PREDICTING DISTRACTED DRIVING

A PROJECT REPORT

Submitted by

LAWRENCE XAVIER F
NISHA M
RAJSARANYA A
SIVARANJANI S

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

LOYOLA ICAM COLLEGE OF ENGINEERING AND TECHNOLOGY

CHENNAI-600034

ANNA UNIVERSITY : CHENNAI 600 025

JUNE 2022

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"PREDICTING DISTRACTED DRIVING"** is the bonafide work of **LAWRENCE XAVIER F (311118104034) ,NISHA M(311118104046) ,A.RAJSARANYA (311118104049),** and **SIVARANJANI S (311118104055)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. K. Gopalakrishnan	Mr Sathish B R
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor	Assistant Professor
Department of Computer Science and Engineering	Department of Computer Science and Engineering
Loyola-ICAM College of Engineering and Technology	Loyola-ICAM College of Engineering and Technology
Loyola Campus,	Loyola Campus,
Nungambakkam, Chennai-600034	Nungambakkam, Chennai-600034

Submitted for the project viva voce held on ………..…22/06/2022…………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Firstly, we are deeply obliged to the almighty for this opportunity and for leading us to the successful completion of the project. The experience has been very fruitful.

We express our sincere gratitude to the beloved Director **Rev Dr S Maria Wenisch SJ**, the Dean of Students **Dr Caleb Chanthi Raj** and the Dean of Engineering Education **Mr Nicolas Juhel**, for their continuous support and Encouragement.

We are deeply obliged to the Principal **Dr L Antony Michael Raj** for his inspiring guidance and invaluable advice during the project work. We also express our sincere thanks to the Head of the Department **Dr K Gopalakrishnan**, Project Coordinator **Mr. Remegius Praveen Sahayaraj L**, the teaching and non-teaching faculty of our department for their advice, support and guidance.

We thank our guide and project in charge, **Mr. B.R Sathish**, for his valuable suggestions and constant guidance throughout this project.

Last but not the least, we are extremely grateful to our family and friends, who have been a constant source of support during the preparation of this project work.

ABSTRACT

Distracted driving is the leading cause of motor vehicle crashes all over the world.The detection of a distracted driver is a key study field for decreasing road accidents.With deep neural networks, this research focuses on a methodology for reducing accidents caused by distracted drivers.The behaviors of the driver are developed using a CNN-based algorithm using the driver picture collection, which is then used to classify distracted drivers into distinct groups.The focus of this research is on an approach for lower the number of collisions caused by distracted drivers Deep neural networks are a type of artificial intelligence. In this research, we propose a set of computer vision approaches for processing picture data captured from inside automobiles to automatically detect when drivers are distracted. We accomplish this by using pictures to detect and localize a variety of things in automobiles that contribute to distracted driving (e.g., hands, cellphones, radio, etc.).Then, inside a picture, we use machine learning techniques to analyze the relative placements of these items in order to develop predictions about distracted driving.A CNN-based method is utilized to develop a driver's activities from a driving picture dataset, which is then used to categorize distracted drivers into different groups.A state farm dataset was used to create a deep neural network model, which includes 10 activities in 26 different subjects such as texting, cell phone use while driving, late arrival, normal driving, and alcohol use. The results from six epochs demonstrate that all of the experiments have above 75 percent accuracy, with the greatest result being 98 percent.

Index Terms—Distracted driver, road accident, action recognition, deep learning, convolutional neural network, vanilla CNN

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACRONYMS	ABBREVIATIONS
CNN	Convolution Neural Network
VGG	Visual Geometry Group
MLP	Multi Layer Perception
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
GPU	Graphic Processing Unit
FC	Fully Connected
EDA	Exploratory data analysis
GPS	Global positioning system
FP	False Positive
TP	True Positive
FN	False Negative
TN	True Negative
NHTSA	National Highway Traffic Safety Administration
UTS	union territories
AUC	American university of cairo dataset
SEC	Statefarm distracted driving dataset
R-CNN	Region-based Convolutional Neural Network.
ROI POOLING	Region of Interest (ROI) pooling
RGB	Red Green Blue
RMS PROP	Root Mean Squared Propagation
RELU	Rectified Linear Unit.
IDE	integrated development environment
N	number of observations
C	number of categories
P	probability predicted by the model

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

According to data released by the Ministry of Road Transport and Highways, more than three lakh road accidents were recorded in India in the calendar year 2020 by various states and union territories (UTs), killing 1,31,714 deaths and injuring 3,48,279 people. Another survey from the United States, published by the National Highway Traffic Safety Administration (NHTSA), said that distracted driving was responsible for 64.4 percent of fatalities. In addition, 94 percent of automobile accidents are caused by driver error. Furthermore, 86 percent of drivers admit to using one or more of the following while driving: answering phone calls, responding to messages, checking radio, conversing with a fellow passenger, eating or drinking while driving, checking maps, watching video, grooming, and surfing the web**.** We create computer vision algorithms to evaluate images from a dataset in order to detect distracted driving while keeping the context of the discovery. The state farm distracted driving dataset from Kaggle is the dataset we utilize for this issue. In this data collection, almost 1000 photographs were taken , with an in-car camera positioned just above the passenger side window and facing the driver window. The photos are separated into nine categories, one for safe driving and nine for distracted driving. Throughout the image collection, real drivers imitated different aspects of distracted and safe driving while the automobile stayed still (for safety reasons). This dataset is well-known, and it's frequently used in peer-reviewed research. For this study, we used photos from all nine distracted driving sessions as well as images from a single safe driving lesson. Talking on the phone (with both hands), texting (with both hands), drinking while driving, adjusting the car radio, interacting with a side passenger, applying cosmetics, and reaching for the back are all examples of distracted driving.In this study, we suggest a two-tiered technique to identify a picture as distracted or safe driving. We construct a Vanilla Convolutional Neural Network algorithm in the first layer to scan an image and then recognise and locate critical components that might contribute to distracted driving. Our dataset includes smartphones and bottles (external devices), as well as steering wheels, radios (internal to the car), and the left hand, right hand, look straight, look right, and keep looking back (human-centric). Following the identification and localization of these objects of interest in a photo, less advanced machine learning techniques are used to process the relative locations of the localized objects and categorize samples based on their proximity. The total contribution of our approaches in this study will create better and contextual inputs to the end user, which (we believe) will lead to greater driver comprehension and distracted driving correction.We go into further information regarding these consequences later in the research.

1.2 TECHNOLOGIES USED:

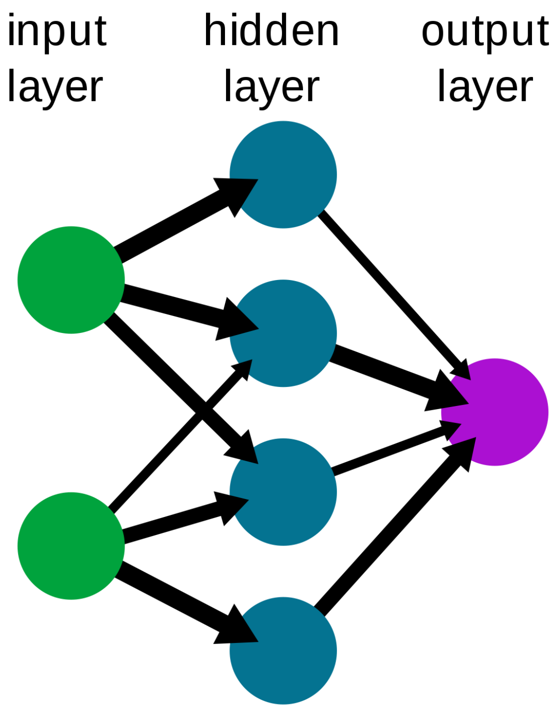
1.2.1 DEEP LEARNING:

Deep learning is a subset of machine learning, which may be thought of as a subset of machine learning. It is a field concerned with computer algorithms that learn and evolve on their own. Machine learning utilizes simpler concepts, but deep learning uses artificial neural networks, which are designed to replicate how people think and learn. Deep learning has aided image categorization, language translation, and speech recognition. It can solve any pattern recognition problem without involving humans..

1.2.2 NEURAL NETWORKS

Neural networks are a type of machine learning technique that uses numerous hidden layers and non-linear activation functions to describe complicated patterns in datasets. A neural network takes an input, runs it through numerous layers of hidden neurons (mini-functions with unique coefficients that must be learnt), and then outputs a prediction that is the sum of all the neurons' inputs.

Figure 1.1 Simple neural network



Layers are commonly used to structure neural networks. There are several layers in a neural network. Layers are built up of linked 'nodes' that each have a 'activation function.' The 'input layer' presents patterns to the network, which communicates with one or more 'hidden layers,' which do the actual processing via a system of weighted 'connections.' After that, the hidden layers connect to a 'output layer.'Each layer serves a distinct purpose, and the more levels there are, the more complicated the network becomes. A neural network is sometimes known as a multi-layer perceptron because of this. The input layer receives and passes input signals to the following layer. It collects information from the outside world.

All of the calculation's back-end operations are handled by the hidden layer. A network can have no hidden layers at all.

A neural network, on the other hand, contains at least one hidden layer. The output layer sends the hidden layer's calculation's ultimate result.

• The first is a multilayer perceptron, which has three or more layers and employs a nonlinear activation function.

• The second is a convolutional neural network, which is a version of multilayer perceptrons.

• The third is a recursive neural network, which produces structured predictions using weights, and the fourth is a recurrent neural network, which connects neurons in a guided cycle. The recurrent neural network design is used in the long short-term memory neural network, which does not require the activation function.

• The last two modules are sequence to sequence modules, which employ two recurrent networks and shallow neural networks to generate a vector space from a set of text. These neural networks are applications of the basic neural network .

1.3 OBJECTIVE OF THE SYSTEM AND ITS NEED:

The evidence implies that early detection of distracted driving, supported by visuals, is a critical component in reducing the severity of accidents.As a result, by utilizing the ability to predict distracted driving at an early stage, accidents can be avoided.As previously said, drivers are prone to disregarding the order to halt and rest, resulting in difficulties for which there is no alternative solution. As a result, the goal of this research is to develop a system that can anticipate the driver in such circumstances. When the driver is distracted, the forecast of distracted driving. A distracted driver can be alerted by the forecast and take essential steps in a timely manner, much as a distracted driver can refocus on the road and controls in a timely manner. The classification model will be created using vanilla CNN to identify and classify the drivers' faces. The forecast is made based on two conditions.If the motorist appears to be distracted in the photograph, he or she will be labeled as a distracted driver. It will be regarded as safe driving if the driver is not distracted or if no distracting objects are discovered.

CHAPTER 2
LITERATURE REVIEW

2.1Arup Kanti Dey, Bharti Goel, Sriram Chellappan,Context-driven detection of distracted driving using images from in-car cameras was published in the year 2021. They create a computer vision approach in this research that processes picture data taken from inside automobiles to automatically infer whether drivers are distracted. They accomplish it by using pictures to detect and localize a variety of elements in automobiles that lead to distracted driving. Then, inside a picture, they use machine learning techniques to evaluate the relative placements of these items in order to create predictions about distracted driving. Based on object localisation, the overall accuracy in detecting instances of inattentive driving is 94 percent.

2.2 Abdul Jamsheed V,Dr. B Janet,Dr. U Srinivasulu Reddy,Real time detection of distracted driving was the paper published in the year 2020. With deep neural networks, this research focuses on an approach for reducing accidents caused by inattentive drivers. The behaviors of the driver are developed using a CNN-based algorithm using the driver picture collection, which is then used to categorize distracted drivers into distinct groups. Vanilla CNN, vanilla CNN with data augmentation, and CNN with transfer learning are the three models included in the proposed system. A state farm dataset was used to create a deep neural network model, which comprises 10 behaviors in 26 distinct subjects such as texting, cell phone use while driving, late arrival, normal driving, and alcohol use. The results of five epochs demonstrate that all of the trials have above 75 percent accuracy, with the greatest result being 97 percent..

2.3Demeng Feng ,Yumeng Yue,Machine Learning Techniques for Distracted Driver Detection was published in the year 2019. This study focuses on detecting driver distracted activities using photos and other machine learning approaches. Our model is fed photos of the driver captured in the automobile as input. We employ several classifiers (linear SVM, softmax, naive bayes, decision tree, and 2-layer neural network) to generate a projected type of distracting activity that drivers are engaging in after preprocessing these photos to get input vectors. They were able to effectively develop a two-layer Neural Network model that performs well on the distracted driver detection test and has a 92.24 percent assessment accuracy.

2.4 Binbin Qin , Jiangbo Qian , Yu Xin, Baisong Liu, and Yihong Dong,Distracted Driver Detection Based on a CNN With Decreasing Filter Size was published in the year 2021.They suggested a novel D-HCNN model based on a decreasing filter size with just 0.76M parameters, a much lower number of parameters than many previous research' models. To boost performance, D-HCNN employs HOG feature pictures, L2 weight regularization, dropout, and batch normalization. We go through the benefits and concepts of D-HCNN in depth, as well as do experimental assessments on the AUC Distracted Driver (AUCD2) dataset. On AUCD2, the accuracy is 95.59 percent.

2.5.O. G. Basubeit, D. N. T. How, Y. C. Hou, K. S. M. Sahari,Distracted Driver Detection with Deep Convolutional Neural Network was published in the year 2019. This research looks at the best deep learning network design for reliably detecting distracted drivers in a visual stream. A thorough examination and extensive benchmark comparisons of pretrained deep convolutional neural networks are carried out specifically. The study compared the performance of twelve pre trained deep convolutional neural network models that were retrained on the State Farm dataset and then tested on the test dataset.The findings suggest that repurposing the VGG16 model to diagnose distracted drivers with up to 96 percent accuracy on unseen photos might be a good idea.

2.6 Furkan Omerustaoglu, C. Okan Sakar , Gorkem Kar,Distracted driver detection by combining in-vehicle and image data using deep learning was published in 2020.To increase the system's generalization capabilities, they included sensor data into the vision-based distracted driver detection model. In this study, they combined in-vehicle sensor data with photos of the driver captured while driving to improve the distracted driver detection task's correct classification rate. Finally, they used two alternative fusion algorithms to map the inputs to one of the 10 actions, combining the predictions of vision and sensor-based modalities. The results indicated that while the VGG16 and Inception V3-based CNN models' pure transfer learning accuracies on the public dataset were 61 percent and 32 percent, respectively, fine-tuning raised them to 77 percent and 80 percent.

2.7 Prof..Manya Gidwani, Deep Ruparel, Abhay Rajde ,Sahil Shah,distracted driver detection was published in the year 2020. As a result, an attempt is made to build a procedure and implement it utilizing various approaches and algorithms.
They demonstrate a robust Convolutional Neural Network-based method for detecting distracted drivers in this paper. To avoid overfitting to the training data, we use the pretrained VGG16 architecture weights and apply the Dense and Flatten layers to this model, as well as SGD as an optimizer. Experimental findings reveal that the model correctly predicts distracted drivers 93% of the time.

2.8.Bandar Alotaibi ,Munif Alotaibi,Distracted Driver Classification Using Deep Learning Published in 2019. To improve the performance of detecting a distracted driver's behavior, they proposed a method that combines three of the most advanced models in deep learning, namely the residual network, the Inception module, and the hierarchical recurrent neural network. They test their approach using two datasets (the State Farm dataset on the Kaggle platform and the AUC dataset). The datasets' photos were captured with a dashboard camera in order to detect inattentive drivers from 2D photographs. Their system has a precision of 96.28 percent.

2.9.Leonel Cuevas Valeriano; Paolo Napoletano; Raimondo Schettini,Recognition of driver distractions using deep learning was published in the year 2018. They devised and presented a comparison of various deep learning-based algorithms for classifying driver behavior using 2D camera data. The State Farm dataset was used to evaluate 10 distinct behaviors done by 26 people, including regular driving, texting, chatting on the phone, manipulating the radio, drinking, reaching behind, and so on. The results of three rounds of 5-fold cross validation demonstrate that all of the tested approaches are more than 90% accurate, with the best getting about 97 percent.

2.10 Khalid Alshalfan,**Mohammed Zakariah,Detecting Driver Distraction Using Deep-Learning Approach was published in the year 2021.**In this research, For driver-distraction detection, the suggested architecture was tested using the StateFarm dataset. This dataset is publicly available on Kaggle and is frequently used for this type of research. The proposed architecture achieved 96.95% accuracy.they used VGG16 ,CNN.

CHAPTER 3
SYSTEM ANALYSIS

3.1 SYSTEM REQUIREMENTS:

INTRODUCTION:

Design is a process that takes into account the structure of software, the details of how it works, and the interface between different modules. The design process translates requirements into a presentation of software that can be accessed for quality assurance before coding begins. Software design is constantly evolving as new methods and better analysis and understanding of borders emerge.At this early stage of the software design revolution, there is much room for improvement. Therefore, software design methodology is not as deep, flexible, and quantitative as more traditional engineering disciplines. There are techniques for designing software, and there are criteria for good design. Design notation can be used to help describe design patterns.

3.2 SOFTWARE REQUIREMENTS:

❖ Operating System	:	Windows 7,8,10 (64 bit)
❖ Software	:	Python
❖ Tools	:	Anaconda (Jupyter notebook IDE)
❖ Hard Disk	:	500GB and above
❖ RAM	:	8 GB and above
❖ Processor	:	I5 and above

❖ Python libraries : OpenCV, Tensorflow, Scikit Learn, Pandas,...

PYTHON:

Python is a high-level, general-purpose programming language that is interpreted.Python's design philosophy prioritizes code readability, as seen by its extensive use of indentation.Its language elements and object-oriented approach are aimed at assisting programmers in writing clear, logical code for both small and large-scale projects.Python is garbage-collected and dynamically typed.It supports a variety of programming paradigms, including structured (especially procedural) programming, object-oriented programming, and functional programming.Because of its extensive standard library, Python is frequently referred to as a "batteries included" language.

Python has the following features:

- Free and Open Source
- OOPs concept
- Support for GUI Programming
- High level language,
- Extensible feature:
- Python is Portable language
- Python is Integrated language
- Interpreted Language
- Large Standard Library
- Dynamically Typed Language
- Easy to use etc

3.3 Packages:

Open CV:

OPENCV (Open Source Computer Vision Library) is a programming library primarily for real-time computer vision.It was created by Intel and then sponsored by Willow Garage and Itseez (which was later acquired by Intel).The Apache 2 License makes the library cross-platform and free to use.Since 2011, GPU acceleration has been available in

OpenCV for real-time processing.

TENSORFLOW:

TensorFlow is an open source machine learning platform that runs from start to finish.This session focuses on using a specific TensorFlow API to develop and train machine learning models. TensorFlow is a rich framework for managing all parts of a machine learning system; however, this class focuses on using a specific TensorFlow API to develop and train machine learning models.

SCIKIT-LEARN

Scikit-learn (previously scikits.learn and also known as sklearn) is a Python machine learning library that is available for free.It includes support vector machines, random forests, gradient boosting, k-means, and DBSCAN, among other classification, regression, and clustering techniques, and is designed to work with the Python numerical and scientific libraries NumPy and SciPy.

PANDAS

PANDAS is a data processing and analysis software package created in the Python computer language.It includes data structures and methods for manipulating numerical tables and time series, in particular.It's free software distributed under the BSD three-clause licence.The word "panel data" is an econometrics term for data sets that comprise observations for the same persons over multiple time periods.Its moniker is a pun on the term "Python data analysis."While working as a researcher at AQR Capital from 2007 to 2010, Wes McKinney began developing what would become pandas.It is projected that it is faster than existing approaches for matching pathnames in directories, based on benchmarks.Apart from exact string search, we can employ wildcards ("*,?, [ranges]) with glob to make path retrieval more straightforward and convenient.

CHAPTER 4
SYSTEM DESIGN AND IMPLEMENTATION

4.1.1 EXISTING SYSTEM:

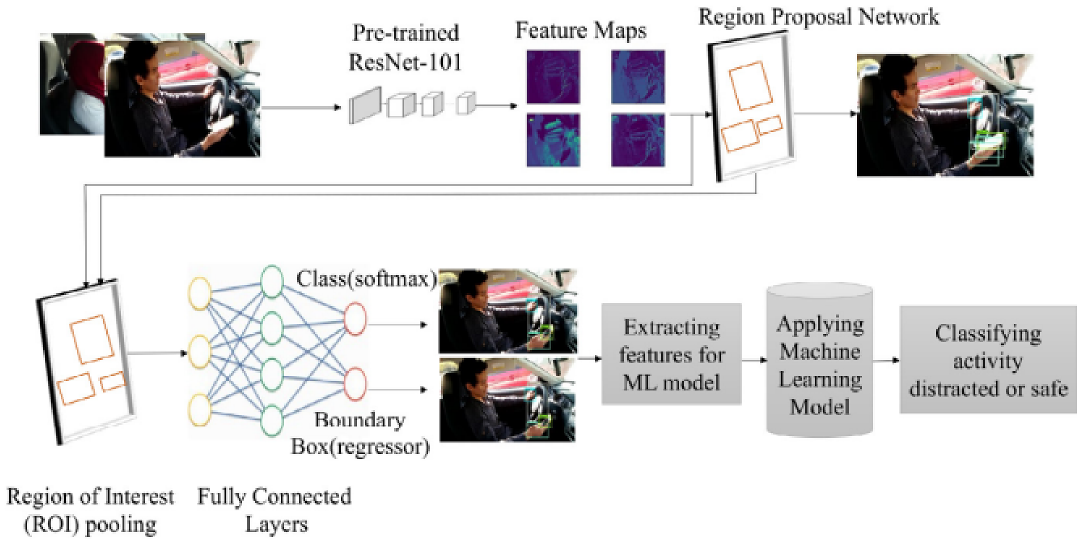
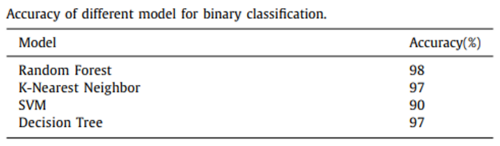

Pre-trained
ResNet-101
Feature Maps
Region Proposal Network
Class(softmax)
Boundary
Box(regressor)
Extracting features for ML model
Applying Machine Learning Model
Classifying activity distracted or safe
Region of Interest (ROI) pooling
Fully Connected Layers

Figure 4.1 system architecture of existing system

They utilised the AUC distracted driving dataset in the previous system. They isolated 150 unseen photos from the image collection divided across all classes under consideration for testing reasons (nine distracted and one safe driving class). Based on intuition and observation of the AUC dataset, the nine elements of interest in this present system are left hand, right hand, smartphone, bottle, radio, steering wheel, face facing straight, face looking behind, and face looking to the passenger side. They look for these things of interest in a picture and then locate them in the image to estimate their relative locations. They then use their relative locations to judge if the image is suggestive of inattentive driving.They used image processing to see if any of these things of interest are present, and then they localised them in the image to see where they are in relation to one another. They determine whether or not the image is suggestive of inattentive driving. The Faster R-CNN architecture for recognising and localising objects of interest in an image is one of the approaches used in their proposed distracted driving detection systems. They use the Faster R-CNN (Faster Regional-Convolutional Neural Networks) approach for object recognition and localization. For their challenge, they chose 1317 photos for training and validation that were evenly dispersed throughout all 10 classes of interest in their investigation. For training the ten courses, there were 102 photographs from six classes and 103 images from four classes, totaling 1024 images. They then chose 293 additional photos for algorithm validation (equally distributed across all 10 classes). Initially, the picture size for training was 1920 x 1080 pixels, but they lowered it to 960 x 540 pixels for faster training. They utilised the ResNet-101 network to extract features from training photos, and they now have the fundamental feature maps for the complete image dataset. They used numerous bounding boxes to encompass the full picture, some of which had foreground and some of which contained background. They created and optimised a simplified CNN to distinguish between boxes having foreground and background components. They used the idea of ROI pooling to turn these variable-sized feature maps into fixed-sized feature maps, in which the variable-sized feature maps are broken into smaller portions of fixed-sized feature maps to ease processing without sacrificing accuracy. Finally, they created a regressor to tighten the localised object, since the narrower the box is, the better our ability to discern relative locations of many items of interest in an image for contextual categorization of distracted driving will be. They then analyse the identified and localised items of interest (i.e., foreground classes) to reach a final conclusion on whether they are distracted driving or not. To get the greatest results, they devised a basic random forest-based method. The characteristics they detected and processed for this categorization task. They constructed and evaluated four different machine learning models to identify distracted driving from safe driving after localising items of interest and determining attributes. They sought to categorise merely distracted driving or not using the same localization technique, followed by characteristics retrieved, in addition to multi-class classification of distracted driving.On average, their approach required 200ms to process one image when run end-to-end on a PC .This included localization of items of interest, categorization of the image as suggestive of distracted driving, and so on.

Figure 4.2 Accuracy table of existing system



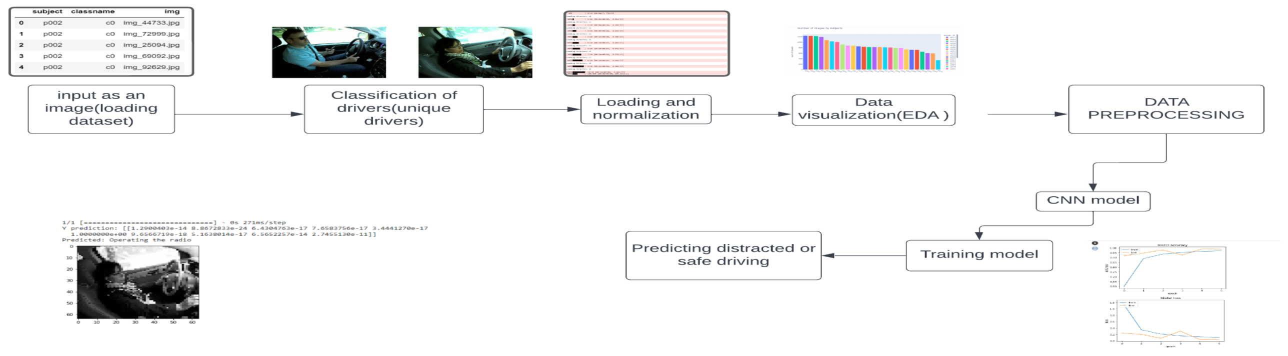
Accuracy of different model for binary classification.

Model	Accuracy(%)
Random Forest	98
K-Nearest Neighbor	97
SVM	90
Decision Tree	97

4.1.2 PROPOSED SYSTEM:

Deep learning networks generate relevant characteristics from raw data by learning from several layers of representation. All of the techniques used in this study are either convolutional neural networks or small variations on CNN design.The main methods of CNN are explained first, followed by a discussion of the CNN model's modifications. Convolutional layer, max pool layer, ReLU layer, dropout layer, and fully linked layers are only a few of the layers of a CNN. The State Farm data collection is used to train the model, and it contains photographs of drivers drinking, conversing with co-passengers, texting, and chatting on the phone while driving, among other things.Convolutional neural networks are built using the Keras and TensorFlow packages. The model was created using the Jupyter notebook research environment, which allows for a high-performance configuration for training and testing. The data set's images are all cleaned to the same standard, and ten classes are defined. Cropping the image to an equal size, straightening the pixels, and deleting the undesirable data are all part of the cleaning process. The excel sheet keeps track of the image's name and the classes that go with it. Class names are used as labels for the images in the CSV file, and image names are used to match the labels with the relevant photos. We create a modified vanilla CNN model to produce results with high accuracy.

4.1.3 SYSTEM ARCHITECTURE



	subject	classname	img
0	p002	c0	img_44733.jpg
1	p002	c0	img_72999.jpg
2	p002	c0	img_25094.jpg
3	p002	c0	img_69092.jpg
4	p002	c0	img_92629.jpg

```
1/1 [==============================] - 0s 271ms/step
Y prediction: [[1.2900403e-14 8.8672833e-24 6.4304763e-17 7.6583756e-17 3.4441270e-17
  1.0000000e+00 9.6566719e-18 5.1638014e-17 6.5652257e-14 2.7455130e-11]]
Predicted: Operating the radio
```

input as an image(loading dataset)
Classification of drivers(unique drivers)
Loading and normalization
Data visualization(EDA)
DATA PREPROCESSING
CNN model
Training model
Predicting distracted or safe driving

Figure 4.3 System Architecture of proposed system

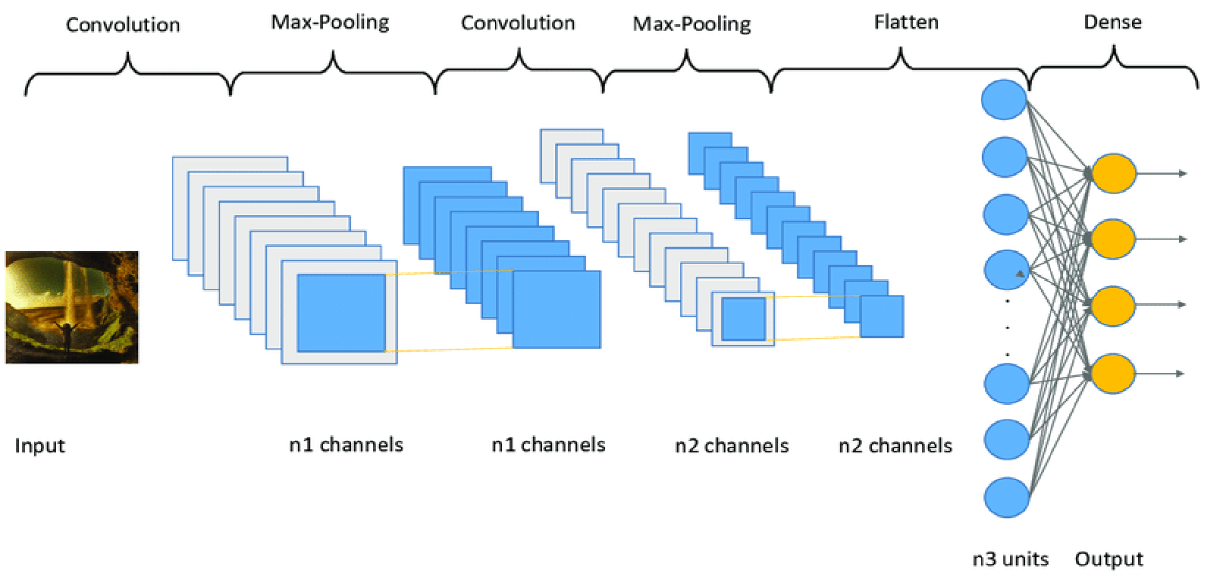
The photos are collected from the state farm distracted driving dataset as input. The drivers are then categorized using the provided dataset. Following that, we must go through the loading and normalization procedures. We must first load the training data set. The pictures are divided into 10 classes, numbered c0 through c9, with each class including various driver actions and graphics. The dataset must then be validated. further Using the EDA principle, we must visualize the data. EDA stands for exploratory data analysis, which is a method of analyzing and summarizing the primary properties of a dataset, such as class distribution, size, and distribution. In our project, for example.In our project, we utilized the aforementioned categories to indicate the quantity of photographs by category and the driver's ID who is distracted. In addition, we analyze the data and create an image of the driver in their mode of distraction. The primary architecture of the vanilla CNN model is made up of three convolution layers: a flat layer, two dense layers with relu and dropouts, and one dense layer for classification using softmax. We can discover the total number of parameters using the technique. Which parameters can you train and which can't you? training For the batches, the model is complete. Accuracy, loss, validation loss, and validation accuracy are all measured in each batch. We'll plot the validation accuracy when we've trained all of the batches.We'll plot the validation accuracy and validation loss across epochs once we've trained all the batches. Then we can anticipate the driver's actions. To forecast the activities of drivers, we concentrate on distractions, which are classified into ten categories for the drive greyscale photos in the data trained and verified dataset.

4.1.4 ALGORITHMS AND TECHNIQUES:

VANILLA CONVOLUTION NEURAL NETWORKS (CNN):

The vanilla CNN model is made up of three convolutional layers, a flatten layer, and three dense layers in total. Layer 1 of the convolutional network has 60 filters, each with a 3x3 kernel size. The relu activation function is used with the same padding as before, and the weights are set to 0.0001. This layer receives a 0.3 dropout. A total of 90 filters, each measuring 3x3, are employed in the second convolutional layer. This layer also has a 0.3 dropout applied to it.In the third convolutional, a total of 200 filters with a dropout of 0.5 are utilized in a similar way. A flatten layer and three dense layers with 512, 128 and 10 filters each are utilized after three convolutional layers. Loss is also calculated using the RMS-Prop optimizer and category cross entropy. ReLU is the activation function for three convolutional layers, ReLU is the activation function for the first dense layer, and softmax is the activation function for the last dense layer.

Figure 4.4 Vanilla CNN architecture



4.2 MODULES

4.2.1 DATASET

Data collection is the systematic process of acquiring and measuring information on variables of interest in order to answer research questions, test hypotheses, and make conclusions and In this study, the dataset we used is the Statefarm Distracted Driving Dataset . The collection contains 1000 tagged photos of 26 persons of various colors, ethnicities, actions, and ages. These subjects were used to classify them into ten categories, including normal/safe driving, talking on the phone with both hands, controlling the radio, text messaging with left hand, text messaging with right hand, talking to a co-passenger, drinking while driving, hair and makeup, reaching behind, and so on. The action class of each picture is labelled.The current study made use of the StateFarm distraction-detection dataset.This is the most widely used dataset for detecting driver attention, and it has been utilized in several research.Each class has around 2,200 RGB pictures at a size of 640x480 pixels.In this data set it shows the number of photos for each class.To build fresh training and testing subsets from the initial training set, the holdout split approach was used to create 10% and 30% testing sets.

Figure 4.5 Collection of the images

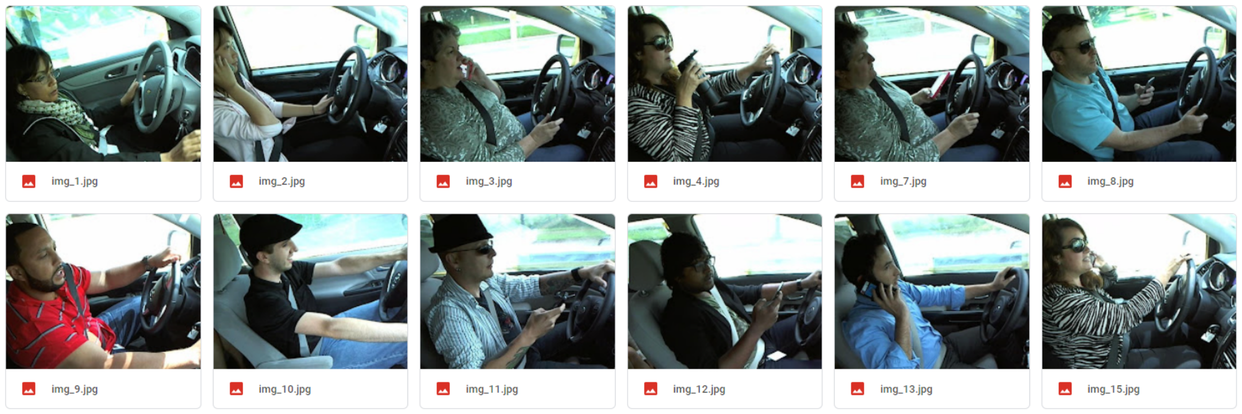


Table 4.1:Dataset details with number of images per class

CLASSES	DRIVER ACTION	IMAGES
C0	safe driving	2489
C1	texting - right	2267
C2	talking on the phone - right	2317
C3	texting - left	2346
C4	talking on the phone - left	2326
C5	operating the radio	2312
C6	drinking	2325
C7	reaching behind	2002
C8	hair and makeup	1911
C9	talking to passenger	2129
SUM		22424

4.2.2 LOADING AND NORMALIZATION

Modeling is the process of inferring a representative model from data. The training dataset is the set of data used to build the model, which includes known properties and a target. The test dataset, also known as the validation dataset, will be used to verify the correctness of the newly generated model. The whole known dataset may be split into a training dataset and a test dataset to make this procedure easier. The training dataset is being loaded. Return the original path's train photos and train labels.The size of the validation dataset we load is 200000, with img rows=64, img cols=64, and color type=3.Set the number of classes to ten in advance to load each and every directory completely .Using open cv to load a grayscale color picture and minimize its size.

4.2.2.1.THE TEST DATASET:

The test set is a distinct collection of data that is used to put the model to the test once it has been trained.It generates an impartial final model performance metric in terms of precision, accuracy, and other factors.

4.2.2.2.THE TRAINING DATASET:

It's the collection of data used to train the model and teach it how to find hidden features and patterns in the data.The same training data is supplied to the neural network design periodically throughout each epoch, and the model continues to learn the data's attributes.
The training set should include a diverse collection of inputs so that the model may be trained in all settings and predict any future data sample.

4.2.2.3.THE VALIDATION DATASET

The validation set is a collection of data that is independent from the training set and is used to test the performance of our model during training.This validation procedure provides data that we may use to fine-tune the model's hyperparameters and settings. It's as if we have a critic informing us whether or not our training is progressing in the proper path.
After every epoch, the model is trained on the training set and evaluated on the validation set.

The major goal of dividing the dataset into a validation set is to avoid overfitting, which occurs when a model becomes extremely effective at categorizing samples in the training set but is unable to generalize and make correct classifications on the data it has.

4.2.3 EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis is an unavoidable step in fine-tuning the given data set(s) in a different form of analysis to understand the insights of the key characteristics of various entities of the data set, such as column(s), row(s), using Pandas, NumPy, Statistical Methods, and Data visualization packages. The goal of the EDA process would be to get the most out of a dataset.We took 500 test photos using our suggested approach. We would be able to determine the total number of photos using the EDA idea (22924). There are 17939 training photos in total, divided into 10 training categories. We could also find out how many validation photos there are (4485).

4.2.4 DATA VISUALIZATION

The graphical depiction of information and data in a pictorial or graphical manner is known as data visualization (Example: charts, graphs, and maps). Data visualization tools make it easy to observe and comprehend trends, patterns, and outliers in data.

Multivariate Analysis: We shall compare more than two variables in the multivariate analysis.
In our system we plot the histogram chart to display the number of images by category as well as number of images by subjects.

4.2.5 DATA PREPROCESSING

The process of converting raw data into a comprehensible format is known as data preparation. We can't deal with raw data, thus this is a key stage in data mining. Before using machine learning or data mining methods, make sure the data is of good quality.

Raw data is prone to inconsistencies in formatting, human mistakes, and incompleteness. Data preparation overcomes these problems by making datasets more comprehensive and efficient to analyse.Preprocessing is the process of transforming or changing data through a sequence of procedures. It's a transformation that's done to our data before it's fed into an algorithm. Data processing is the process of retrieving, transforming, or changing data, usually using a computer.CV2 is used to resize images to 64 64 3 in order to increase the classifier's computation performance. The dataset is split into an 80:20 Training-Testing ratio using stratified splitting. The training dataset is divided into two parts: 90:10 Training and Validation. As a result, the final training set has 17939 photos, whereas the final validation set contains 4485.

4.2.6.CNN MODEL CLASSIFICATION

The vanilla CNN model is made up of three convolutional layers, a flatten layer, and three dense layers in total. Layer 1 of the convolutional network has 60 filters, each with a 3x3 kernel size. The relu activation function is used with the same padding as before, and the weights are set to 0.0001. This layer receives a 0.3 dropout. A total of 90 filters, each measuring 3x3, are employed in the second convolutional layer. This layer also has a 0.3 dropout applied to it.In the third convolutional, a total of 200 filters with a dropout of 0.5 are utilized in a similar way. A flatten layer and three dense layers with 512, 128 and 10 filters each are utilized after three convolutional layers. Loss is also calculated using the RMS-Prop optimizer and category cross entropy. ReLU is the activation function for three convolutional layers, ReLU is the activation function for the first dense layer, and softmax is the activation function for the last dense layer.

Table 4.2 Model configuration of vanilla CNN

LAYER	OUTPUT SHAPE
Convolutional Layer	(None, 62, 62, 32)
Max pooling	(None, 31, 31, 32)
Convolutional Layer	(None, 31, 31, 64)
Max pooling	(None, 16, 16, 64)
Convolutional Layer	(None, 16, 16, 128)
Max pooling	(None, 8, 8, 128)
Flatten	(None, 8192)
Dense	(None, 512)
Dense	(None, 128)
Dense	(None, 10)

CONVOLUTIONAL LAYERS:

A convolutional neural network (CNN) is a type of neural network that includes certain convolutional layers (and some other layers). A convolutional layer is made up of a number of filters that work together to perform convolutional operations. This layer is the first to extract features from the input pictures. Convolution is conducted between the input picture and a filter of a certain size MxM in this layer. The dot product between the filter and the sections of the input picture with regard to the filter's size is calculated by sliding the filter across the input image (MxM).

FLATTEN LAYER:

A flatten layer reduces the input's spatial dimensions to the channel dimension.It is utilised in the classification layer over feature maps because it is easier to comprehend and less prone to overfitting than a fully linked layer.

DENSE LAYERS:

The dense layer is a simple layer of neurons in which each neuron receives input from all of the neurons in the preceding layer, thus the name. Dense Layers are used to identify images based on convolutional layer output.

MAX POOLING LAYER:

Pooling that chooses the maximum element from the region of the feature map covered by the filter is known as max pooling. As a result, the output of the max-pooling layer would be a feature map with the most prominent features from the preceding feature map.

4.2.7 TRAINING MODEL AND PREDICTION:

Training a model necessitates a methodical, repeatable procedure that makes the most of your available training data and your data science team's time. You must first define your issue statement, retrieve your data set, then clean the data before presenting it to the model before you can begin the training process.The model is trained for six epochs, each with a batch size of 40. The history of loss, accuracy, validation loss, and validation accuracy is shown. Then, using the testing and training datasets, we plot the validation accuracy and validation loss over epochs as a graph.With the x test, y test, and verbose, we are assessing the model into score 1 in the prediction. For the score1, we can forecast the accuracy and amount of loss. Finally, after a few seconds, we can forecast the driver's conduct based on the cause for the distraction.

CHAPTER 5
PERFORMANCE METRICS

One of the most crucial phases in developing an effective Machine Learning model is evaluating its performance. Different metrics are utilized to evaluate the model's performance or quality, and these measures are referred to as performance metrics or evaluation metrics. These performance measures allow us to see how well our model worked with the data we provided. By changing the hyper-parameters, we can increase the model's performance.Each machine learning model aspires to generalize well to new or unknown data, and performance measures aid in determining how well the model generalizes. Each job or issue in machine learning is classified into two categories: classification and regression. Because not all measurements are appropriate for all situations, it is critical to recognise and comprehend which metrics should be utilized. For both Regression and Classification tasks, different assessment measures are utilized. In this topic, we will discuss metrics used for classification and regression tasks.

5.1Accuracy Score

The accuracy score is the percentage of correct predictions made by the categorization model.

Classification Accuracy Score=TP+TN / TP+FN+TN+FP

Where FP stands for False Positive (where we projected YES but got NO), TP stands for True Positive (where we anticipated proper classification), FN stands for False Negative (where we expected NO but got YES), and TN is for True Negative (predictions where correct).

❖ True positive (TP)

The number of records that were actually positive and were classified positive.

❖ False Negative (FN)

The number of records that were negative but were classified positive.

❖ True Negative (TN)

The number of records that were actually negative and were classified negative.

❖ False Positive (FP)

The number of records that were positive but were classified negative.

5.2 Score 1

The Score 1 is a statistic that takes accuracy and recall into account. It is also known as the harmonic mean of the two. The harmonic mean is a method of calculating an "average" of numbers that is said to be better for ratios (such as precision and recall) than the usual arithmetic mean.Because of the unequal distribution of classes across the cases, Score 1 Score is a better metric than accuracy.

$$2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

5.3 Sensitivity/ Recall

Sensitivity is the percentage of people who will be appropriately classified as positive(distracted). Because of the high sensitivity, more people have been properly predicted to be distracted.

Sensitivity=TP / TP+FN

5.4 Specificity

Specificity is the fraction of individuals who will be correctly predicted as a negative class. High specificity means a greater number of individuals have been correctly predicted not Distracted.

Specificity = TN / TN + FP

5.5.Performance on Predicting Distracted driving

After splitting the images for each action class in a 10:30 ratio for training and testing respectively, the following test results were achieved for predicting distracted driving for the given input images. An average accuracy of **98.70 %** was achieved in predicting distracted driving if the driver is distracted or not.

Table 5.1 Performance metrics for Predicting distracted driving

	Accuracy	**Loss**
Vanilla CNN	98	0.0456
Alexnet	93	0.2187
Decission tree	97	0.0534
SVM	90	0.0526
K nearest neighbour	97	0.2870

CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

Today, distracted driving is a serious public health threat. With the introduction of several technologies such as cellphones, smartwatches, GPS devices, tablets, and other devices that people routinely use while driving, the problem has gotten worse. We use computer vision techniques to detect distracted driving incidents using photos from a dataset in this work. Unlike other methods, our proposed method distinguishes unique drivers first, then recognises and localizes crucial elements of interest in a picture that indicate distracted driving. We create algorithms that process the relative locations of these objects in the image to classify nine types of distracted driving, including talking on the phone (with left and right hands), texting (with left and right hands), drinking while driving, operating the car radio, talking to a side passenger, applying makeup, and reaching for the back.Our object identification, localization, and classification accuracies are good, giving us confidence that our suggested methodologies may be used to improve driver safety, which is a pressing issue today. Furthermore, our suggested approach's contextual input to drivers is a unique feature. We're also assessing the impact on avoiding dangerous and inattentive driving on the roadways.In this study, the model is trained to handle the problem of driver distraction. The proposed model reached 98 percent accuracy, and we attempted a new model that required substantially less training time while maintaining the same level of accuracy.With more data, this approach can still be improved. However, by focusing on the positive qualities of this method, the disadvantages may be overlooked.

6.2 FUTURE WORK

We're continuing to work on reducing the amount of parameters and computation time as a result of this effort. Incorporating temporal context may aid in the reduction of misclassification mistakes and, as a result, improve accuracy. We also want to create a system that can identify visual, cognitive, and manual distractions in the future.On our system with 8GB RAM, the system processes 42 pictures per second.Investigate further feature extraction methods.For Image Representation, combine several feature vectors.Modern Deep Neural Network Architectures should be implemented.We may potentially speed up the CNN calculation by compressing the model, as stated in the previous section.

APPENDIX

APPENDIX A

A1: Importing libraries

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
```

A2:Loading Dataset

```
import os
from glob import glob
import random
import time
import tensorflow
import datetime
os.environ['KERAS_BACKEND'] = 'tensorflow'
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3' # 3 = INFO, WARNING, and ERROR
from tqdm import tqdm
import numpy as np
import pandas as pd
from IPython.display import FileLink
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
%matplotlib inline
from IPython.display import display, Image
import matplotlib.image as mpimg
import cv2

from sklearn.model_selection import train_test_split
from sklearn.datasets import load_files
from keras.utils import np_utils
from sklearn.utils import shuffle
from sklearn.metrics import log_loss

from tensorflow import keras

from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout,
BatchNormalization, GlobalAveragePooling2D
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.applications.vgg16 import VGG16


dataset = pd.read_csv('driver_imgs_list.csv')
dataset.head(5)


# Groupby subjects
by_drivers = dataset.groupby('subject')
# Groupby unique drivers
unique_drivers = by_drivers.groups.keys() # drivers id
print('There are : ',len(unique_drivers), ' unique drivers')
print('There is a mean of
',round(dataset.groupby('subject').count()['classname'].mean()), ' images
by driver.')
```

A3.Loading and normalization:

```
NUMBER_CLASSES = 10 # 10 classes
# Read with opencv
def get_cv2_image(path, img_rows, img_cols, color_type=3):
    """
    Function that return an opencv image from the path and the right
number of dimension
    """
    if color_type == 1: # Loading as Grayscale image
        img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    elif color_type == 3: # Loading as color image
        img = cv2.imread(path, cv2.IMREAD_COLOR)
    img = cv2.resize(img, (img_rows, img_cols)) # Reduce size
    return img

# Loading Training dataset
def load_train(img_rows, img_cols, color_type=3):
    """
    Return train images and train labels from the original path
    """
    train_images = []
    train_labels = []
    # Loop over the training folder
    for classed in tqdm(range(NUMBER_CLASSES)):
        print('Loading directory c{}'.format(classed))
        files = glob(os.path.join(r'C:\Users\LENOVO\Documents\source
code\image\image/train/c' + str(classed), '*.jpg'))
        for file in files:
            img = get_cv2_image(file, img_rows, img_cols, color_type)
            train_images.append(img)
            train_labels.append(classed)
    return train_images, train_labels

def read_and_normalize_train_data(img_rows, img_cols, color_type):
    """
    Load + categorical + split
    """
    X, labels = load_train(img_rows, img_cols, color_type)
    y = np_utils.to_categorical(labels, 10) #categorical train label
    x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42) # split into train and test
```

```
    x_train = np.array(x_train,
dtype=np.uint8).reshape(-1,img_rows,img_cols,color_type)
    x_test = np.array(x_test,
dtype=np.uint8).reshape(-1,img_rows,img_cols,color_type)

    return x_train, x_test, y_train, y_test

# Loading validation dataset
def load_test(size=200000, img_rows=64, img_cols=64, color_type=3):
    """
    Same as above but for validation dataset
    """
    path = os.path.join(r'C:\Users\LENOVO\Documents\source
code\image\image/test', '*.jpg')
    files = sorted(glob(path))
    X_test, X_test_id = [], []
    total = 0
    files_size = len(files)
    for file in tqdm(files):
        if total >= size or total >= files_size:
            break
        file_base = os.path.basename(file)
        img = get_cv2_image(file, img_rows, img_cols, color_type)
        X_test.append(img)
        X_test_id.append(file_base)
        total += 1
    return X_test, X_test_id

def read_and_normalize_sampled_test_data(size, img_rows, img_cols,
color_type=3):
    test_data, test_ids = load_test(size, img_rows, img_cols, color_type)
    test_data = np.array(test_data, dtype=np.uint8)
    test_data = test_data.reshape(-1,img_rows,img_cols,color_type)
    return test_data, test_ids
```

```
img_rows = 64 # dimension of images
img_cols = 64
color_type = 1 # grey
nb_test_samples = 200

# loading train images
x_train, x_test, y_train, y_test = read_and_normalize_train_data(img_rows,
img_cols, color_type)

# loading validation images
test_files, test_targets =
read_and_normalize_sampled_test_data(nb_test_samples, img_rows, img_cols,
color_type)
```

A4:Exploratory Data analysis

```
# Statistics
# Load the list of names
names = [item[17:19] for item in
sorted(glob(r"C:\Users\LENOVO\Documents\source
code\image\image/train/*/"))]
test_files_size =
len(np.array(glob(os.path.join(r'C:\Users\LENOVO\Documents\source
code\image\image/test', '*.jpg'))))
x_train_size = len(x_train)
categories_size = len(names)
x_test_size = len(x_test)
print('There are %s total images.\n' % (test_files_size + x_train_size +
x_test_size))
print('There are %d training images.' % x_train_size)
print('There are %d total training categories.' % categories_size)
print('There are %d validation images.' % x_test_size)
print('There are %d test images.'% test_files_size)
```

A5.Data visualization

```
import plotly.express as px

px.histogram(dataset, x="classname", color="classname", title="Number of 
images by categories ")

# Find the frequency of images per driver
drivers_id = 
pd.DataFrame((dataset['subject'].value_counts()).reset_index())
drivers_id.columns = ['driver_id', 'Counts']
px.histogram(drivers_id, x="driver_id",y="Counts" ,color="driver_id",
title="Number of images by subjects ")
```

A6:Data preprocessing:

```
activity_map = {'c0': 'Safe driving',
                'c1': 'Texting - right',
                'c2': 'Talking on the phone - right',
                'c3': 'Texting - left',
                'c4': 'Talking on the phone - left',
                'c5': 'Operating the radio',
                'c6': 'Drinking',
                'c7': 'Reaching behind',
                'c8': 'Hair and makeup',
                'c9': 'Talking to passenger'}


plt.figure(figsize = (12, 20))
image_count = 1
BASE_URL = r'C:\Users\LENOVO\Documents\source code\image\image/train/'
for directory in os.listdir(BASE_URL):
    if directory[0] != '.':
        for i, file in enumerate(os.listdir(BASE_URL + directory)):
            if i == 1:
                break
            else:
                fig = plt.subplot(5, 2, image_count)
                image_count += 1
                image = mpimg.imread(BASE_URL + directory + '/' + file)
                plt.imshow(image)
                plt.title(activity_map[directory])

def create_submission(predictions, test_id, info):
    """
    Submission function for participating to the competition
    """
    result = pd.DataFrame(predictions, columns=['c0', 'c1', 'c2', 'c3',
'c4', 'c5', 'c6', 'c7', 'c8', 'c9'])
    result.loc[:, 'img'] = pd.Series(test_id, index=result.index)

    now = datetime.datetime.now()

    if not os.path.isdir('kaggle_submissions'):
        os.mkdir('kaggle_submissions')

    suffix = "{}_{}".format(info,str(now.strftime("%Y-%m-%d-%H-%M")))
    sub_file = os.path.join('kaggle_submissions', 'submission_' + suffix +
'.csv')

    result.to_csv(sub_file, index=False)

    return sub_file
```

A7: Building Vanilla CNN

```
# Number of batch size and epochs
batch_size = 40 #40
nb_epoch = 6 #10
```

```
models_dir = "saved_models"
if not os.path.exists(models_dir):
    os.makedirs(models_dir)

checkpointer =
ModelCheckpoint(filepath='saved_models/weights_best_vanilla.hdf5',
                               monitor='val_loss', mode='min',
                               verbose=1, save_best_only=True)
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=2)
#callbacks = [checkpointer, es]


def create_model():
    model = Sequential()

    ## CNN 1
    model.add(Conv2D(32,(3,3),activation='relu',input_shape=(img_rows,
img_cols, color_type)))
    model.add(BatchNormalization())
    model.add(Conv2D(32,(3,3),activation='relu',padding='same'))
    model.add(BatchNormalization(axis = 3))
    model.add(MaxPooling2D(pool_size=(2,2),padding='same'))
    model.add(Dropout(0.3))

    ## CNN 2
    model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
    model.add(BatchNormalization())
    model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
    model.add(BatchNormalization(axis = 3))
    model.add(MaxPooling2D(pool_size=(2,2),padding='same'))
    model.add(Dropout(0.3))

    ## CNN 3
    model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
    model.add(BatchNormalization())
    model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
    model.add(BatchNormalization(axis = 3))
    model.add(MaxPooling2D(pool_size=(2,2),padding='same'))
    model.add(Dropout(0.5))

    ## Output
    model.add(Flatten())
    model.add(Dense(512,activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(128,activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(10,activation='softmax'))

    return model

model = create_model()

# More details about the layers
model.summary()

# Compiling the model
model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=['accuracy'])
```

A8:Training using Vanilla CNN

```
history = model.fit(x_train, y_train,
          validation_data=(x_test, y_test),
          epochs=nb_epoch, batch_size=batch_size, verbose=1)

#model.load_weights('saved_models/weights_best_vanilla.hdf5')
print('History of the training',history.history)


model.save("distracted_driving.h5")
```

```
def plot_train_history(history):
    """
    Plot the validation accuracy and validation loss over epochs
    """
    # Summarize history for accuracy
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

    # Summarize history for loss
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

plot_train_history(history)
```

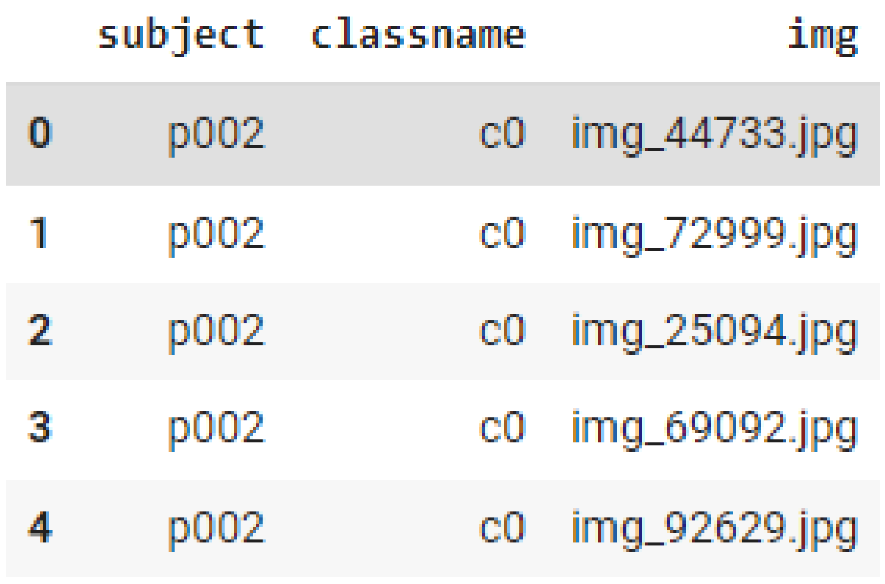
A9 : Prediction

```
def plot_test_class(model, test_files, image_number, color_type=1):
    """
    Function that tests or model on test images and show the results
    """
    img_brute = test_files[image_number]
    img_brute = cv2.resize(img_brute,(img_rows,img_cols))
    plt.imshow(img_brute, cmap='gray')

    new_img = img_brute.reshape(-1,img_rows,img_cols,color_type)
```

```
        y_prediction  =  model.predict(new_img,  batch_size=batch_size,
verbose=1)
    print('Y prediction: {}'.format(y_prediction))
                                                    print('Predicted:
{}'.format(activity_map.get('c{}'.format(np.argmax(y_prediction)))))

    plt.show()



score1 = model.evaluate(x_test, y_test, verbose=1)


print('Loss: ', score1[0])
print('Accuracy: ', score1[1]*100, ' %')


for i in range(10):
    plot_test_class(model, test_files, i)
```

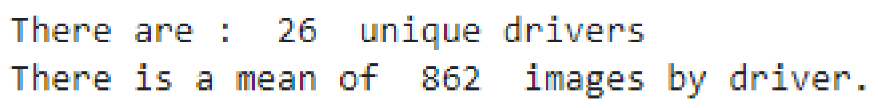
APPENDIX B

B1:Loading the dataset



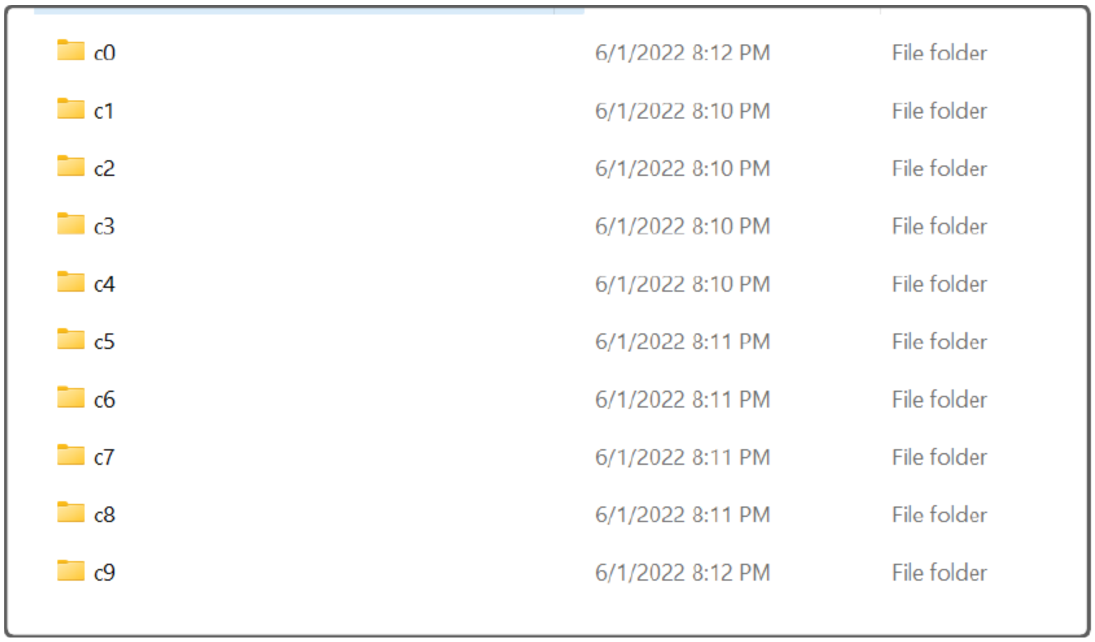
	subject	classname	img
0	p002	c0	img_44733.jpg
1	p002	c0	img_72999.jpg
2	p002	c0	img_25094.jpg
3	p002	c0	img_69092.jpg
4	p002	c0	img_92629.jpg

B2:Classification of drivers

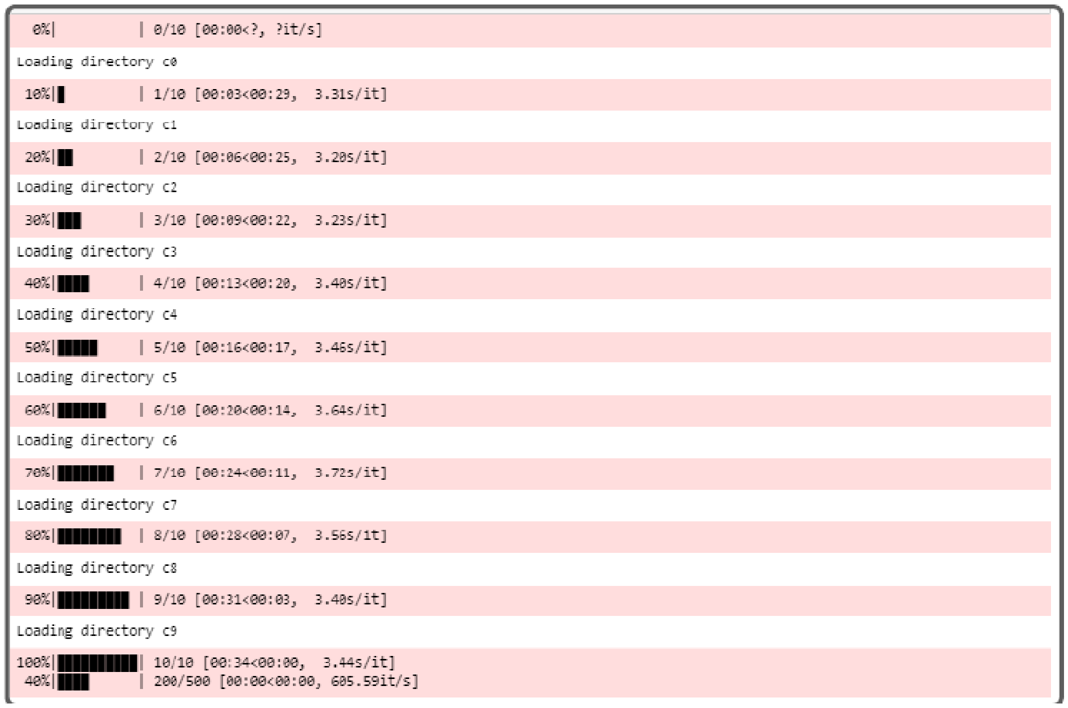


```
There are :  26  unique drivers
There is a mean of  862  images by driver.
```

B3.Loading and normalization:

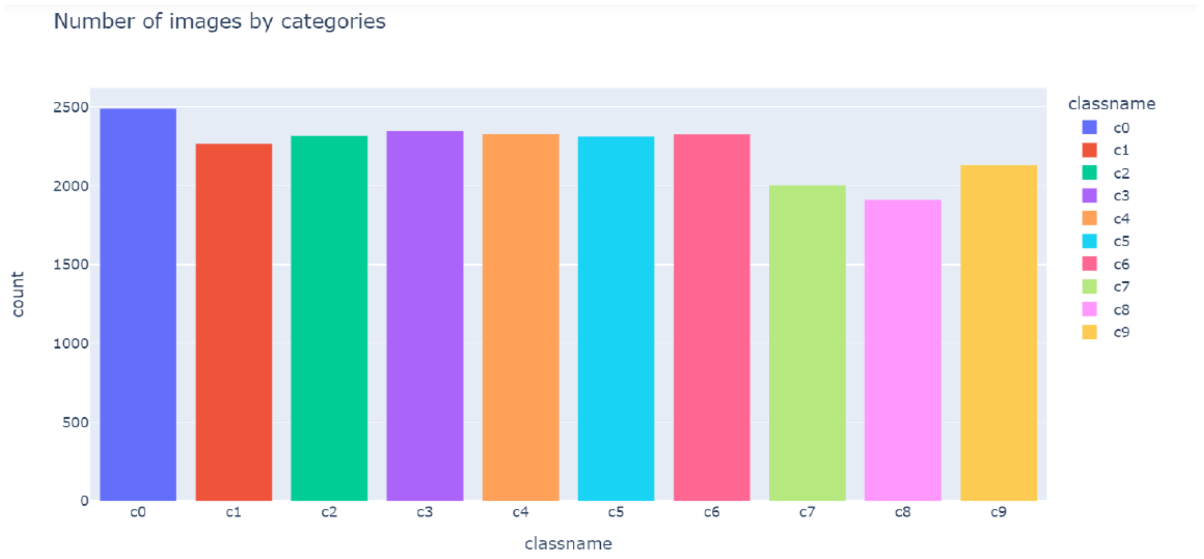


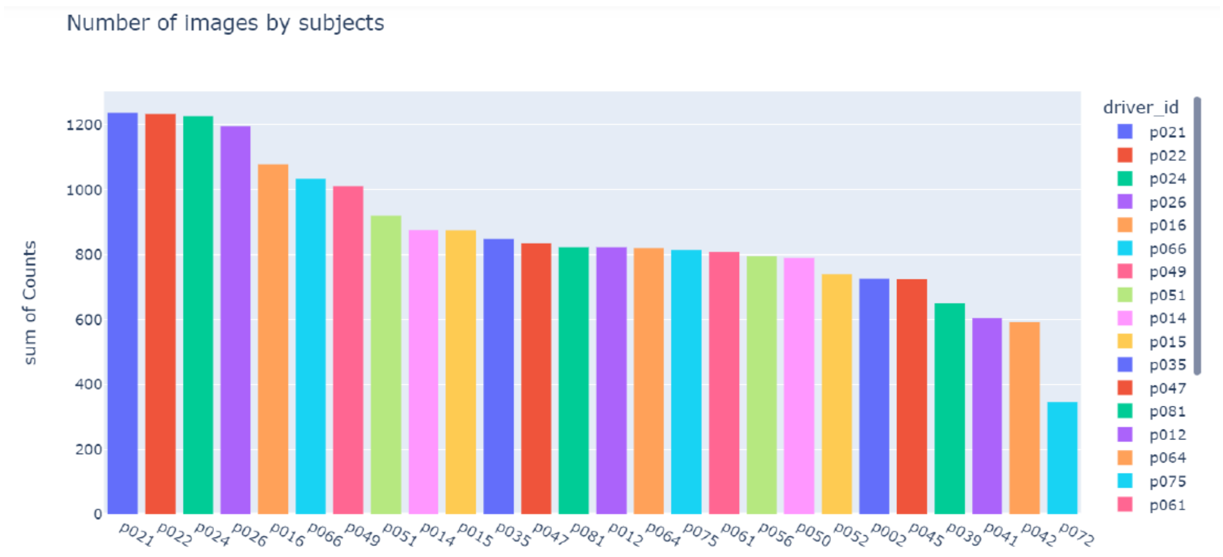
c0	6/1/2022 8:12 PM	File folder
c1	6/1/2022 8:10 PM	File folder
c2	6/1/2022 8:10 PM	File folder
c3	6/1/2022 8:10 PM	File folder
c4	6/1/2022 8:10 PM	File folder
c5	6/1/2022 8:11 PM	File folder
c6	6/1/2022 8:11 PM	File folder
c7	6/1/2022 8:11 PM	File folder
c8	6/1/2022 8:11 PM	File folder
c9	6/1/2022 8:12 PM	File folder



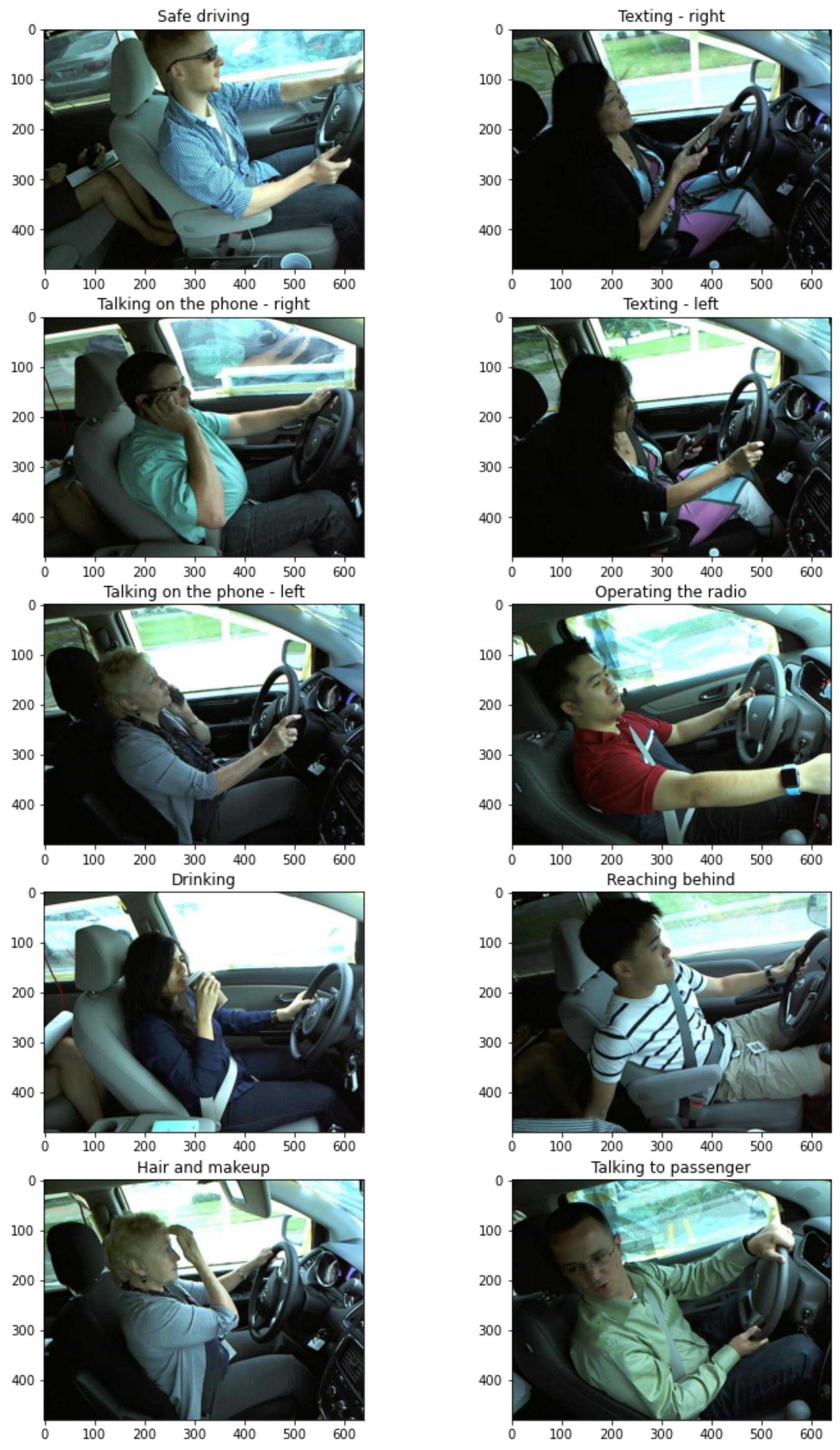
```
  0%|          | 0/10 [00:00<?, ?it/s]
Loading directory c0
 10%|█         | 1/10 [00:03<00:29,  3.31s/it]
Loading directory c1
 20%|██        | 2/10 [00:06<00:25,  3.20s/it]
Loading directory c2
 30%|███       | 3/10 [00:09<00:22,  3.23s/it]
Loading directory c3
 40%|████      | 4/10 [00:13<00:20,  3.40s/it]
Loading directory c4
 50%|█████     | 5/10 [00:16<00:17,  3.46s/it]
Loading directory c5
 60%|██████    | 6/10 [00:20<00:14,  3.64s/it]
Loading directory c6
 70%|███████   | 7/10 [00:24<00:11,  3.72s/it]
Loading directory c7
 80%|████████  | 8/10 [00:28<00:07,  3.56s/it]
Loading directory c8
 90%|█████████ | 9/10 [00:31<00:03,  3.40s/it]
Loading directory c9
100%|██████████| 10/10 [00:34<00:00,  3.44s/it]
 40%|████      | 200/500 [00:00<00:00, 605.59it/s]
```

B4:Data visualization (Exploratory Data analysis)





B5.Data preprocessing:

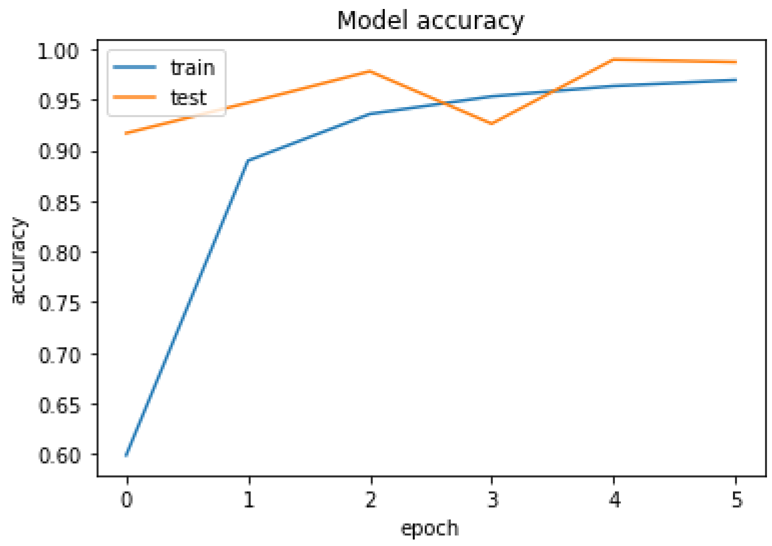


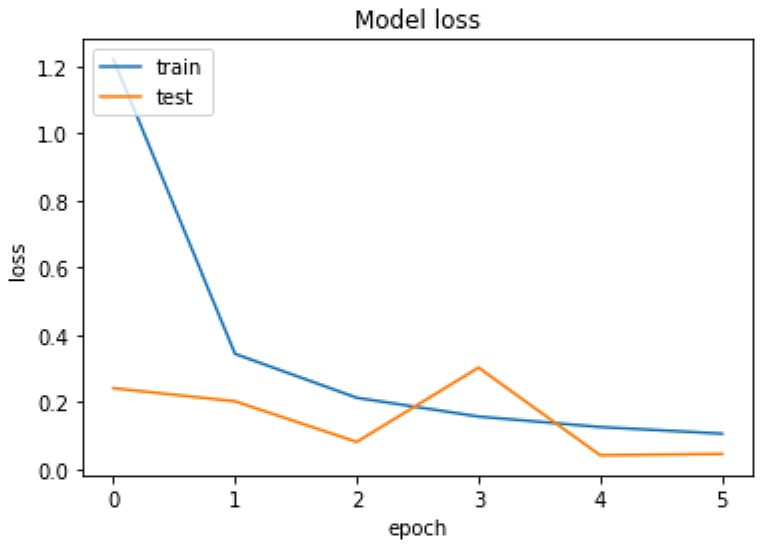
B6: Model configuration of vanilla CNN

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        320

 batch_normalization (BatchN  (None, 62, 62, 32)       128
 ormalization)

 conv2d_1 (Conv2D)           (None, 62, 62, 32)        9248

 batch_normalization_1 (Batc  (None, 62, 62, 32)       128
 hNormalization)

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 dropout (Dropout)           (None, 31, 31, 32)        0

 conv2d_2 (Conv2D)           (None, 31, 31, 64)        18496

 batch_normalization_2 (Batc  (None, 31, 31, 64)       256
 hNormalization)

 conv2d_3 (Conv2D)           (None, 31, 31, 64)        36928

 batch_normalization_3 (Batc  (None, 31, 31, 64)       256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 16, 16, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 16, 16, 64)        0
 conv2d_4 (Conv2D)           (None, 16, 16, 128)       73856

 batch_normalization_4 (Batc  (None, 16, 16, 128)      512
 hNormalization)

 conv2d_5 (Conv2D)           (None, 16, 16, 128)       147584

 batch_normalization_5 (Batc  (None, 16, 16, 128)      512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 8, 8, 128)        0
 2D)

 dropout_2 (Dropout)         (None, 8, 8, 128)         0

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 512)               4194816

 batch_normalization_6 (Batc  (None, 512)              2048
 hNormalization)

 dropout_3 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 128)               65664

 dropout_4 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 10)                1290

=================================================================
Total params: 4,552,042
Trainable params: 4,550,122
Non-trainable params: 1,920
_________________________________________________________________
```

B7:Training model

```
Epoch 1/6
449/449 [==============================] - 376s 829ms/step - loss: 1.2209
- accuracy: 0.5989 - val_loss: 0.2415 - val_accuracy: 0.9168
Epoch 2/6
449/449 [==============================] - 368s 819ms/step - loss: 0.3438
- accuracy: 0.8897 - val_loss: 0.2026 - val_accuracy: 0.9469
Epoch 3/6
449/449 [==============================] - 386s 860ms/step - loss: 0.2130
- accuracy: 0.9358 - val_loss: 0.0816 - val_accuracy: 0.9779
Epoch 4/6
449/449 [==============================] - 377s 840ms/step - loss: 0.1571
- accuracy: 0.9531 - val_loss: 0.3030 - val_accuracy: 0.9262
Epoch 5/6
449/449 [==============================] - 372s 829ms/step - loss: 0.1258
- accuracy: 0.9634 - val_loss: 0.0418 - val_accuracy: 0.9895
Epoch 6/6
449/449 [==============================] - 384s 855ms/step - loss: 0.1061
- accuracy: 0.9693 - val_loss: 0.0456 - val_accuracy: 0.9871
History of the training {'loss': [1.2208954095840454, 0.34383442997932434,
0.21301789581775665, 0.157071053981781, 0.12579722702503204,
0.1061093881726265], 'accuracy': [0.5989185571670532, 0.8896816968917847,
0.9357823729515076, 0.9531189203262329, 0.9633758664131165,
0.9692847728729248], 'val_loss': [0.24154655635356903,
0.20257622003555298, 0.08157601952552795, 0.3029603958129883,
0.04180203378200531, 0.045643556863069534], 'val_accuracy':
[0.9168338775634766, 0.9469342231750488, 0.9779264330863953,
0.926198422908783, 0.9895206093788147, 0.9870679974555969]
```


Model accuracy
accuracy
epoch
train
test
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
1
2
3
4
5

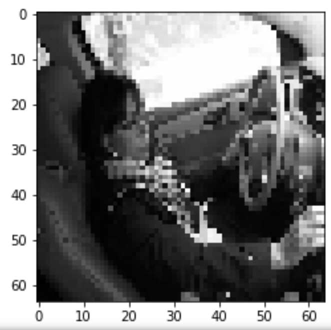

Model loss
loss
epoch
train
test
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5

B8 : Prediction

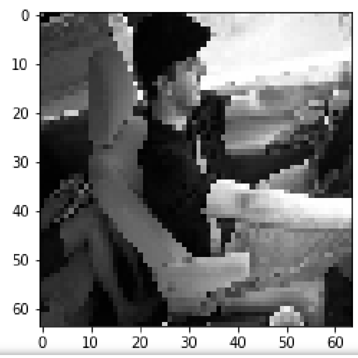
```
141/141 [==============================] - 19s 138ms/step - loss: 0.0456 - accuracy: 0.9871
```

```
Loss:  0.045643534511327744
Accuracy:  98.70679974555969  %
```

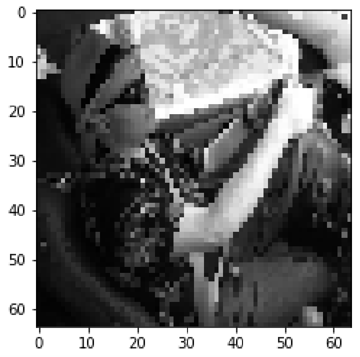
```
1/1 [==============================] - 0s 271ms/step
Y prediction: [[1.2900403e-14 8.8672833e-24 6.4304763e-17 7.6583756e-17 3.4441270e-17
  1.0000000e+00 9.6566719e-18 5.1638014e-17 6.5652257e-14 2.7455130e-11]]
Predicted: Operating the radio
```



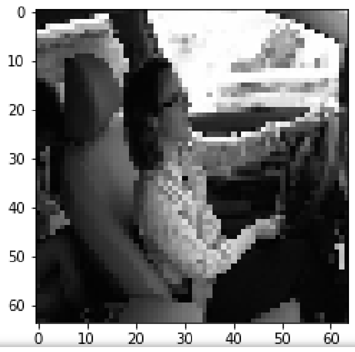
```
1/1 [==============================] - 0s 21ms/step
Y prediction: [[1.0228247e-14 2.7179788e-24 1.7436765e-15 1.6862242e-14 8.6801655e-16
  9.9999988e-01 1.4515789e-19 1.7629562e-18 8.1738812e-16 1.0003797e-07]]
Predicted: Operating the radio
```



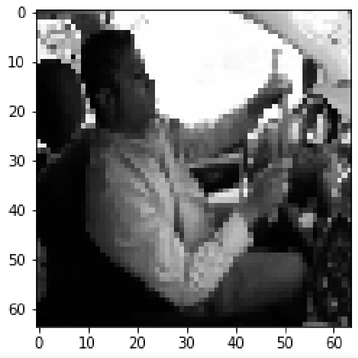
```
1/1 [==============================] - 0s 25ms/step
Y prediction: [[9.9999189e-01 1.0847873e-08 1.7288036e-11 2.1877956e-06 4.1129420e-07
  4.5848819e-10 4.7474616e-13 3.3711855e-12 1.2431403e-07 5.5207597e-06]]
Predicted: Safe driving
```



```
1/1 [==============================] - 0s 21ms/step
Y prediction: [[9.9897170e-01 6.0157228e-04 5.7869438e-05 7.2767165e-07 2.4754602e-08
  4.5576020e-05 3.8040422e-08 1.9479992e-07 3.1068604e-04 1.1633889e-05]]
Predicted: Safe driving
```

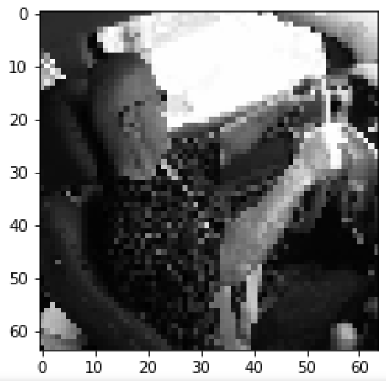


```
1/1 [==============================] - 0s 21ms/step
Y prediction: [[9.5438749e-01 7.0790725e-04 1.1269179e-06 4.3289725e-02 5.2355445e-04
  9.3189906e-04 1.2208337e-06 6.9908412e-09 7.6772354e-05 8.0277772e-05]]
Predicted: Safe driving
```

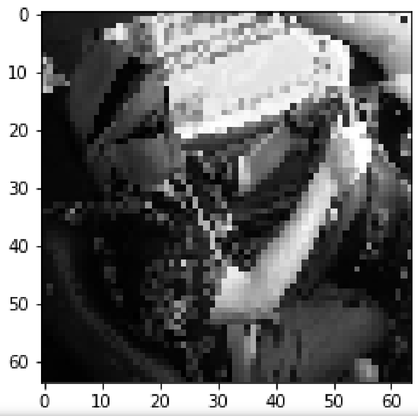


```
1/1 [==============================] - 0s 32ms/step
Y prediction: [[4.2196962e-13 2.4756135e-19 7.8589809e-15 1.8230115e-09 9.1342672e-10
  1.0000000e+00 9.7075448e-15 6.6345793e-16 1.0887172e-13 3.8716832e-11]]
Predicted: Operating the radio
```

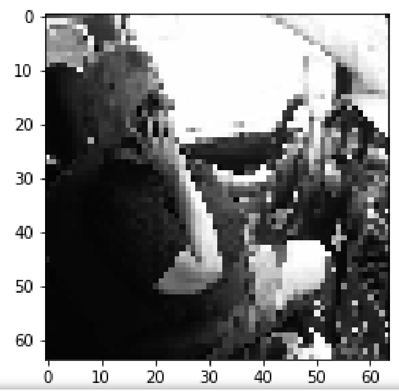
```
1/1 [==============================] - 0s 35ms/step
Y prediction: [[9.9999952e-01 6.9906268e-11 9.6975779e-09 5.1993176e-10 4.5278446e-07
  5.1979541e-08 2.6836210e-11 1.4242778e-13 5.8037043e-12 2.6787516e-08]]
Predicted: Safe driving
```



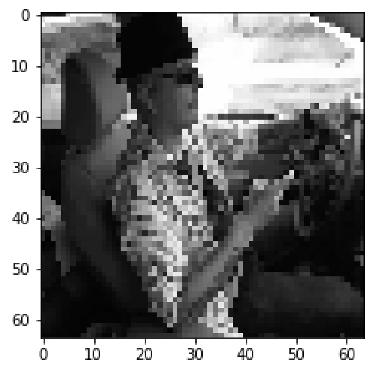
```
1/1 [==============================] - 0s 27ms/step
Y prediction: [[9.8741376e-01 4.9744779e-07 8.1304572e-09 1.8836209e-03 4.3606786e-03
  5.6219860e-06 7.6360408e-08 1.4229878e-08 4.9550631e-06 6.3308966e-03]]
Predicted: Safe driving
```



```
1/1 [==============================] - 0s 31ms/step
Y prediction: [[3.2157851e-03 1.8474795e-05 7.3784751e-01 4.8609655e-03 6.2413388e-03
  1.1446198e-02 1.0243221e-03 1.5157370e-01 4.5125775e-02 3.8645938e-02]]
Predicted: Talking on the phone - right
```



```
1/1 [==============================] - 0s 20ms/step
Y prediction: [[3.1019499e-05 9.9953771e-01 8.2020679e-06 4.9035066e-06 1.6032541e-08
  1.1738894e-09 2.7268903e-05 9.6845752e-06 3.1762934e-04 6.3652617e-05]]
Predicted: Texting - right
```



REFERENCES

[1] Abdul Jamsheed, V Dr. B Janet,Dr. U Srinivasulu Reddy,Real Time Detection of driver distraction using CNN,Conference,2020

[2]Arup Kanti Dey, Bharti Goel, Sriram Chellappan,Context-driven detection of distracted driving using images
from in-car cameras,Journal,2021

[3]Demeng Feng ,Yumeng Yue ,Machine Learning Techniques for Distracted Driver Detection,,2019

[4]Binbin Qin , Jiangbo Qian , Yu Xin, Baisong Liu, and Yihong Dong,Distracted Driver Detection Based on a CNN With Decreasing Filter Size,Journal,2021

[5]O. G. Basubeit, D. N. T. How, Y. C. Hou, K. S. M. Sahari,Distracted Driver Detection with Deep Convolutional Neural Network,Journal,2019

[6]Furkan Omerustaoglu, C. Okan Sakar , Gorkem Kar,Distracted driver detection by combining in-vehicle and image data using deep learning,Journal,2020

[7]Prof..Manya Gidwani Deep Ruparel Abhay Rajde Sahil Shah.,distracted driver detection,Journal,2020

[8]Bandar Alotaibi Munif Alotaibi,Distracted Driver Classification Using Deep Learning,article,2019

[9]Leonel Cuevas Valeriano; Paolo Napoletano; Raimondo Schettini,Recognition of driver distractions using deep learning,Conference,2018

[10]Annual report 2020, Technical Report, Ministry of Road Transport and Highways,2020.