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

7.2 CODING STANDARDS

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Program should be simple, clear and easy to understand.

Naming conventions

Value conventions

Script and comment procedure

Message box format

Exception and error handling

7.2.1 NAMING CONVENTIONS

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

Class names

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

Member Function and Data Member name

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

7.2.2 VALUE CONVENTIONS

Value conventions ensure values for variable at any point of time. This involves the following:

Proper default values for the variables.

Proper validation of values in the field.

Proper documentation of flag values.

7.2.3 SCRIPT WRITING AND COMMENTING STANDARD

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

7.2.4 MESSAGE BOX FORMAT

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

X – User has performed illegal operation.

! – Information to the user.

7.3 TEST PROCEDURE

SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

Static analysis is used to investigate the structural properties of the Source code.

Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

7.4 TEST DATA AND OUTPUT

7.4.1 UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

7.4.2 FUNCTIONAL TESTS

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

Performance Test

Stress Test

Structure Test

7.4.3 PERFORMANCE TEST

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

7.4.4 STRESS TEST

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

7.4.5 STRUCTURED TEST

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

- Exercise all logical decisions on their true or false sides.

- Execute all loops at their boundaries and within their operational bounds.

- Exercise internal data structures to assure their validity.

- Checking attributes for their correctness.

- Handling end of file condition, I/O errors, buffer problems and textual errors in output information

7.4.6 INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The

objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

7.5 TESTING TECHNIQUES / TESTING STRATERGIES

7.5.1 TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

7.5.1.1 WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graph notation

- Cyclometric complexity

- Deriving test cases

- Graph matrices Control

7.5.1.2 BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

Graph based testing methods

Equivalence partitioning

Boundary value analysis

Comparison testing

7.5.2 SOFTWARE TESTING STRATEGIES:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

Testing begins at the module level and works “outward” toward the integration of the entire computer based system.

Different testing techniques are appropriate at different points in time.

The developer of the software and an independent test group conducts testing.

Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

7.5.2.1 INTEGRATION TESTING:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

7.5.2.2 PROGRAM TESTING:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

7.5.2.3 SECURITY TESTING:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability

from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

7.5.2.4 VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

- * The function or performance characteristics confirm to specifications and are accepted.
- * A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

7.5.2.5 USER ACCEPTANCE TESTING

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screen design.
- Output screen design.

Source Code

```
import numpy as np

import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)

while 1:

    ret, img= cap.read()

    # img = cv2.imread('DZcijliW0AE0bwc.jpg') #this line to do it with a photo
    instead of webcam

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x,y,w,h) in faces:

        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

        roi_gray = gray[y:y+h, x:x+w]
```

```
roi_color = img[y:y+h, x:x+w]
```

```
cv2.imshow('Face recognition',img)
```

```
k = cv2.waitKey(30) & 0xff
```

```
if k == 27:    #ESC key
```

```
    break
```

```
cap.release()
```

```
cv2.destroyAllWindows()
```

EYE PREPROCESSOR:

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
import os
```

```
from six.moves import cPickle as pickle
```

```
import cv2
```

```
dirs = ['Dataset/openLeftEyes', 'Dataset/openRightEyes']
```

```
dirs2 = ['Dataset/closedLeftEyes', 'Dataset/closedRightEyes']
```

```
def generate_dataset():
```

```
    dataset = np.ndarray([1231 * 2, 24, 24, 1], dtype='float32')
```

```
    i = 0
```

```
    for dir in dirs:
```

```
        for filename in os.listdir(dir):
```

```
            if filename.endswith('.jpg'):
```

```
                im = cv2.imread(dir + '/' + filename)
```

```
                    im = np.dot(np.array(im, dtype='float32'), [[0.2989], [0.5870],  
[0.1140]]) / 255
```

```
                    dataset[i, :, :, :] = im[:, :, :]
```

```
                    i += 1
```

```
    labels = np.ones([len(dataset), 1], dtype=int)
```

```
    return dataset, labels
```

```

def generate_dataset_closed():

    dataset = np.ndarray([1192 * 2, 24, 24, 1], dtype='float32')

    i = 0

    for dir in dirs2:

        for filename in os.listdir(dir):

            if filename.endswith('.jpg'):

                im = cv2.imread(dir + '/' + filename)

                im = np.dot(np.array(im, dtype='float32'), [[0.2989], [0.5870],
[0.1140]]) / 255

                dataset[i, :, :, :] = im[:, :, :]

                i += 1

    labels = np.zeros([len(dataset), 1], dtype=int)

    return dataset, labels


dataset_open, labels_open = generate_dataset()

dataset_closed, labels_closed = generate_dataset_closed()

print("done")

```

```
split = int(len(dataset_closed) * 0.8)

train_dataset_closed = dataset_closed[:split]

train_labels_closed = labels_closed[:split]

test_dataset_closed = dataset_closed[split:]

test_labels_closed = labels_closed[split:]


pickle_file = 'closed_eyes.pickle'


try:

    f = open(pickle_file, 'wb')

    save = {

        'train_dataset': train_dataset_closed,

        'train_labels': train_labels_closed,

        'test_dataset': test_dataset_closed,

        'test_labels': test_labels_closed,

    }

    pickle.dump(save, f, pickle.HIGHEST_PROTOCOL)

    f.close()
```

except Exception as e:

print ('Unable to save data to', pickle_file, ':', e)

raise

statinfo = os.stat(pickle_file)

print ('Compressed pickle size:', statinfo.st_size)

split = int(len(dataset_open) * 0.8)

train_dataset_open = dataset_open[: split]

train_labels_open = labels_open[: split]

test_dataset_open = dataset_open[split:]

test_labels_open = labels_open[split:]

pickle_file = 'open_eyes.pickle'

try:

f = open (pickle_file, 'wb')

save = {

'train_dataset': train_dataset_open,

```

        'train_labels': train_labels_open,

        'test_dataset': test_dataset_open,

        'test_labels': test_labels_open,

    }

    pickle.dump(save, f, pickle.HIGHEST_PROTOCOL)

    f.close()

except Exception as e:

    print ('Unable to save data to', pickle_file, ':', e)

    raise


statinfo = os.stat(pickle_file)

print ('Compressed pickle size:', statinfo.st_size)


EYE.CNN:

from __future__ import absolute_import

from __future__ import print_function

import numpy as np

import os

from keras.utils import plot_model

```

```

import cv2

np.random.seed(1337) # for reproducibility

from keras.models import Sequential

from keras.layers.core import Dense, Dropout, Activation, Flatten

from keras.layers.convolutional import Convolution2D, MaxPooling2D

from keras.utils import np_utils

from keras.optimizers import SGD, Adadelta, Adagrad

from six.moves import cPickle as pickle

import tensorflow as tf

import tensorflow.compat.v1 as tf

tf.disable_v2_behavior()

sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))

#Os.      environ["PATH"]      +=      os.pathsep      +      'C:/Program      Files
(x86)/Graphviz2.38/bin/'

```

```
os.environ["PATH"] += os.pathsep + 'C:/Users/gts/Downloads/bin/'
```

```
pickle_files = ['open_eyes.pickle', 'closed_eyes.pickle']
```

```
i = 0
```

```
for pickle_file in pickle_files:
```

```
    with open (pickle_file, 'rb') as f:
```

```
        save = pickle.load(f)
```

```
        if i == 0:
```

```
            train_dataset = save['train_dataset']
```

```
            train_labels = save['train_labels']
```

```
            test_dataset = save['test_dataset']
```

```
            test_labels = save['test_labels']
```

```
        else:
```

```
            print("here")
```

```
            train_dataset = np.concatenate((train_dataset, save['train_dataset']))
```

```
            train_labels = np.concatenate((train_labels, save['train_labels']))
```

```
            test_dataset = np.concatenate((test_dataset, save['test_dataset']))
```

```
            test_labels = np.concatenate((test_labels, save['test_labels']))
```

```
del save # hint to help gc free up memory
```

```
i += 1
```

```
print ('Training set', train_dataset.shape, train_labels.shape)
```

```
print ('Test set', test_dataset.shape, test_labels.shape)
```

```
batch_size = 30
```

```
nb_classes = 1
```

```
epochs = 12
```

```
X_train = train_dataset
```

```
X_train = X_train.reshape((X_train.shape[0], X_train.shape[3] +  
X_train.shape[1:3])
```

```
Y_train = train_labels
```

```
X_test = test_dataset
```

```
X_test = X_test.reshape((X_test.shape[0], X_test.shape[3] +  
X_test.shape[1:3])
```

```
Y_test = test_labels
```

Print shape of data while model is building

Print ("{1} train samples, {4} channel {0}, {2}x{3}". format (" " if X_train.shape[1] == 1 else "s", *X_train.shape))

Print ("{1} test samples, {4} channel {0}, {2}x{3}". format (" " if X_test.shape[1] == 1 else "s", *X_test.shape))

Input image dimensions

_, img_channels, img_rows, img_cols = X_train.shape

Convert class vectors to binary class matrices

Y_train = np_utils.to_categorical(y_train, nb_classes)

Y_test = np_utils.to_categorical(y_test, nb_classes)

model = Sequential ()

model.add(Convolution2D(32, (3, 3), padding='same',

input_shape= (img_channels, img_rows, img_cols),data_format='channels_first'))

model.add(Activation('relu'))

```
model.add(Convolution2D(24, (3, 3), data_format='channels_first'),)
```

```
model.add(Activation('relu'))
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Dropout(0.25))
```

```
model.add(Convolution2D(64, (3, 3), padding='same',  
data_format='channels_first'))
```

```
model.add(Activation('relu'))
```

```
model.add(Convolution2D(64, (3, 3)))
```

```
model.add(Activation('relu'))
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Dropout(0.25))
```

```
model.add(Flatten())
```

```
model.add(Dense(512))
```

```
model.add(Activation('relu'))
```

```
model.add(Dropout(0.5))
```

```
model.add(Dense(nb_classes))
```

```
model.add(Activation('sigmoid'))
```

let's train the model using SGD + momentum (how original).

sgd = SGD (lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

**model.compile(loss='binary_crossentropy', optimizer=sgd,
metrics=['accuracy'])**

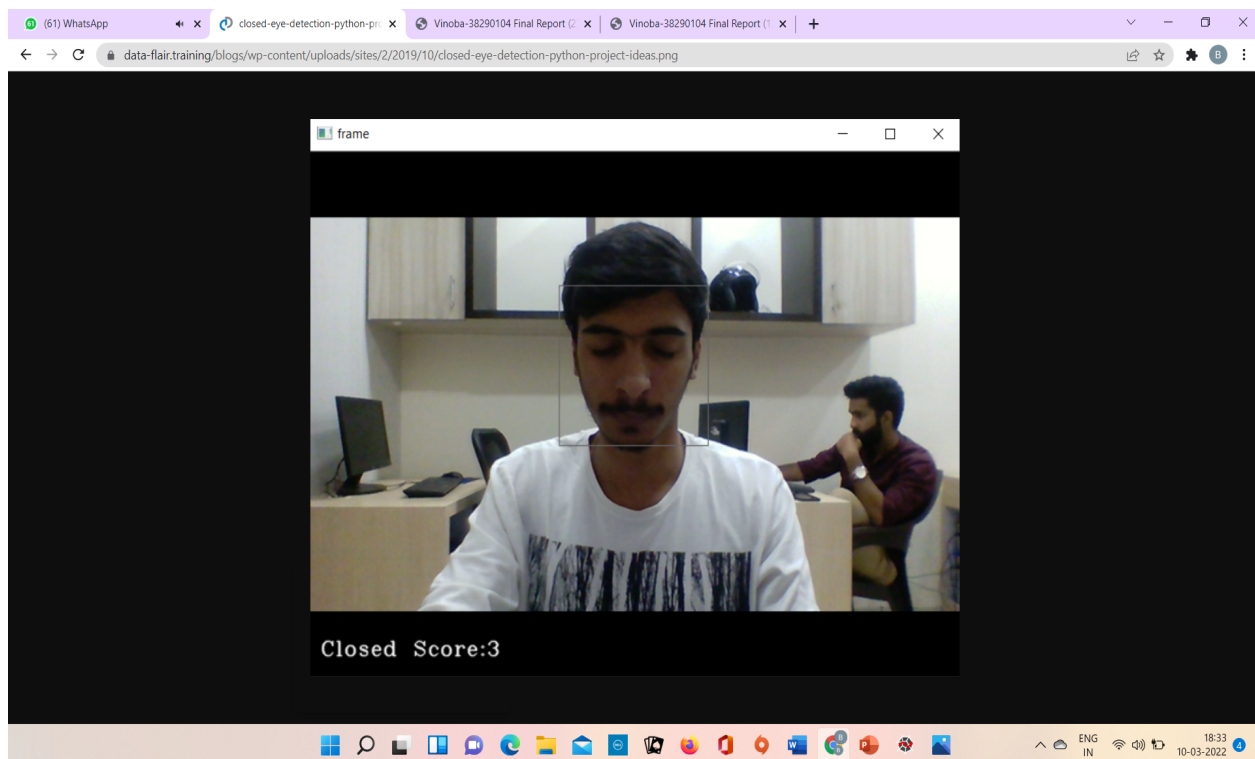
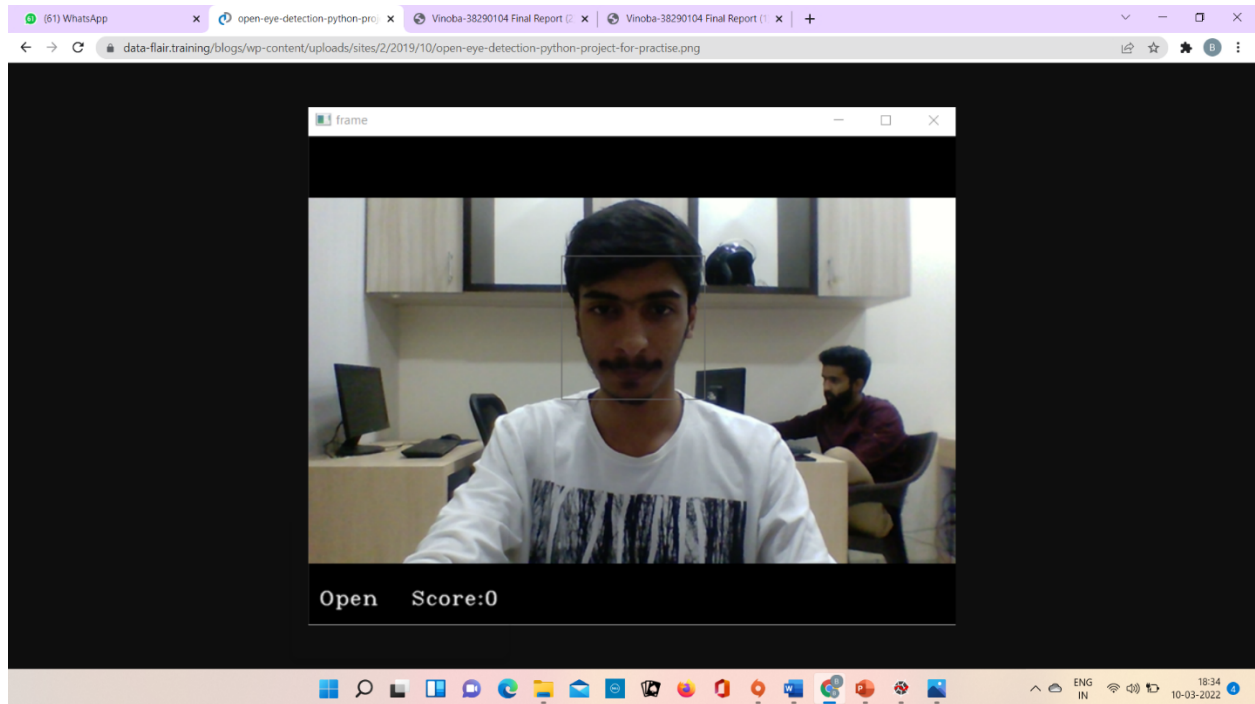
**model.fit(X_train, Y_train, batch_size=batch_size, epochs=epochs, verbose=2,
validation_data=(X_test, Y_test))**

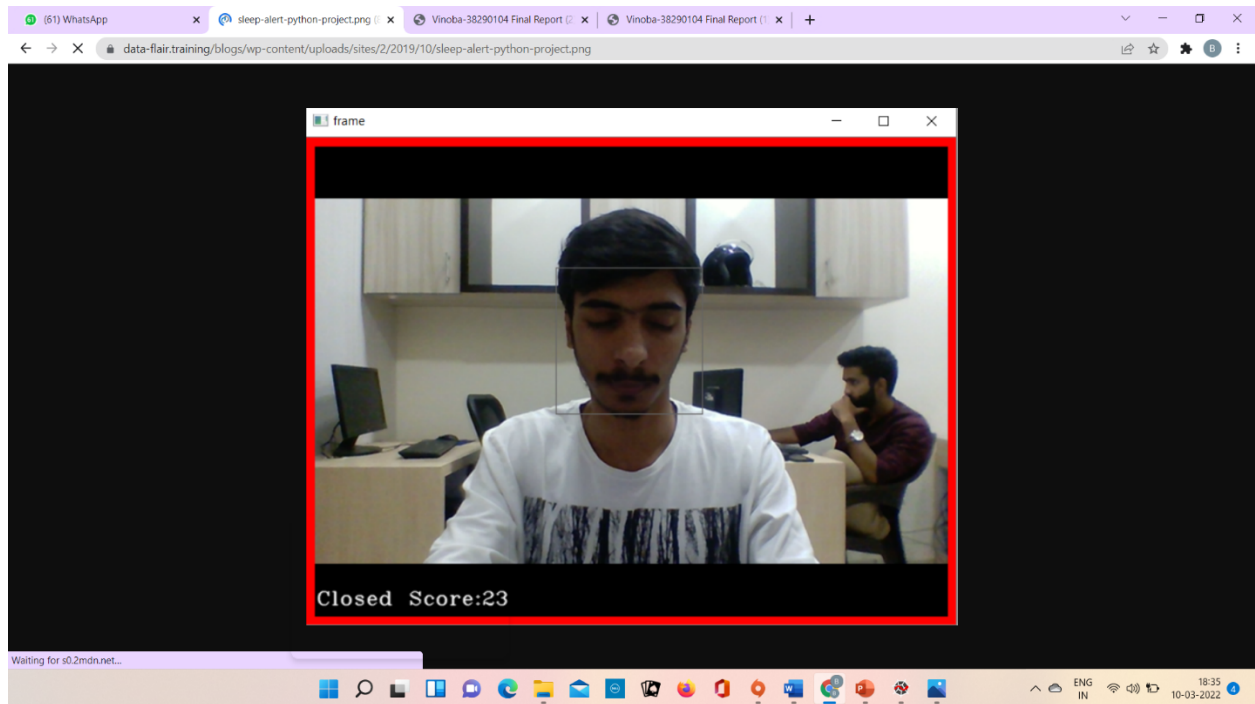
score = model.evaluate(X_test, Y_test, verbose=1)

print ('Test score:', score [0])

print ('Test accuracy:', score [1])

Screenshots:





REFERENCES

- [1] Dr. Priya Gupta, Nidhi Saxena, Meetika Sharma, Jagriti Tripathi, Deep Neural Network for Human Face Recognition International Journal of Engineering and Manufacturing, vol.8, no.1, pp. 63-71. January 2018.
- [2] Jeyasekar A, Vivek Ravi Iyengar, Based on Behavioural Changes using ResNet, International Journal of Recent Technology and Engineering (IJRTE), vol. 8, no. 3, pp. 25-30, 2019.
- [3] Conference (IACC), Gurgaon, pp. 995-999, 2014.
- [4] Ki Wan Kim, Hyung Gil Hong, Gi Pyo Nam and Kang Ryoung Park, -
- [5] Luigi Celona, Lorenzo Mammana, Simone Bianco, Raimondo Schettini, -Task CNN Framework for Driver Face Berlin, 2018.
- [6] Mandalapu Sarada Devi and Dr. Preeti R Bajaj, Detection Based on Eye Tracking, First International Conference on Emerging Trends in Engineering and Technology, vol.1, pp. 649-652, 2008.
- [7] Sanghyuk Park, Fei Pan, Sunghun Kang and Chang D. Yoo, Driver drowsiness detection system based on feature representation learning Springer International Publishing, Computer Vision ACCV 2016 Workshops.
- [8] Tawsin Uddin Ahmed, Sazzad Hossain, Mohammad Shahadat Hossain, Raihan Ul Islam, Karl Andersson, Facial Expression Recognition using Convolutional Neural Network with Data th International Conference on Informatics, Electronics & Vision (ICIEV), Washington, USA, 2019,
- [9] Upasana Sinha, Kamal K. Mehta, A.K. Shrivastava, Real Time Implementation for Monitoring Drowsiness Condition of a Train Driver using Brain Wave Sensor, International Journal of Computer Applications, vol. 139, no.9, pp. 25-30, 2016.
- [10] Xiaoxia Ma, Lap-Pui Chau and Kim- -based Twostream Convolutional Neural Networks for Driver Fatigue IEEE International Conference on Orange Technologies

(ICOT), pp. 155-158, 2017.

[11] Weiwei Zhang, Jinya Su Driver Yawning Detection based on Long
, IEEE Symposium Series on
Computational Intelligence (SSCI), Honolulu, HI, USA, 2017.

[12] Zhongke Gao, Xinmin Wang, Yuquan Yang, Chaoxu Mu , Qing Cai,
Weidong Dang, and Siyang Zuo EEG-Based Spatial-Temporal
, IEEE
Transactions on Neural networks and learning systems, vol. 30, no. 9,
pp. 2755-2763, 2019.
2021 International Conference on Computing, Communication, and Intelligent Systems
(ICCCIS)