CHAPTER 1

INTRODUCTION

1.1 Introduction

Driving a car is a complex task, and it requires complete attention. Distracted driving is any activity that takes away the driver's attention from the road. Several studies have identified three main types of distraction: visual distractions (driver's eyes off the road), manual distractions (driver's hands off the wheel) and cognitive distractions (driver's mind off the driving task). The National Highway Traffic Safety Administration (NHTSA) reported that 36,750 people died in motor vehicle crashes in 2018, and 12% of it was due to distracted driving. Texting is the most alarming distraction. Sending or reading a text takes your eyes off the road for 5 seconds. At 55 mph, that's like driving the length of an entire football field with your eyes closed. Many states now have laws against texting, talking on a cell phone, and other distractions while driving. We believe that computer vision can augment the efforts of the governments to prevent accidents caused by distracted driving. Our project automatically detects the distracted activity of the drivers and alerts them. We envision this type of product being embedded in cars to prevent accidents due to distracted driving.

1.2 Scope of the project

This proposed model provides the important information with the highest accuracy. According to the report compiled by the ministry Transport Research Wing [1], there has been a 3.2% rise in road fatalities which corresponds to the death of 1,50,785 people across the country in 2016. The number of road accidents in 2016 and 2015 was 4,80,652 and 5,01,000 respectively. Our project aims to mitigate the above problem by correctly identifying whether the driver is distracted or not. The software, if integrated with hardware can warn the driver if he gets distracted and thus prevent an accident from happening. We input images of the driver to our model. Each image belongs to one of the 10 classes mentioned in the dataset section. The model then predicts the class of an image by giving as an output a probability for each class.

1.3Objectives

The objective of our project is to analyze dataset which consists of numerous classes of distraction caused to the driver and predicting the type of distraction he has undergone while driving a car. It can be implemented in the car to analyze, predict and alert the driver in case of any distraction caused while driving. Using the concept of Machine Learning we can extract previously known, useful information from a structured data

1.4 Organization of the report

➢ Chapter-1: Introduction

This chapter describes the organization of the report, objectives of the proposed model and brief introduction of our proposed model.

➢ Chapter-2: Literature Survey

This chapter mainly deals with all the observation, which is conducted as initial study before the actual development of the project. It also describes the details regarding existing system and its disadvantages

➢ Chapter-3: System Requirement Specification

This chapter speaks about the product perspective, user characteristics, its assumptions and dependencies, specific requirements, functionality along with resource requirements

➢ Chapter-4: System Design

This chapter deals with the advance software engineering where the entire flow of the project is represented by data flow diagram and the architecture of the project.

➢ Chapter-5: Gantt Chart

A Gantt chart is a type of bar chart, developed by Henry Gantt that illustrates a project schedule. Gantt charts illustrate the start and finish of the terminal elements and summary elements of the project..

➢ Chapter-6: Implementation

This chapter deals with the steps involved in the creation of the project work. It is defined with the assistant of pseudo code explanation for the ease of reader.

➢ Chapter-7: Testing

This chapter deals with various test cases and test levels such as unit testing, system testing, integration testing and acceptance testing.

- **Chapter-8: Results and discussion**

  This chapter mainly deals with the graphical user interface of the project to show the output of the application and also emphasize on the performance comparisons.

- **Chapter-9: Future work**

  These sections also suggest some of the enhancement idea which couldn't be covered up due to constraint of time and resources.

- **Chapter-10: Conclusion**

  These sections are mainly the summary of the entire project development.

CHAPTER 2

LITERATURE SURVEY

LITERATURE REVIEW

A literature survey shows the various analysis and research made in the field of interest and results already published, taking into account the various parameters of the project and the extent of the project. It includes researches made by various analysts-their methodology and the conclusion they have arrived at. It is the most important part of the report as it gives the direction in the area of research.

2.1 BACKGROUND RESEARCH

Driver inattention and distraction are the main causes of road accidents, many of which result in fatalities. To reduce road accidents, the development of information systems to detect driver inattention and distraction is essential. Currently, distraction detection systems for road vehicles are not yet widely available or are limited to specific causes of driver inattention such as driver fatigue. Despite the increasing automation of driving due to the availability of increasingly sophisticated assistance systems, the human driver will continue to play a longer role as supervisor of vehicle automation.

Cell Phone Usage Detection

[1] presents an SVM-based model that detects the use of mobile phone while driving (i.e. distracted driving). Their dataset consists of frontal image view of a driver's face. They also make pre-made assumptions about hand and face locations in the picture. [2] presents another SVM-based classification method to detect cell phone usage. However, their dataset is collected from transportation imaging cameras that are deployed in highways and traffic lights which is, indeed, more competitive. [3] uses AdaBoost classifier and Hidden Markov Models to classify a Kinect's RGB-D data. Their solution depends on indoor-produced data. They sit on a chair and a mimic a certain distraction (i.e. talking on the phone). This setup misses two essential points: the lighting conditions and the distance between a Kinect and the driver. In real-life applications, a driver is exposed to a variety of lighting conditions (i.e. sunlight and shadow). [4] suggests using a Hidden Conditional Random Fields (HCRF) model to detect cell phone usage. Their model operates face, mouth, and hand features of images obtained from a camera mounted above the dashboard. [5] devised a Faster-RCNN model to detect driver's cell-phone usage and "hands on the wheel". Their model is mainly geared towards face/hand segmentation. They train their Faster-RCN Non the dataset proposed in [6] (that we also use in this paper). Their proposed

solution runs at a 0.06, and 0.09 frames per second for cell-phone usage, and "hands on the wheel" detection. [7] tackles the problem of cell phone usage detection. Their approach doesn't hold any static assumptions though (i.e. in which region of the image a face is expected to be found). They use a Supervised Descent Method (SDM) to localize the face landmarks, and then, extract two bounding boxes to the left and the right side of the face. They train a classifier on each of the two regions to detect cell phone usage: right hand, left hand, or no usage. Using a histogram of gradients (HOG) and an AdaBoost classifier, they achieve a 93.9% classification accuracy and operate in a near real-time speed (7.5 frames per second).

UCSD's Laboratory of Intelligent and Safe Automobiles Work

[8] presents an vision-based analysis framework that recognizes in-vehicle activities using two Kinect cameras that provide frontal and back views of the driver. Their approach provides "hands on the wheel" information (i.e. left hand only, both hands, no hands), and uses these information to detect three types of distractions: adjusting the radio, operating the gear, and adjusting the mirrors. [9] presents a fusion of classifiers where the image is to be segmented into three regions: wheel, gear, and instrument panel(i.e. radio). It proposes a classifier for each segment to detect existence of hands in those regions. The hand information (i.e. output of the classifiers) is passed to an "activity classifier" that infers the actual activity (i.e. adjusting the radio, operating the gear). [10] extends existing research to include eye cues to previously existing head and hands cues. However, it still considers three types of distractions: "wheel region interaction with two hands on the wheel, gear region activity, and instrument cluster region activity". [11] presents a region-based classification approach. It detects hands presence in certain pre-defined regions in an image. A model is learned for each region separately. All regions are later joined using a second-stage classifier.

South east University Distracted Driver Dataset

[12] designs a more inclusive distracted driving dataset with a side view of the driver and more activities: grasping the steering wheel, operating the shift lever, eating a cake and talking on a cellular phone. It introduces a contourlet transform for feature extraction, and then, evaluates the performance of different classifiers: Random Forests (RF), k-Nearest Neighbors (KNN), and Multilayer Perceptron (MLP). The random forests achieved the highest classification accuracy of 90.5%. [13] showed that using a multi wavelet transform improves the accuracy of Multilayer Perceptron classifier to 90.61% (previously 37.06% in [12]). [14] showed that using a Support Vector Machine (SVM) with an intersection kernel, followed by Radial Basis Function (RBF) kernel, achieved the highest accuracies of 92.81% and 94.25%, respectively (in comparison with [12] and [13]). After testing against other classification methods, they concluded that an SVM with intersection kernel offers the best real-time

quality (67 frames per second) and better classification performance. [15] improves the Multilayer Perceptron classifier using combined features of Pyramid Histogram of Oriented Gradients (PHOG) and spatial scale feature extractors. Their Multilayer Perceptron achieves a 94.75% classification accuracy.

2.2 Existing System

- The existing distracted driver detection system can automatically predict the distraction related to drowsiness of the driver based on his eye movement.
- A time limit can be set for the system to check the drowsiness of the driver.
- It gives a alert to driver whenever he is trying to sleep while driving which makes him to focus on driving.
- This helps in reducing the accidents on the road.

2.3 Limitations of existing systems

Some of the common flaws noticed in application are:

- Now a days there are drowsy driver detection which detects the drowsiness of the driver based on the eyes.
- But there are many more distractions to the driver such as drinking, talking on the phone, reaching behind etc
- So Drowsy Driver Detection system will not classify other distraction that causes to the driver.

2.4 PROPOSED SYSTEM

Among various network architectures used in deep learning, convolutional neural networks (CNN) are widely used in image recognition. CNNs consist of convolutional layers, which are sets of image filters convoluted to images or feature maps, along with other (e.g., pooling) layers. In image classification, feature maps are extracted through convolution and other processing layers repetitively and the network eventually outputs a label indicating an estimated class.

The complete process is divided into several necessary stages, starting with gathering images for classification process using neural networks.

- **Image Preprocessing**: Images downloaded from the TensorFlow were in various formats along with different resolutions and quality. Pre processing our raw data into usable format. Rescaling image values between (0-1) called Normalization. It is to change the values of numeric columns in the dataset to a common scale,without distorting differences in the range

of values.

- **Augmentation Process**: The main purpose of applying augmentation is to increase the dataset and introduce slight distortion to the images which helps in reducing overfitting during the training stage.The process of augmentation was chosen to fit the needs the leaves in a natural environment could vary in visual perspective.

Building the model:

- **Convolution Layer**: Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. ReLU stands for Rectified Linear Unit for a non-linear operation. The output is $f(x) = \max(0,x)$.ReLU's purpose is to introduce non- linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values. The activation function is an integral part of a neural network. Without an activation function, a neural network is a simple linear regression model. This means the activation function gives non-linearity to the neural network.

- **Pooling Layer**: Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

- **Fully Connected Layer**: The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network.

- **2D Convolution Layer**: The most common type of convolution that is used is the 2D convolution layer, and is usually abbreviated as conv2D. A filter or a kernel in a conv2D layer has a height and a width. They are generally smaller than the input image and so we move them across the whole image. The area where the filter is on the image is called the receptive field. Conv2D filters extend through the three channels in an image (Red, Green, and Blue). The filters may be different for each channel too. After the convolutions are performed individually for each channels, they are added up to get the final convoluted image. The output of a filter after a

convolution operation is called a feature map.Each filter in this layer is randomly initialized to some distribution (Normal, Gaussian, etc.). By having different initialization criteria, each filter gets trained slightly differently. They eventually learn to detect different features in the image.If they were all initialized similarly, then the

chances of two filters learning similar features increase dramatically. Random initialization ensures that each filter learns to identify different features.Since each conv2D filter learns a separate feature, we use many of them in a single layer to identify different features. The best part is that every filter is learnt automatically.Each of these filters are used as inputs to the next layer in the neural network.Conv2D filters are used only in the initial layers of a Convolutional Neural Network.

Fig 2.4 CNN Block diagram

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

A system requirements specification is the official statement of what the system developers should implement. It should include both the user requirements for a system and a detailed specification of the system requirements. A requirement specification for a system is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describes all the interactions the users will have with the system. In addition to the use cases the SRS will also have non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards or design constraints). Requirement specification adds more information to requirement definition. The different desirable characteristics of a SRS are understandable, unambiguous, completeness, verifiable, consistent, modifiable and traceable. This also provides a reference for validation of the final product.

3.1 Overall Description

The SRS is a document which describes what the software will do and how it will be expected to perform.

3.2 Specific Requirements

This section includes the detailed description about the hardware requirements, software requirement, functional requirement and non-functional requirements.

3.2.1 Hardware Requirements

Hardware requirements refer to the physical parts of a computer and related devices. Internal Hardware devices include motherboards, hard drives and RAM. External hardware devices include monitors, keyboards, mice, printers, and scanners.

- Processor with 2GHz frequency or above.

- A minimum of 8GB of RAM.
- A minimum of 20GB of available disk space.
- A Web Cam

3.6.2 Software Requirements

Software requirement is a field within Software Engineering that deals with establishing the needs of stakeholders that are to be solved by the software. The software requirement of our project are given below:

- Front end tool: OpenCV.
- Operating System : Windows 10 or any 64 bit platform and 8gb RAM.
- Back end tool: Visual Studio Code.
- Scripting language: python.

Python:

Python is powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation. The Python interpreter is easily extended with new functions and data types implemented in C or C++.

Tensorflow:

Initially released in 2015, Tensor Flow is an open source machine learning frame work that is easy to use and deploy across a variety of platforms. It is one of the most well- maintained and extensively used frameworks for machine learning. The main Features are it helps in building and training your models and it helps in neural network a full cycle deep learning system it is used to train as well as build ML models effortlessly using high-level APIs like Keras with eager execution. This is an open source software and highly flexible. Efficiently deploy and train the model in the cloud.

Keras:

Keras is an open source neural network library written in Python that runs on top of Theano or TensorFlow. It is designed to be modular, fast and easy to use. It was developed by Francois Chollet, a Google engineer. Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend". So Keras is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano. Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function.Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

OpenCV:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-

the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

NumPy:

NumPy is a great general purpose array processing package. It provides a high performance multidimensional array object, and tools for working with these arrays.It is the fundamental package for scientific computing with python.It contains various features including these are the important ones:

- A powerful N-dimensional array object
- Sophisticated(broadcasting) functions.
- Tools for integrating C/C++ and Fortran Code.
- Useful Linear algebra, Fourier transformation and random number capabalities.

Besides its obvious scientific use, NumPy can also be used as an efficient multi-dimensional container of the generic data. Arbitrary data-types can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

CHAPTER 4

SYSTEM DESIGN

The software application design is a difficult entity. The improvement of the application or a system follows the Software Development Life Cycle (SDLC). The next step after analysis of the entire requirement is design. The main purpose of system design is to attain the entire design of the system, also to catch the idea about the different modules present in the system, relation between the different modules, to know about the purpose of each module, and how all the modules merge together to form a complete system.

4.1 Architectural design

An architecture diagram is a graphical representation of a set of concepts, that are part of architecture, including their principles, elements and components. Sometimes, it's helpful to get a big picture view of something for context and an architectural diagram.An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system components or building blocks and provides a plan from which products can be procured, and system developed , that will work together to implement the overall system.

Architectural diagram help in:

- They help with comprehension: architectural diagrams help convey complex information in a single image like there is a saying a picture is worth a thousand words.
- They improve communication and collaboration: when we are working on anything that involves multiple people, there is always a risk of miscommunication between them .so it is important to standardize information, which is where the architectural diagrams are very useful.

The architectural design gives the description about how the overall system is designed. It is specified by identifying the components defining the control and data flow between them. The arrows indicate the connection and the rectangular boxes represent the functional units. The fig. 5.1 shows the Architectural Design Diagram of this project. This system consists of various modules that are used to develop the application.

Figure 4.1.1 Architectural design for the system

4.2 Data flow diagram

A data flow diagram is a graphical representation of the "flow" of data through an information system, modeling its process aspects. It illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it gets stored. The below figure shows the data flow diagram.

Figure 4.2.1 Data Flow Diagram

4.3 Use case diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.Elements of use case diagram are:

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

• Actors: An actor is a person, organization, or external system that plays a role in more interactions with the system.

The use cases are represented by either circles or ellipses. Fig. 5.2 shows a use case diagram of the project.System consists of different use cases .User interacts with few of the use cases and admin also interacts with the use cases.

Figure 4.3.1 Use Case diagram

In fig 5.2, the image will be uploaded by the application and system will pre process the data to get the required features and it is deployed into CNN model. CNN model will train the data to get the predicted output.

4.4 Activity Diagram

Activity diagram is an important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

In fig 5.6, The user will input the image through the application,where in it pre process the image to get desired feature.Then testing of that image is done using trained CNN model to classify the distraction or safe driving.

Figure 4.4.1 Activity Diagram

CHAPTER 5

GANTT CHART

A Gantt chart is a type of bar chart, developed by Henry Gantt that illustrates a project schedule. Gantt charts illustrate the start and finish of the terminal elements and summary elements of the project. Terminal elements and summary elements comprise the work breakdown structure of the project.

The following is the Gantt chart of the project "Distracted Driver Detection"

Activity/Month	Oct 2020	Nov 2020	Dec 2020	Jan 2021	Feb 2021	Mar 2021	Apr 2021	May 2021
Synopsis								
Presentation on Idea								
SRS								
Design								
Implementation								
Testing								
Report								

CHAPTER 6

SYSTEM IMPLEMENTATION

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, algorithm or policy. Implementation refers to conversion of a new system design to an operation. An implementation plan is made before starting the actual implementation of the system. The new system may be totally new, replacing an existing manual or automated system or it may be a major modification to an existing system.

6.1 Code Snippets

6.1.1 Code Snippet – Pre Processing and Data Augmentation

```
def create_training_data(self):
    for category in utils.CATEGORIES:
        path = os.path.join(utils.TRAIN_DATADIR,category)
        class_num = utils.CATEGORIES.index(category)
        for img in tqdm(os.listdir(path)):
            try:
                img_array = cv2.imread(os.path.join(path,img) ,cv2.IMREAD_COLOR)
                new_array = cv2.resize(img_array, (utils.IMG_SIZE, utils.IMG_SIZE))
                self.train_npy.append([np.array(new_array),
                self.create_label(utils.CATEGORIES.index(category))])
            except Exception as e:
                pass
    random.shuffle(self.train_npy)
    np.save(utils.TRAIN_DATA_COLOR_NPY, self.train_npy)
    return utils.TRAIN_DATA_COLOR_NPY
```

6.1.2 Code Snippet – Building CNN Model

```
def tflearn_definition(self):
        ops.reset_default_graph()
        convnet = input_data(shape=[None, utils.IMG_SIZE, utils.IMG_SIZE, 3].
name='input')
        convnet = conv_2d(convnet, 32, 3, activation='relu')
        convnet = max_pool_2d(convnet, 2)
        convnet = conv_2d(convnet, 64, 3, activation='relu')
        convnet = max_pool_2d(convnet, 2)
        convnet = conv_2d(convnet, 128, 3, activation='relu')
        convnet = max_pool_2d(convnet, 2)
        convnet = conv_2d(convnet, 64, 3, activation='relu')
        convnet = max_pool_2d(convnet, 2)
        convnet = conv_2d(convnet, 32, 3, activation='relu')
        convnet = max_pool_2d(convnet, 2)
        convnet = fully_connected(convnet, 1024, activation='relu')
        convnet = dropout(convnet, 0.6)
        convnet = fully_connected(convnet, 512, activation='relu')
        convnet = dropout(convnet, 0.6)
```

6.1.3 Code Snippet – Applying Regression

```
convnet = fully_connected(convnet, 10, activation='softmax')

convnet = regression(convnet, optimizer='adam', learning_rate= utils.LR,

        loss='categorical_crossentropy', name='targets')
```

6.1.4 Code Snippet – Training The Model

```
def create_model(self , train_data ):
    train_data = np.load(train_data, encoding="latin1",allow_pickle='true')
    train = train_data[:-7000]
    validation = train_data[-7000:-1000]
    x_train = np.array([i[0] for i in train]).reshape(-1, utils.IMG_SIZE, utils.IMG_SIZE, 3)
    y_train = [i[1] for i in train]
    x_validation = np.array([i[0] for i in validation]).reshape(-1, utils.IMG_SIZE,
utils.IMG_SIZE, 3)
    y_validation = [i[1] for i in validation]
    convnet = self.tflearn_definition()
    model = tflearn.DNN(convnet, tensorboard_dir='./models/log', tensorboard_verbose=0)
    if os.path.isfile("./models/cnn.model.meta"):
        model.load('./models/cnn.model')
    else:
        model.fit({'input': x_train}, {'targets': y_train}, n_epoch=10,
            validation_set=({'input': x_validation}, {'targets': y_validation}),
            snapshot_step=500, show_metric=True, run_id=utils.MODEL_NAME)
        model.save('./models/cnn.model')
    return model
```

6.1.5 Code Snippet – Testing Unknown Image

```
def test(self):
    i = 0
    for img in tqdm(os.listdir(utils.TEST_UNKNOWN)):
        try:
            img_array = cv2.imread(os.path.join(utils.TEST_UNKNOWN,img)
,cv2.IMREAD_COLOR)
            output = self.predict_output(img_array)
            cv2.putText(img_array,output,(10,100), self.font,1,(255,0,0),2,cv2.LINE_AA)
            cv2.imwrite( "./output_test/output_image_{}".format(i)+".jpg", img_array)
            i += 1
        except Exception as e:
            print("Exception Occured : " + str(e))
    print("prediction is DONE and output is saved in output folder.....")
def predict_output(self, image):
    image = cv2.resize(image, (utils.IMG_SIZE, utils.IMG_SIZE))
    image = np.array(image)
    image = image.reshape(utils.IMG_SIZE, utils.IMG_SIZE, 3)
    model_out = self.model.predict([image])
    prediction = utils.image_class[np.argmax(model_out)]
    return prediction
```

6.1.6 Code Snippet – Confusion Matrix

```
def confusion_matrix_fn( model):
    train_data_npy = np.load(utils.TRAIN_DATA_COLOR_NPY,
encoding="latin1",allow_pickle='true')
    test_data = train_data_npy[-1000:]
    total = 0
    real = 0
    confusion_matrix =  [ [0] * 10 for _ in range(10)]
    for num, data in enumerate(test_data[:]):
        img_num = np.argmax(data[1])
        img_data = data[0]
        orig = img_data
        data = img_data.reshape(utils.IMG_SIZE, utils.IMG_SIZE, 3)
        model_out = model.predict([data])[0]
        confusion_matrix[img_num][np.argmax(model_out)] += 1
        if(np.argmax(model_out) == img_num):
            real +=1
        total +=1
    print("correct pics : " , real, "total pics : ",total,"accuracy : ",float(real*100)/total)
    for i in range(10):
        print(confusion_matrix[i])
    class_wise_acc = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0]
    for i in range(10):
        sum = 0
        for j in range(10):
            sum += confusion_matrix[i][j]
        class_wise_acc[i] =   ( float(confusion_matrix[i][i] )/ sum ) * 100
```

```
fig, ax = plt.subplots(figsize=(4, 4))
plt.ylabel('Accuracy ( % )')
plt.xlabel('Classes')
plt.bar(np.arange(10), class_wise_acc)
plt.xticks(np.arange(10), ('c0', 'c1', 'c2', 'c3','c4', 'c5', 'c6', 'c7', 'c8', 'c9'))
fig.savefig('graphs/clsacc.png')
plt.gcf().clear()
df_cm = pd.DataFrame(confusion_matrix)
svm = sn.heatmap(df_cm, annot=False,cmap='coolwarm', linecolor='white', linewidths=1)
plt.savefig('graphs/conmtrx.png')
print("Confusion Matrix Done .....")
return confusion_matrix , class_wise_acc
```

CHAPTER 7

SYSTEM TESTING

Testing is an important phase in the development life cycle of the product. During the testing, the program to be tested was executed with a set of test cases and the output of the program for the test cases was evaluated to determine whether the program is performing as expected. Errors were found and corrected by using the following testing steps and correction was recorded for future references. Thus, a series of testing was performed on the system before it was ready for implementation. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

7.1 Testing Levels

Testing is part of Verification and Validation. Testing plays a very critical role for quality assurance and for ensuring the reliability of the software.

The objective of testing can be stated in the following ways.

- A successful test is one that uncovers as-yet-undiscovered bugs.
- A better test case has high probability of finding un-noticed bugs.
- A pessimistic approach of running the software with the intent of finding errors. Testing can be performed in various levels like unit test, integration test and system test.

7.1.1 Unit Testing

Unit testing tests the individual components to ensure that they operate correctly. Each component is tested independently, without other system component. This system was tested with the set of proper test data for each module and the results were checked with the expected output. Unit testing focuses on verification effort on the smallest unit of the software design module.

7.1.2 Integration Testing

Integration testing is another aspect of testing that is generally done in order to uncover errors associated with the flow of data across interfaces. The unit-tested modules are grouped together and tested in small segment, which makes it easier to isolate and correct errors. This approach is continued until we have integrated all modules to form the system as a whole.

7.1.3 System Testing

System testing tests a completely integrated system to verify that it meets its requirements.After the completion of the entire module they are combined together to test whether the entire project is working properly. It deals with testing the whole project for its intended purpose. In other words, the whole system is tested here.

7.1.4 Acceptance Testing

Project is tested at different levels to ensure that it is working properly and was meeting the requirements which are specified in the requirement analysis. The complete system is accepted which meets all the mentioned functional and nonfunctional requirements specified.

7.2 Test Cases

A test case is a software testing document, which consists of events, action, input, output, expected result and actual result. Technically a test case includes test description, procedure, expected result and remarks. Test cases should be based primarily on the software requirements and developed to verify correct functionality and to establish conditions that reveal potential errors.

Individual PASS/FAIL criteria are written for each test case. All the tests need to get a PASS result for proper working of an application.

Test Case 1:

Objective: Pre process the input image

Steps: The following steps have to be followed to carry out the test.

1. Normalization of input image.
2. Augmented data to train the test directories.

Expected Results: To get modified version of the image.

Result: Successful.

Test Case 2:

Objective: Send the video from file to model

Steps: The following steps have to be followed to carry out the test.

1. Check for the path of the file
2. Augment the video as per requirements and send it to model

Expected Results: To predict the contents in the video

Result: Successful

Test Numbers	Test Case	Expected results	Status
1	Pre process the input image	To get modified version of the image	Successful
2	Send the video from file to the model	To predict the contents in the video	Successful

Table 7.1 Test Cases

CHAPTER 8

RESULTS AND DISCUSSIONS

8.1 Home Application

The Following image depicts the home page of our application.

8.2 Video Input

The following image depicts to select a video for predicting it after clicking on Video Input Button

8.3 Video Output

This image depicts the predicted video done by our model.

8.4 Test on Unknown Data

This image depicts the testing of unknown images inserted to predict the classes of distraction when clicked on Test On Unknown button

8.5 Accuracy

This image depicts the accuracy and performance of the application using graph and confusion matrix

CHAPTER 9

SCOPE FOR FUTURE WORK

Currently, the similarities in postures of different behaviours result in incorrect classifications. The behaviours that have more misclassifications are ‘drinking’, ‘hair and makeup’ and ‘texting on the phone – left’. In the future, we need to solve this misclassification issue. To improve the accuracy, we can also combine our approach with the face-based approach or replace the last FC layers by the traditional classifiers such as SVM or HMM to classify the distracted behaviours. In addition, we can enhance the distraction detection system by using other sensing modalities. For example, we can use microphones to capture the sound and voice in the car, which offers rich clues to detect different distracted driving behaviours. Moreover, we can also enhance the models by using dynamic data.

CHAPTER 10

CONCLUSION

Driver distraction is a serious problem leading to large number of road crashes worldwide. Hence the main motive of our project is to reduce the amount of accidents caused due to driver distractions, by providing information about the distraction caused to the driver and alert him Our best model utilizes a genetically weighted ensemble of convolutional neural networks to achieve a 96% classification accuracy. This will help to avoid people from violating the traffic rules and help them follow the traffic rules sincerely. This project helps people to reach their destination safe and sound

REFERENCES

[1] R. A. Berri, A. G. Silva, R. S. Parpinelli, E. Girardi, and R. Arthur. A pattern recognition system for detecting use of mobile phones while driving. In Computer Vision Theory and Applications (VISAPP), 2014 International Conference on, volume 2, pages 411–418. IEEE, 2014.

[2] Y. Artan, O. Bulan, R. P. Loce, and P. Paul. Driver cell phone usage detection from hov/hot nir images. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pages 225–230, 2014.

[3] C. Craye and F. Karray. Driver distraction detection and recognition using RGB-D sensor. arXiv preprint arXiv:1502.00250, 2015.

[4] X. Zhang, N. Zheng, F. Wang, and Y. He. Visual Recognition of Driver Hand-held Cell Phone Use Based on Hidden CRF. 2011 IEEE International Conference on Vehicular Electronics and Safety (ICVES), pages 248–251, 2011.

[5] T. Hoang Ngan Le, Y. Zheng, C. Zhu, K. Luu, and M. Savvides. Multiple Scale Faster-RCNN Approach to Driver's Cell-Phone Usage and Hands on Steering Wheel Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages46–53, 2016. ISSN 21607516. doi:10.1109/CVPRW.2016.13.

[6] N. Das, E. Ohn-Bar, and M. M. Trivedi. On performance evaluation of driver hand detection algorithms: Challenges, dataset, and metrics. In Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on, pages 2953–2958. IEEE, 2015.

[7] K. Seshadri, F. Juefei-xu, D. K. Pal, M. Savvides, and C. P. Thor. Driver Cell Phone Usage Detection on Strategic Highway Research Program (SHRP2) Face View Videos. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 35—43, 2015.

[8] S. Martin, E. Ohn-Bar, A. Tawari, and M. M. Trivedi. Understanding head and hand activities and coordination in naturalistic driving videos. In Intelligent Vehicles Symposium Proceedings, 2014 IEEE, pages 884–889. IEEE, 2014.

[9] E. Ohn-bar and S. Martin. Driver hand activity analysis in naturalistic driving studies : challenges , algorithms , and experimental studies challenges , algorithms , and experimental studies. Journal of Electronic Imaging, 22(4), 2013. doi:10.1117/1.JEI.22.4.041119.

[10] E. Ohn-bar, S. Martin, A. Tawari, and M. Trivedi. Head , Eye , and Hand Patterns for Driver Activity Recognition. 22nd International Conference on Pattern Recognition (ICPR), 2014, pages 660—-665, 2014. doi:10.1109/ICPR.2014.124.