Detecting Driver Distraction Using Deep-Learning Approach

Probed by:

Rawabi, Fatimah, Raghad, Ghadeer, Afrah.

Introduction

Statistics for the year 2011 estimated the number of daily deaths in the Kingdom due to car accidents as 20 deaths per day, (1) as well as behind deaths and disabilities in the world. It is expected that in 2030, car accidents will be the fourth cause of death worldwide. Also, ‹King Saud University conducted a study that used a detailed questionnaire to study the prevalence of scattering among drivers and its impact on accidents. The study was conducted in three regions, "Al-Wusta, Eastern and Western" on a sample of private car drivers whose number exceeded 1200 drivers, and the information was collected for six months, and the average age of drivers was 32 years. The study showed frightening results, as (33%) admitted that they were about to fall into at least one accident due to scattering during the study period (6 months), and (12%) of the drivers who were involved in a real traffic accident admitted that the cause of the accident was scattering during the study period. Leadership. A frightening result revealed by the study. (64%) admitted that they felt very scattering as a result of their concentration while driving at least once during the study. (2)

Another poll conducted in Britain showed that 11 percent of drivers admitted to falling scattering at least once while driving. A study conducted by the National Committee for scatter Disorders in the United States showed that scattering while driving was one of the causes of 36% of fatal accidents, while a report published by the Department of Transport and Environment in Britain showed that 20% of fatal accidents and serious accidents resulted from scattering while driving. The survey showed that only a fifth of drivers stop their cars to snooze when they feel very drowsy.

Researchers in the United States have found that one in 30 drivers on long roads feels very drowsy while driving, and statistics from the National Traffic and Security Administration in the United States have shown that driver's drowsiness while driving is the main cause of more than 100,000 accidents annually.

The second reason, which is no less dangerous than drowsiness, is the driver's distraction while driving. The most distracting driver is his use of a mobile phone while driving, a study revealed a high rate of traffic violations among drivers in the city of Riyadh and the reason for their use of mobile phones. (3)

As well as eating and drinking while driving, and a study conducted by the "Mas" women's car insurance company found that about 20% of female motorists admitted to wearing make-up during driving periods at least and applying foundation cream, while 3% of them admitted that they caused a collision with a pedestrian because makeup. (4). Many of the accidents that occur with women may be caused by distraction due to preoccupation with applying makeup.

Riyadh, Qassim, Makkah, Jeddah and Madinah are the most Saudi cities Traffic accidents are caused by distraction while driving. (5). And to reduce the problem of accidents caused by distraction while driving. We aim to build a system that anticipates driver distraction while driving. using deep learning.

Questions/Needs

- What type of things that distract drivers while they drive?
- What are the most dangerous distractions that cause accidents?
- Designing a mechanism to find the relationship between facial features and idleness that leads to distraction?

- Analyze the behavior of drivers to find causes of accidents.

Methodology

The method built in this work to detect a distraction for Drivers works on two main systems that are Convolutional Neural Networks (CNN) and Algorithms used. Convolutional Neural Networks are Deep Learning algorithms which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. Algorithms if the driver is Distracted or not, with high accuracy. The proposed methodology works in a number of phases and in the model, overview is given a general model of driver states.

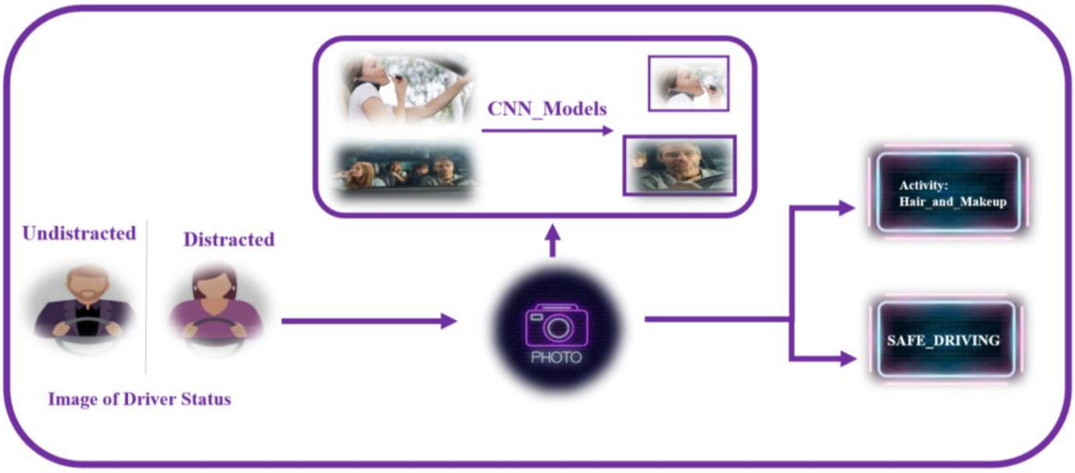


Figure1: General Model of the Driver States Detection System.

Basic stages of the proposed framework

The proposed approach is interlinked, certain experimental outputs become experimental inputs for other phases. The experimental results are conducted in the following steps.

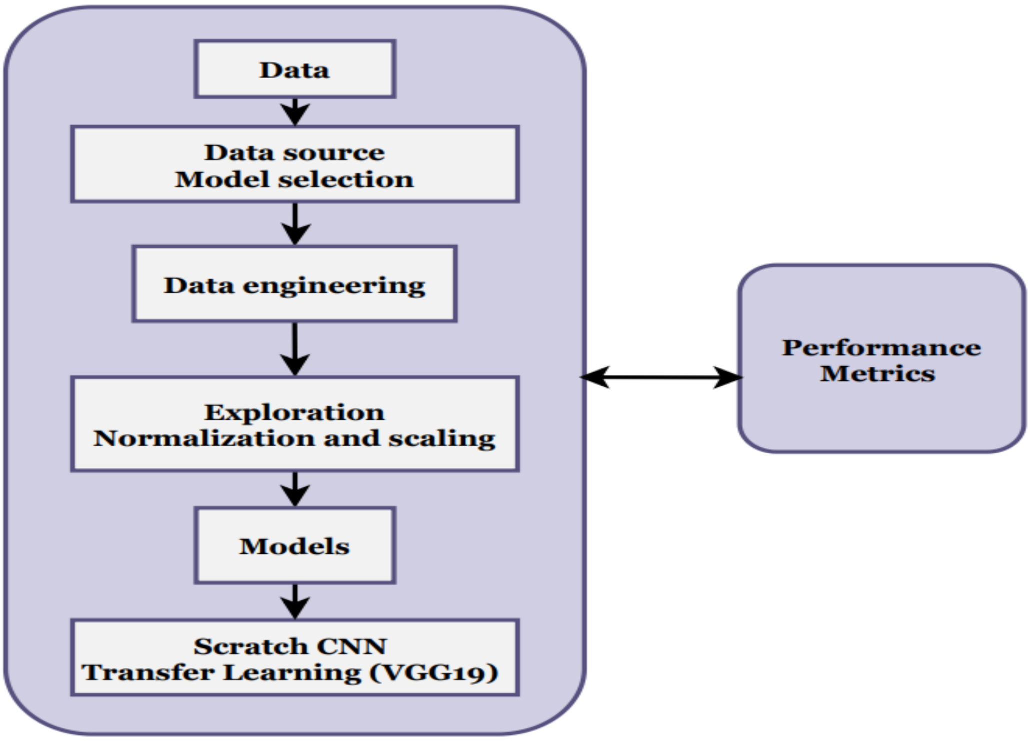


Figure2: The Proposal Methodology.

Step 1: Data sources and selection

The image dataset used in the project is readily available on the internet. Basically, machine learning projects are open-source projects, so anyone can download the dataset and build classification models. In this project, the data was taken from Kaggle for the driver images.

Step 2: Data Engineering

1) Exploration

The dataset gives us driver images, each taken in a car with a driver doing something in the car (texting, eating, talking on the phone, makeup, reaching behind, etc.). The data comprises around 102,152 images of people under different conditions. There 10 classes to classify are:

- 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.

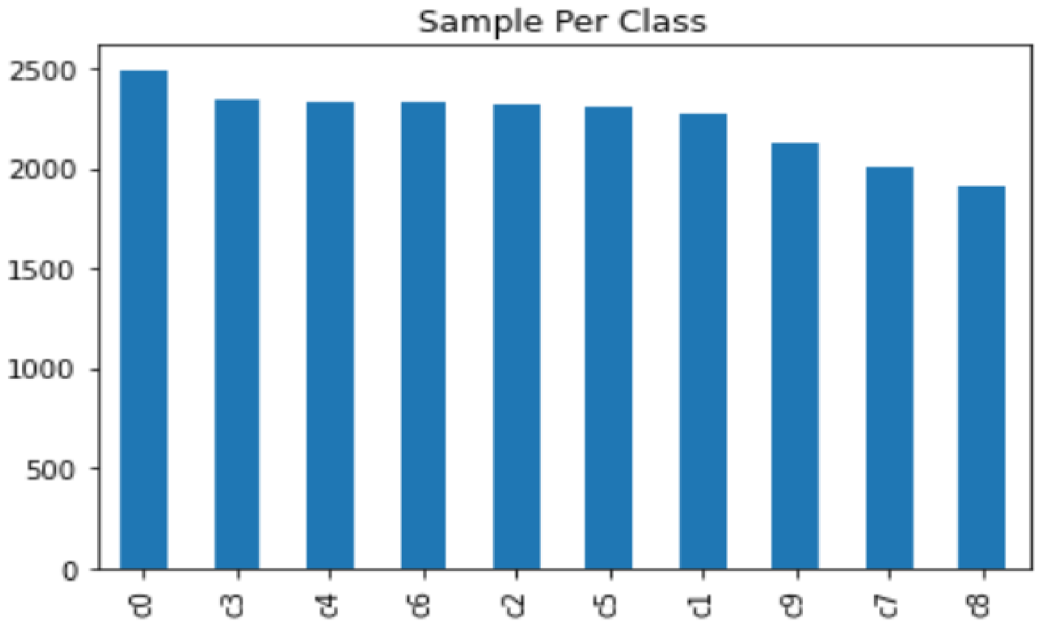


Figure 3: Bar chart to show size per class.

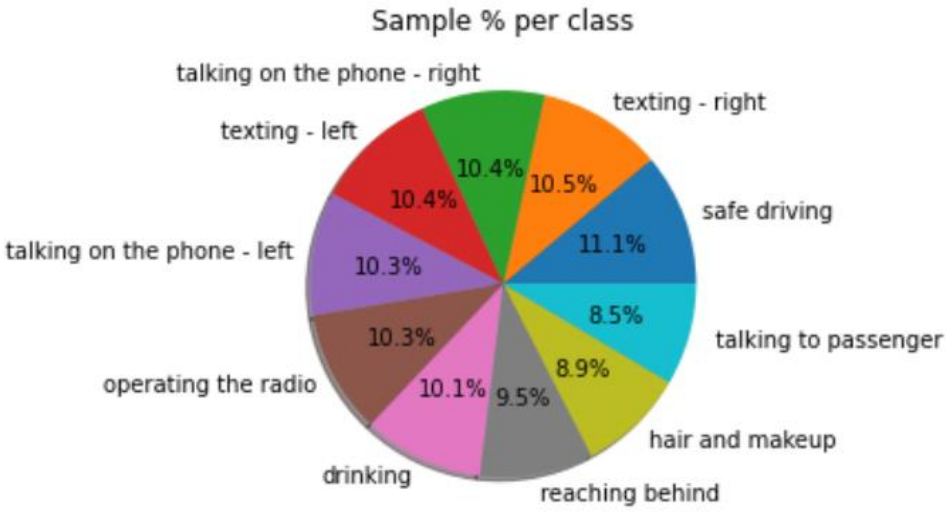


Figure 4: Pie chart to show percentage per class.

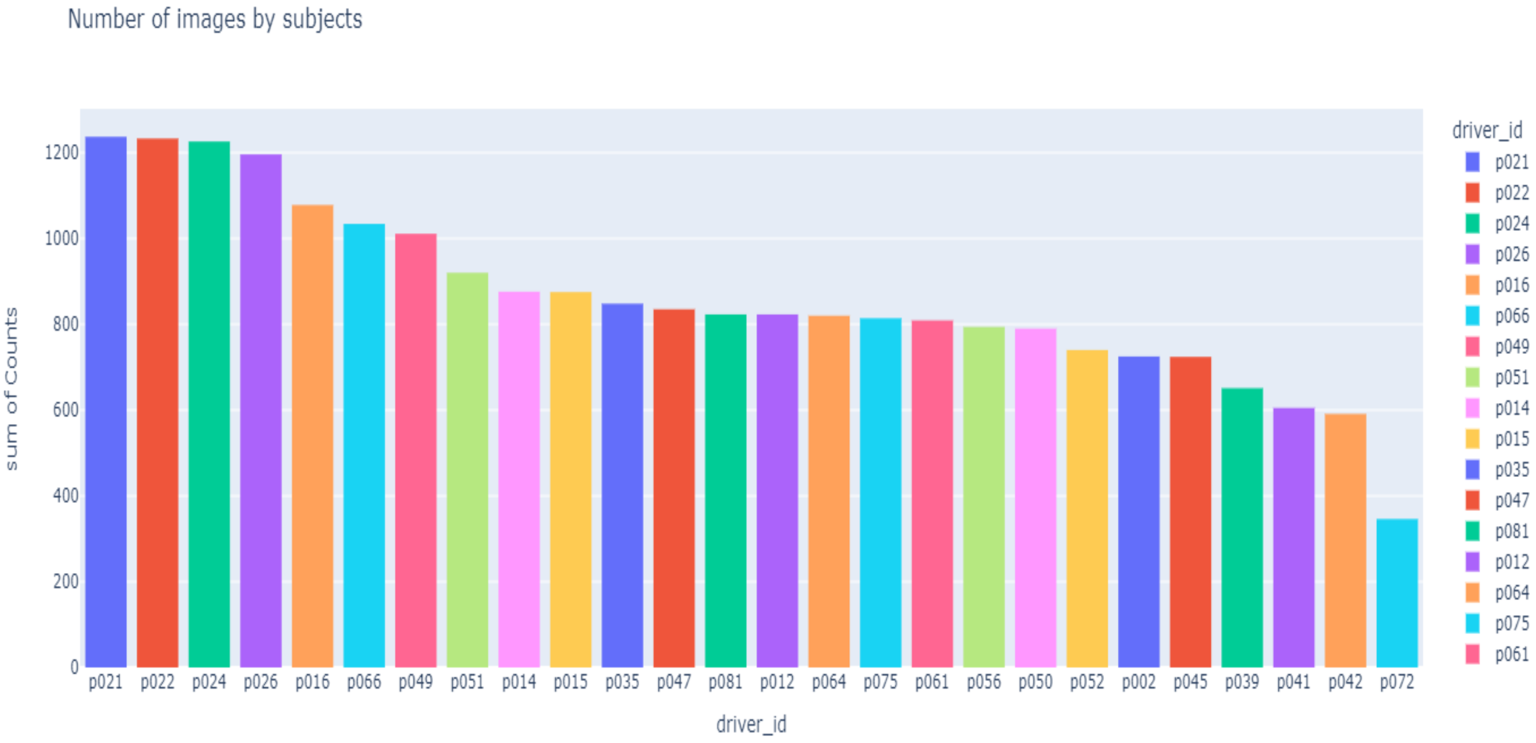


Figure 5: Number of images by subjects.

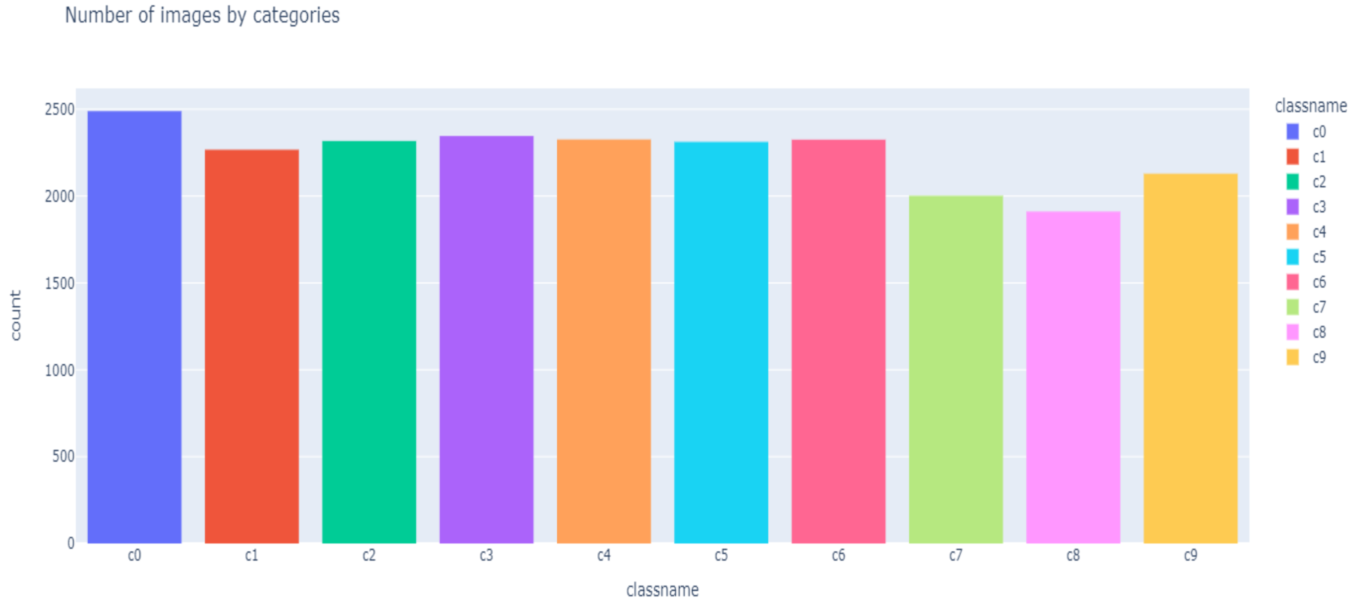


Figure6: Number of images by categories.

2) Normalization and Scaling

Raw data comes in all kinds of strange distributions so sometimes it is difficult to analyze and specially to create models without some preprocessing. By transforming our data, we are not only normalizing the observations but the residuals as well. Normalization makes training models less sensitive to the scale of features, so we can better solve for coefficients. The coefficients are statistical measures of the degree to which the changes to the value of one variable predict change to the value of another variable. Normalizing and scaling are two types of transformations that are important in data cleaning. In this project, we transformed each image to RGB. The next step was resizing images to (224,224) and applying them to preprocesses needed for VGG model input. After that, we normalized images to 255 - 0.5. With Unnormalized data, our network is tasked with learning how to combine inputs through a series of linear combinations and nonlinear activations, the parameters associated with each input will exist on different scales. Otherwise, by normalizing our inputs to a standard scale, we're allowing the network to more quickly learn the optimal parameters for each input node.

Step 3: Modeling

1) Scratch CNN

CNNs have become the go-to method for solving any image data challenge. Their use is being extended to video analytics as well but we'll keep the scope to image processing for now. Any data that has spatial relationships is ripe for applying CNN – let's just keep that in mind for now. The first layers of a neural network detect edges from an image. Deeper layers might be able to detect the cause of the objects and even deeper layers might detect the cause of complete objects (like a person's face).

Suppose we are given the below image and as you can see, there are many vertical and horizontal edges in the image. The first thing to do is to detect these edges: To complete our CNN, we need to give it the ability to actually make predictions. We'll do that by using the standard final layer for a multiclass classification problem: the SoftMax layer, a fully-connected (dense) layer that uses the SoftMax function as its activation. In the result section, we discussed that in detail.

2) Transfer Learning

The most significant benefit of transfer learning is its modularity and the fact that it builds on top of previously trained models, which have been trained on intensive datasets. Most supervised and unsupervised machine learning models are trained in isolation and on single datasets. Solving real-world problems with these applications necessitates significant resources and databases in the millions, which may not be accessible. A convolution neural network is a deep learning algorithm that takes in an input image, assigns characteristics to different aspects of the image, and is able to classify that image. The preprocessing required in the Convolution network is much lower as compared with other classification problems. When we train a deep convolutional neural network on a dataset the images are transferred through the network during the training process by adding multiple filters to the images on each layer. The filter matrix values are multiplied by the image activations on each sheet. To figure out the class the picture belongs to, the activations coming out of the final layer are used. Our aim is to find the optimal values for each of these filter matrices when we train a deep network such that the output activations can be used when an image is propagated across the network to correctly find the class to which the image belongs. Gradient descent is the method used to locate these filter matrix values.

Over the years, there are many variants of CNN architectures that have been developed to solve real-world problems. LeNet is the first successful application of CNNs and was developed by Yann Lacuna in the 1990s that was used to read zip codes, digits, etc. The latest 33 work is called LeNet-5 which is a 5-layer CNN that reaches 99.2 % accuracy on isolated character recognition.

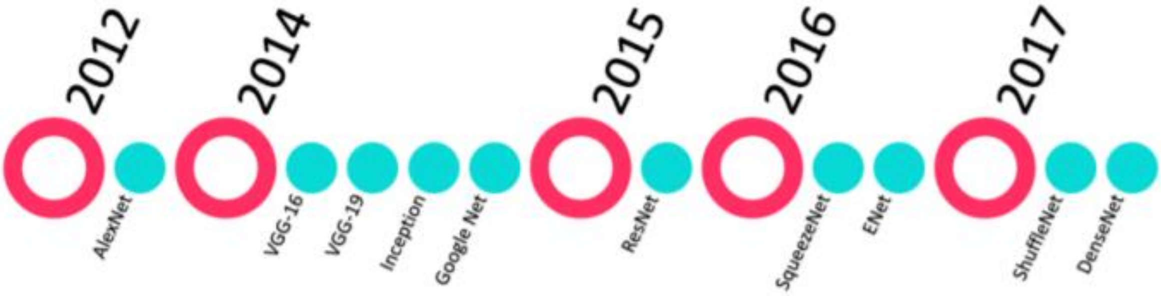


Figure 7: CNN architectures.

- **VGG19 architecture:**

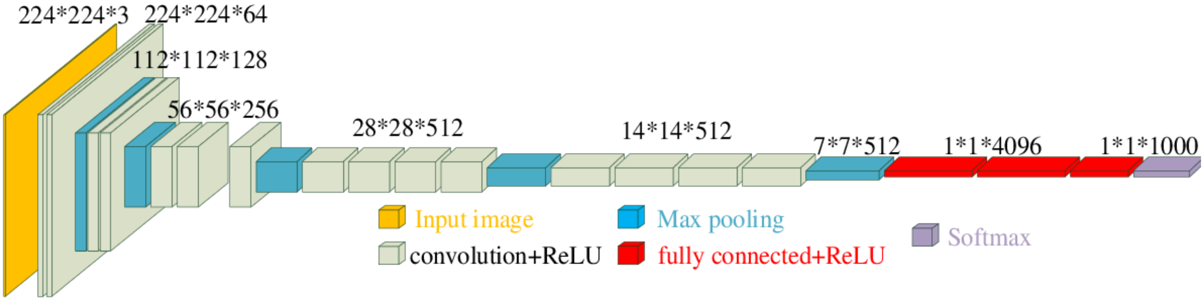


Figure 8: Siamese CNN architecture using VGG19 branches.

VGG CNN has six main structures, each of which is mainly composed of multiple connected convolutional layers and full-connected layers. The size of the convolutional kernel is 3*3, and the input size is 224*224*3. The number of layers is generally concentrated at 16~19. The VGG-19 model structure is shown in Figure 7.

So, in simple language VGG is a deep CNN used to classify images. The layers in VGG19 model are as follows:

- Conv3x3 (64)
- Conv3x3 (64)
- MaxPool
- Conv3x3 (128)

- Conv3x3 (128)
- MaxPool
- Conv3x3 (256)
- Conv3x3 (256)
- Conv3x3 (256)
- Conv3x3 (256)
- MaxPool
- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- MaxPool
- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- Conv3x3 (512)
- MaxPool
- Fully Connected (4096)
- Fully Connected (4096)
- Fully Connected (1000)
- SoftMax.

Step 4: Activation Function

In Glorot and Bengio's paper, one of the ideas put forward is that the vanishing/ exploding gradient problems were in part due to a poor choice of the activation function. Since then, most people have expected that if they use the sigmoid activation functions in biological neurons, this should be a good option, but it turns out that other activation functions behave better in deep neural networks. Besides, the logistic sigmoid function can cause a neural network to get stuck at the training time. Therefore, it is preferable to use the sigmoid activation function in only the output layers and avoid using it in all layers. In our case, we used the ReLU function rather than sigmoid. The ReLU function is another non-linear activation function that has gained popularity in the deep learning domain. ReLU stands for Rectified Linear Unit. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time. This means that the neurons will only be deactivated if the output of the linear transformation is less than 0.

Step 5: Optimization

Gradient Descent is a generic optimization algorithm that can identify optimal solutions to a wide variety of problems. The main idea behind Gradient Descent is to iteratively change parameters in order to minimize a cost function. Gradient descent is measuring the local gradient of the error function with regards to the parameter vector θ, and it goes in the direction of the descending gradient. Once the gradient is zero, you have attained a minimum; it begins with random numbers (this is known as random initialization), and then gradually improves, taking one small step at a time, each step attempting to decrease the cost function until the algorithm converges to a minimum.

The optimization used is Adaptive Moment Estimator (Adam) proposed by Diederik Kingma. Adam optimizer is a combination of RMSprop and Stochastic Gradient Descent that was proposed by Herbert and Sutton with momentum.

Step 6: Loss Function

The final goal in machine learning is to increase or decrease the “Objective function”. So, to optimize the algorithm, the error for the current state of the model must be estimated repeatedly. This requires the choice of an error function, conventionally called a loss function that can be used to compute to estimate the prediction given by the model in terms of generalizability. In our project, we used binary Cross-Entropy Loss. Cross-entropy is the default loss function to use for binary classification problems. It is intended for use with binary classification where the target values are in the set (0, 1). Mathematically, it is the preferred loss function under the inference framework of maximum likelihood. We specified the Cross-entropy of the loss function in Keras by ‘binary_crossentropy ‘when compiling the models.

Step 7: Architecture

A fixed size of (224 * 224) RGB image was given as input to this network which means that the matrix was of shape (224,224,3).

The preprocessing that was done is that they subtracted the mean RGB value from each pixel, computed over the whole training set. Used kernels of (3 * 3) size with a stride size of 1 pixel, this enabled them to cover the whole notion of the image. spatial padding was used to preserve the spatial resolution of the image. max pooling was performed over a 2 * 2pixel windows with stride 2.

This was followed by Rectified linear unit (ReLu) to introduce non-linearity to make the model classify better and to improve computational time as the previous models used tanh or sigmoid functions that proved much better than those.

implemented three fully connected layers from which first two were of size 4096 and after that a layer with 1000 channels for 1000-way ILSVRC classification and the final layer is a SoftMax function.

Step 7: Performance metrics

It is very important to evaluate the classification performance in image classification studies to scientifically support the results of the study. Otherwise, the classification study would remain incomplete and weak. There are various performance evaluation metrics that have been used for a long time in image classification studies and have become standard performance evaluation metrics in similar studies. These are accuracy, precision, recall, and F1-Score. These metrics that are accepted as standard performance evaluation metrics in image classification studies are also used to measure the accuracy and reliability of the classification process.

*Accuracy = TruePositive (TP) + TrueNegative (TN)/ (Total No of Samples) * 100 (1)*

Precision estimates how many positive labels we had predicted.

Precision = TruePositive (TP)/ (TruePositive (TP) + FalsePositive (FP) (2)

Recall evaluates how many positive labels we had correctly predicted from our data

Recall = TruePositive (TP)/ (TruePositive (TP) + FalseNegative (FN) (3)

While F1-Score is the weighted mean of Recall and Precision

F1 − Score = 2 × (Precision × Recall)/ (Precision + Recall) (4)

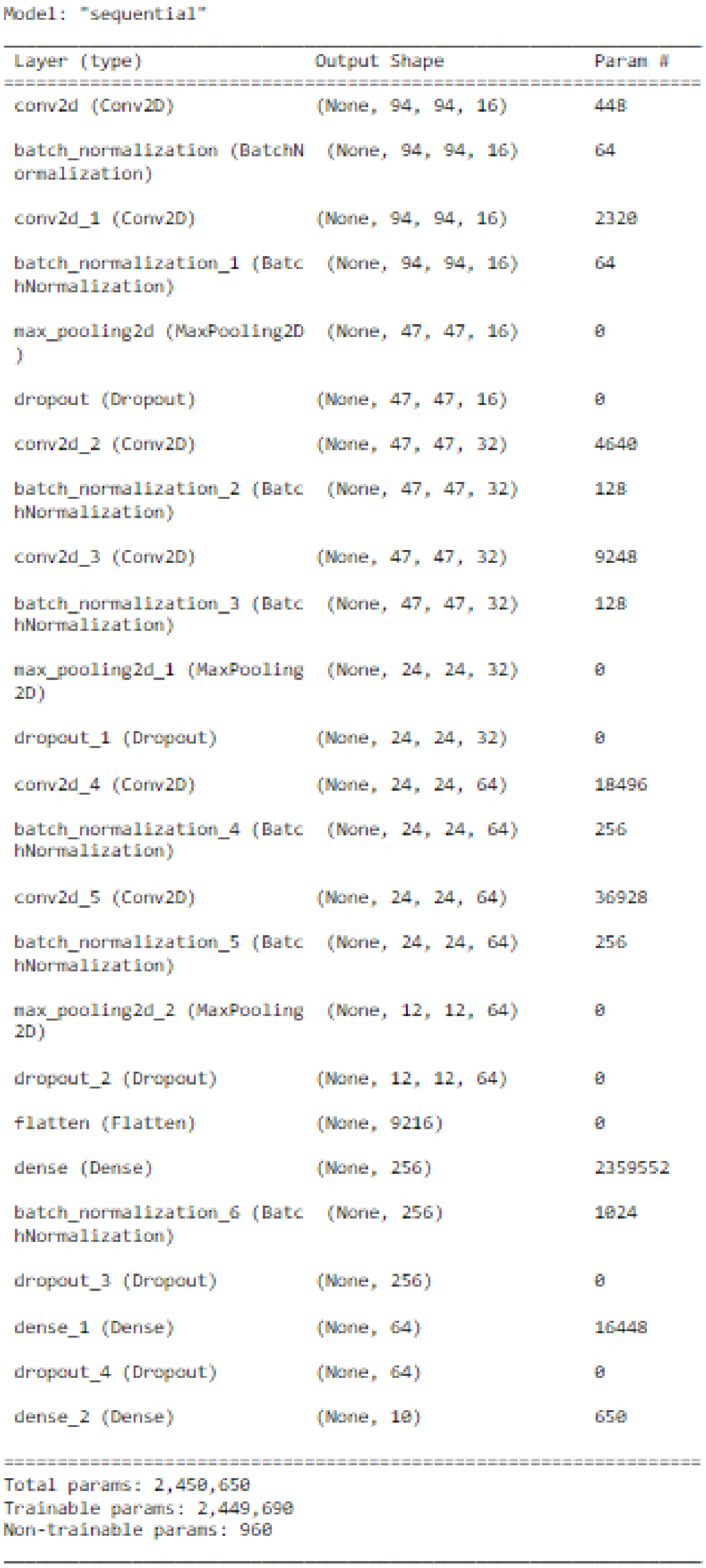
the true positive rate and false positive rate.

TruePositiveRate (T PR) = T ruePositive (T P)/ (T ruePositive (T P) + FalseNegative (FP)) (5)

FalsePositiveRate (FPR) = FalsePositive (FP)/ (FalsePositive (FP) + T rueNegative (T N)) (6)

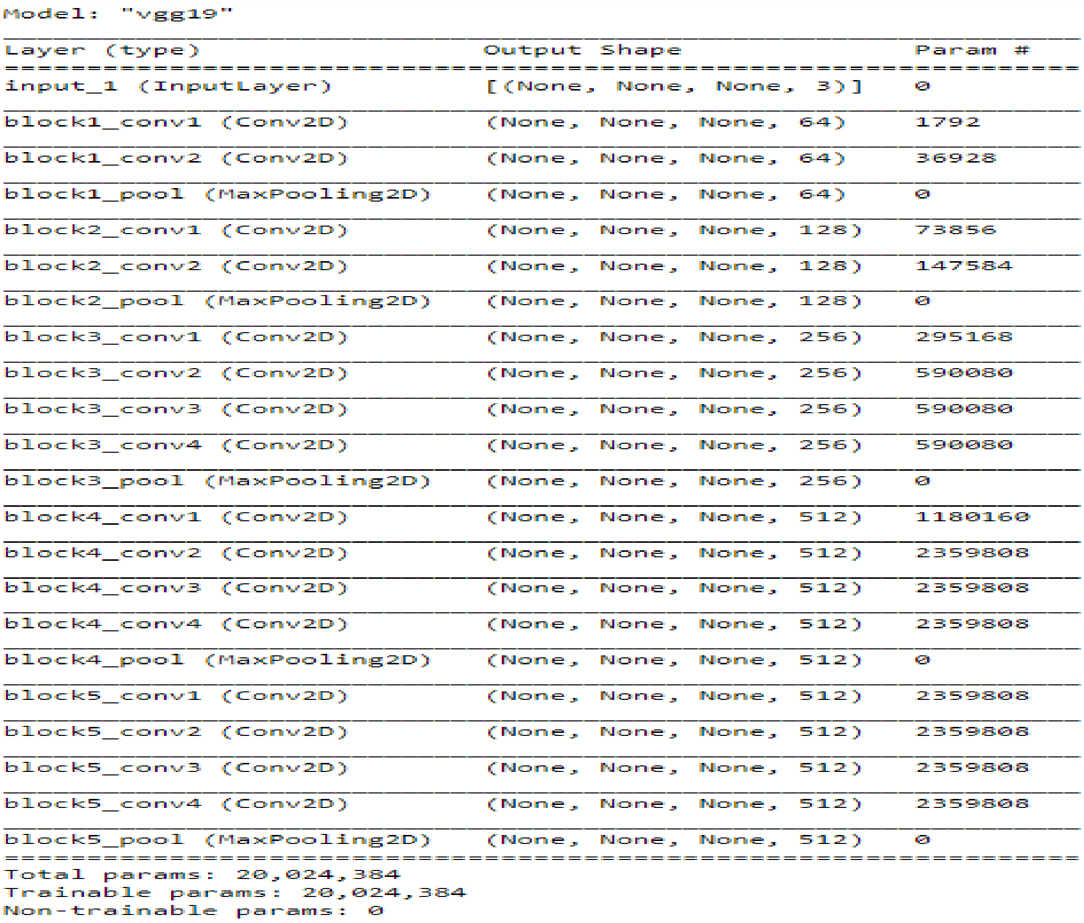
Results and discussion

In this section, we will discuss the results obtained from the first stages. From the stage of data acquisition to the data engineering and the model selection. And after obtaining this data, it was ready to enter the model, and then it was subjected to training. After this work, we discuss the test results of the algorithm and make sure that the algorithm works well.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 94, 94, 16)        448       
                                                                 
 batch_normalization (BatchN  (None, 94, 94, 16)       64        
 ormalization)                                                   
                                                                 
 conv2d_1 (Conv2D)           (None, 94, 94, 16)        2320      
                                                                 
 batch_normalization_1 (Batc  (None, 94, 94, 16)       64        
 hNormalization)                                                 
                                                                 
 max_pooling2d (MaxPooling2D  (None, 47, 47, 16)       0         
 )                                                               
                                                                 
 dropout (Dropout)           (None, 47, 47, 16)        0         
                                                                 
 conv2d_2 (Conv2D)           (None, 47, 47, 32)        4640      
                                                                 
 batch_normalization_2 (Batc  (None, 47, 47, 32)       128       
 hNormalization)                                                 
                                                                 
 conv2d_3 (Conv2D)           (None, 47, 47, 32)        9248      
                                                                 
 batch_normalization_3 (Batc  (None, 47, 47, 32)       128       
 hNormalization)                                                 
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 24, 24, 32)       0         
 2D)                                                             
                                                                 
 dropout_1 (Dropout)         (None, 24, 24, 32)        0         
                                                                 
 conv2d_4 (Conv2D)           (None, 24, 24, 64)        18496     
                                                                 
 batch_normalization_4 (Batc  (None, 24, 24, 64)       256       
 hNormalization)                                                 
                                                                 
 conv2d_5 (Conv2D)           (None, 24, 24, 64)        36928     
                                                                 
 batch_normalization_5 (Batc  (None, 24, 24, 64)       256       
 hNormalization)                                                 
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 12, 12, 64)       0         
 2D)                                                             
                                                                 
 dropout_2 (Dropout)         (None, 12, 12, 64)        0         
                                                                 
 flatten (Flatten)           (None, 9216)              0         
                                                                 
 dense (Dense)               (None, 256)               2359552   
                                                                 
 batch_normalization_6 (Batc  (None, 256)              1024      
 hNormalization)                                                 
                                                                 
 dropout_3 (Dropout)         (None, 256)               0         
                                                                 
 dense_1 (Dense)             (None, 64)                16448     
                                                                 
 dropout_4 (Dropout)         (None, 64)                0         
                                                                 
 dense_2 (Dense)             (None, 10)                650       
                                                                 
=================================================================
Total params: 2,450,650
Trainable params: 2,449,690
Non-trainable params: 960
_________________________________________________________________
```

Figure 9: summary Scratch CNN.



```
Model: "vgg19"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
input_1 (InputLayer)        [(None, None, None, 3)]   0
_________________________________________________________________
block1_conv1 (Conv2D)       (None, None, None, 64)    1792
_________________________________________________________________
block1_conv2 (Conv2D)       (None, None, None, 64)    36928
_________________________________________________________________
block1_pool (MaxPooling2D)  (None, None, None, 64)    0
_________________________________________________________________
block2_conv1 (Conv2D)       (None, None, None, 128)   73856
_________________________________________________________________
block2_conv2 (Conv2D)       (None, None, None, 128)   147584
_________________________________________________________________
block2_pool (MaxPooling2D)  (None, None, None, 128)   0
_________________________________________________________________
block3_conv1 (Conv2D)       (None, None, None, 256)   295168
_________________________________________________________________
block3_conv2 (Conv2D)       (None, None, None, 256)   590080
_________________________________________________________________
block3_conv3 (Conv2D)       (None, None, None, 256)   590080
_________________________________________________________________
block3_conv4 (Conv2D)       (None, None, None, 256)   590080
_________________________________________________________________
block3_pool (MaxPooling2D)  (None, None, None, 256)   0
_________________________________________________________________
block4_conv1 (Conv2D)       (None, None, None, 512)   1180160
_________________________________________________________________
block4_conv2 (Conv2D)       (None, None, None, 512)   2359808
_________________________________________________________________
block4_conv3 (Conv2D)       (None, None, None, 512)   2359808
_________________________________________________________________
block4_conv4 (Conv2D)       (None, None, None, 512)   2359808
_________________________________________________________________
block4_pool (MaxPooling2D)  (None, None, None, 512)   0
_________________________________________________________________
block5_conv1 (Conv2D)       (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv2 (Conv2D)       (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv3 (Conv2D)       (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv4 (Conv2D)       (None, None, None, 512)   2359808
_________________________________________________________________
block5_pool (MaxPooling2D)  (None, None, None, 512)   0
=================================================================
Total params: 20,024,384
Trainable params: 20,024,384
Non-trainable params: 0
```

Figure10: Summary VGG19.

Table1: Comparison of models.

Model	**Scratch CNN**	**VGG19**
Accuracy	**98 %**	**99 %**
Loss	**0.41**	**0.028**

Comparison of Scratch CNN Model and VGG19. As shown in the tables, the highest accuracy in the model is VGG19 with a rate of 99% compared to the other models, and that is why we chose it.

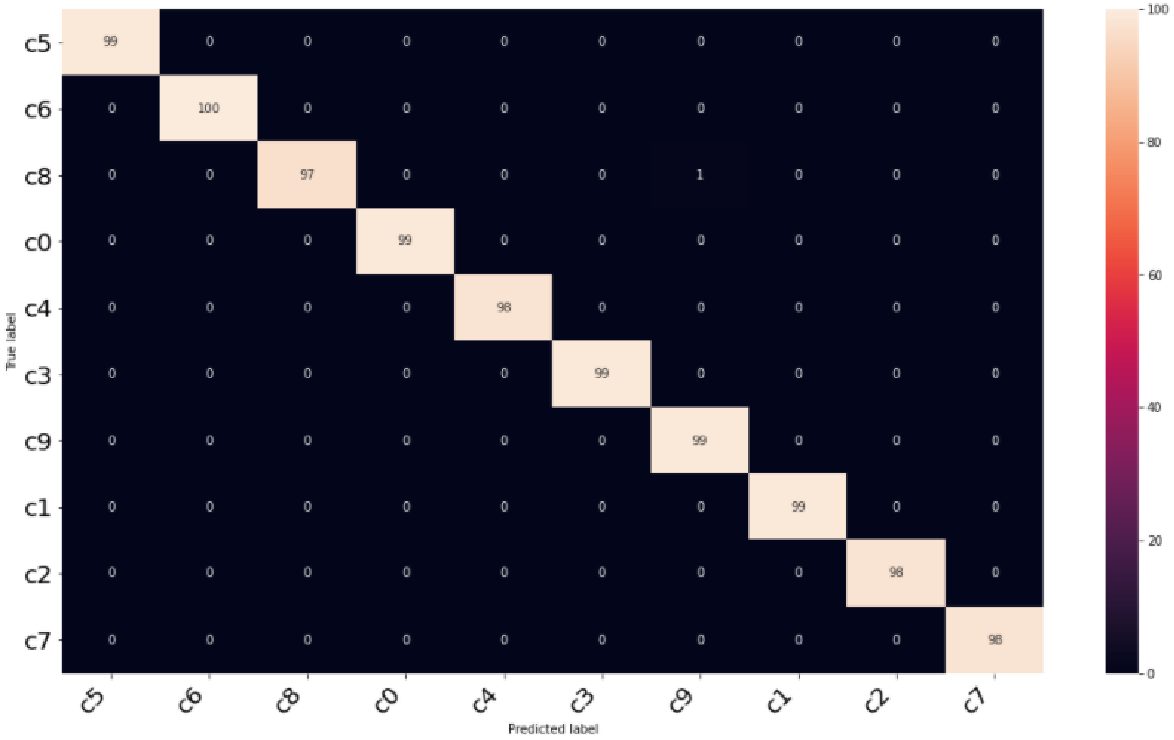


Figure 11: Heat Map.

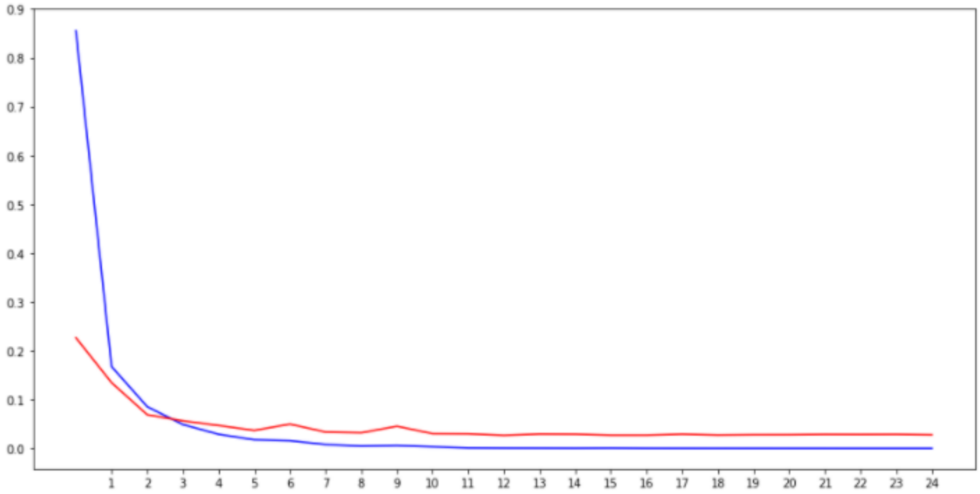


Figure12: Loss

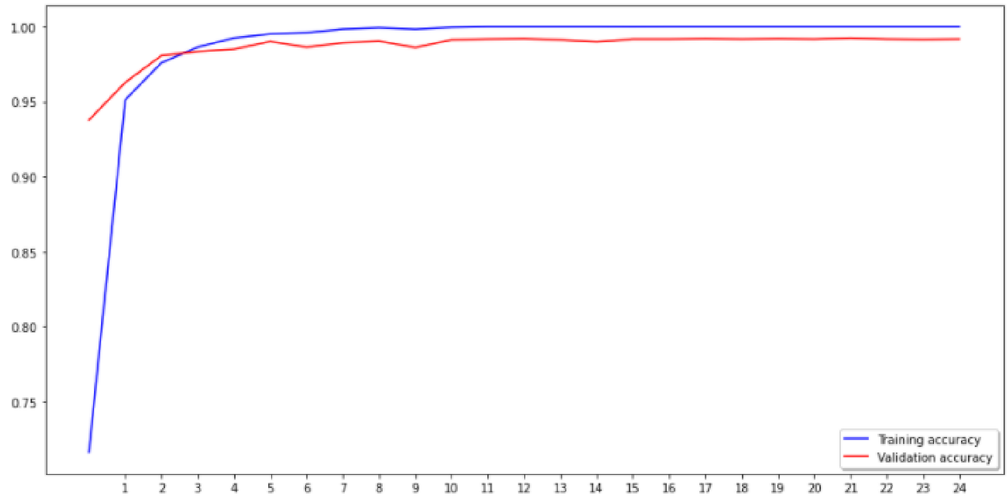


Figure 13: Accuracy

Table2: Accuracies rate.

Model	Accuracy	Precision	Recall	F1 score
VGG19	0.991750	0.991785	0.991750	0.991750

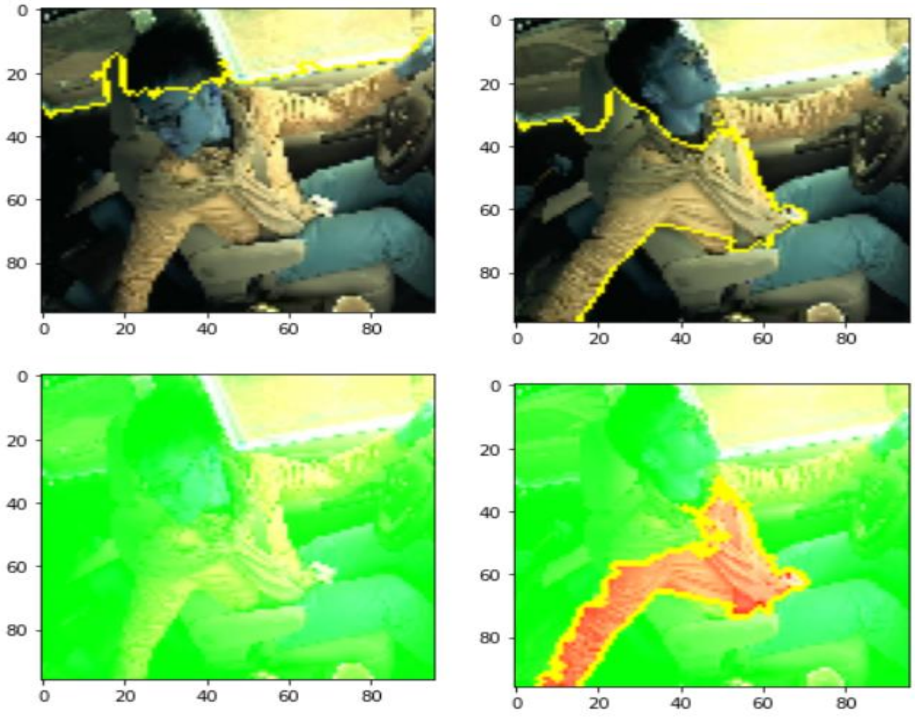


Figure14: Example correctly classified with a high likelihood of class membership.

After we worked out the algorithm and found out the accuracy on its basis, we applied the VGG19 numbers, and when we applied them to the train and test, we concluded that you could not predict and that we fell into the overfitting and misled.

After we encountered some problems, we came up with a solution. It is that we define the target area and extract the future from it. Then we made cuts to the place that moved and focused our attention on it. Hence, it is predicted after what we entered into the algorithm. He became completely aware of the body. Then he became aware of the head and hands, and the accuracy increased, and this was the center of our attention and the goal we wanted to reach.

Sample of Train

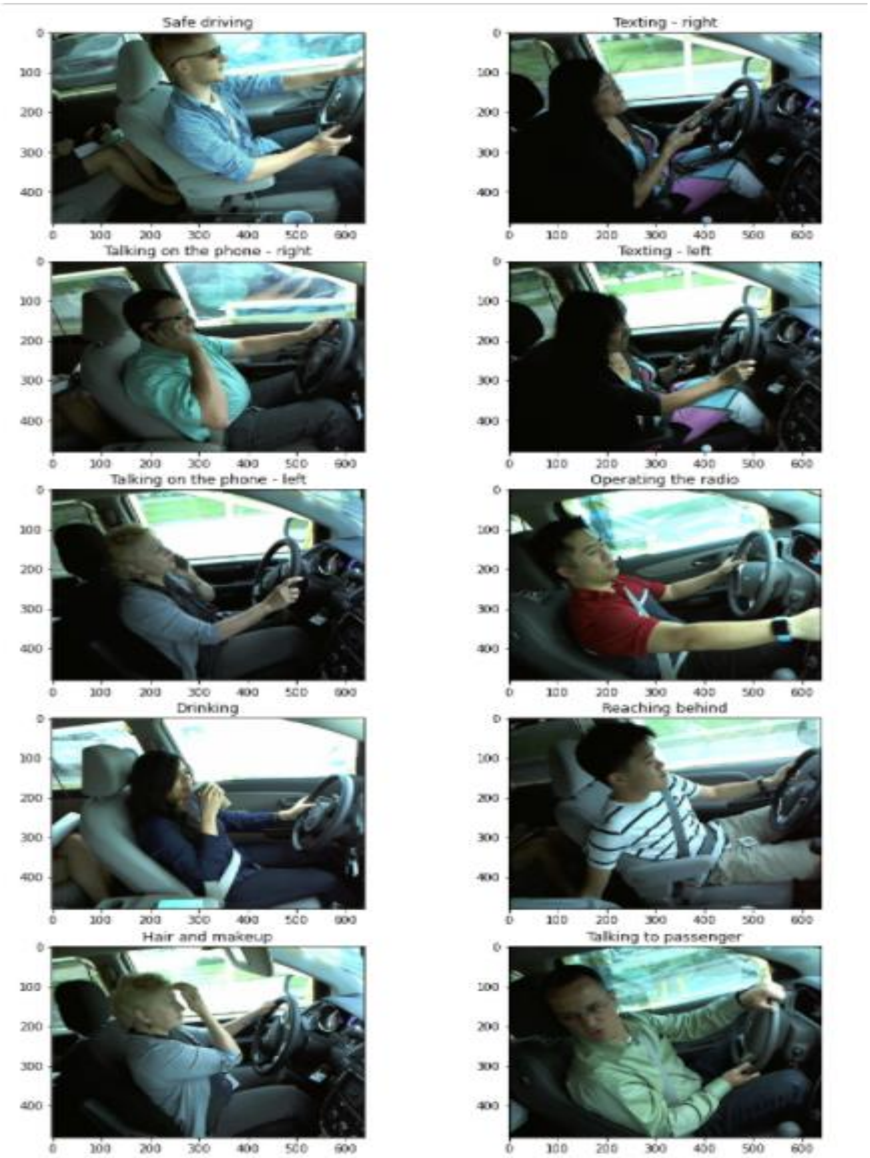
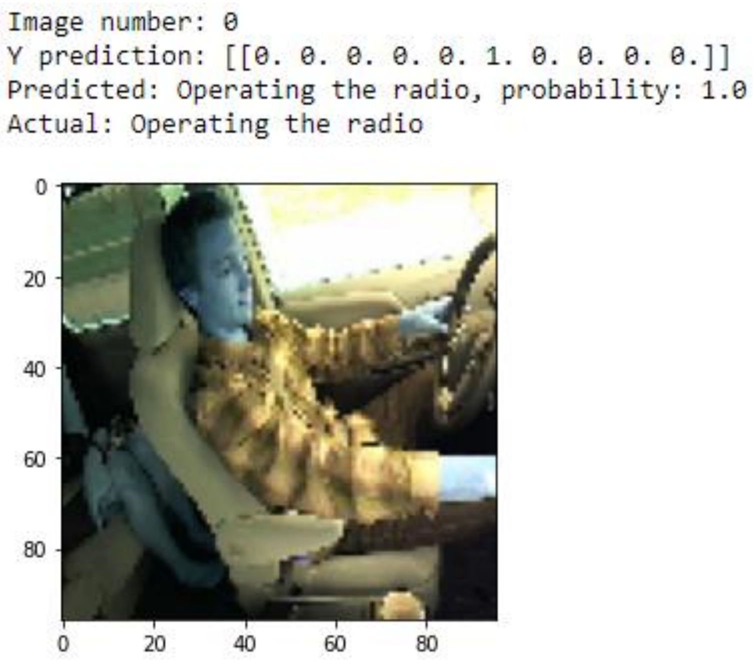


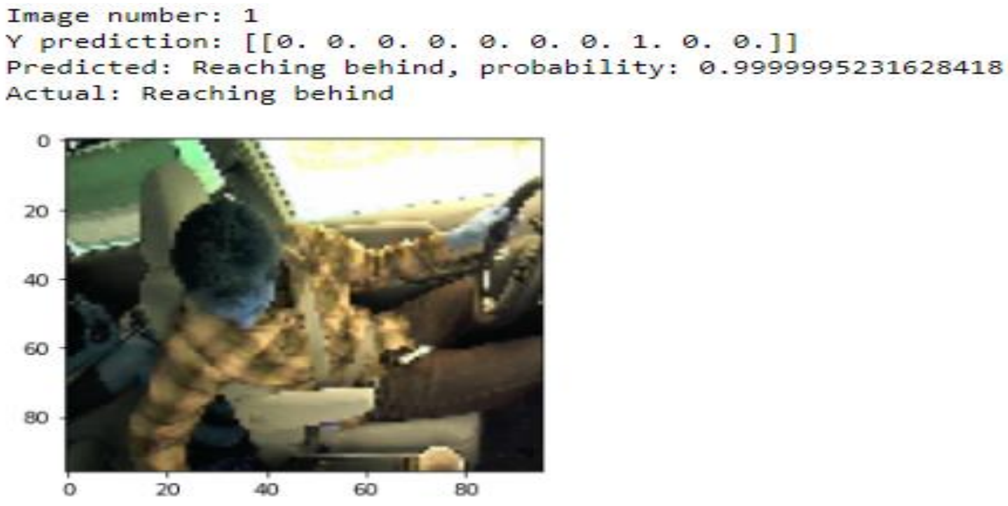
Figure15: Training Dataset.

Sample of Data



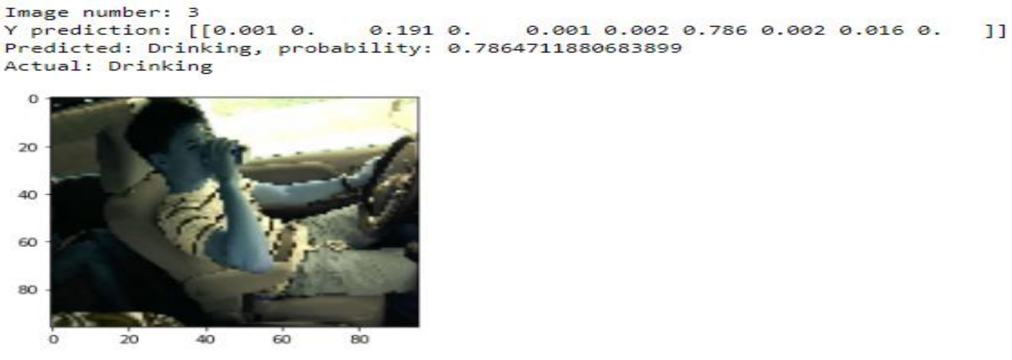
```
Image number: 0
Y prediction: [[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]]
Predicted: Operating the radio, probability: 1.0
Actual: Operating the radio
```

Figure16: Image Test1.



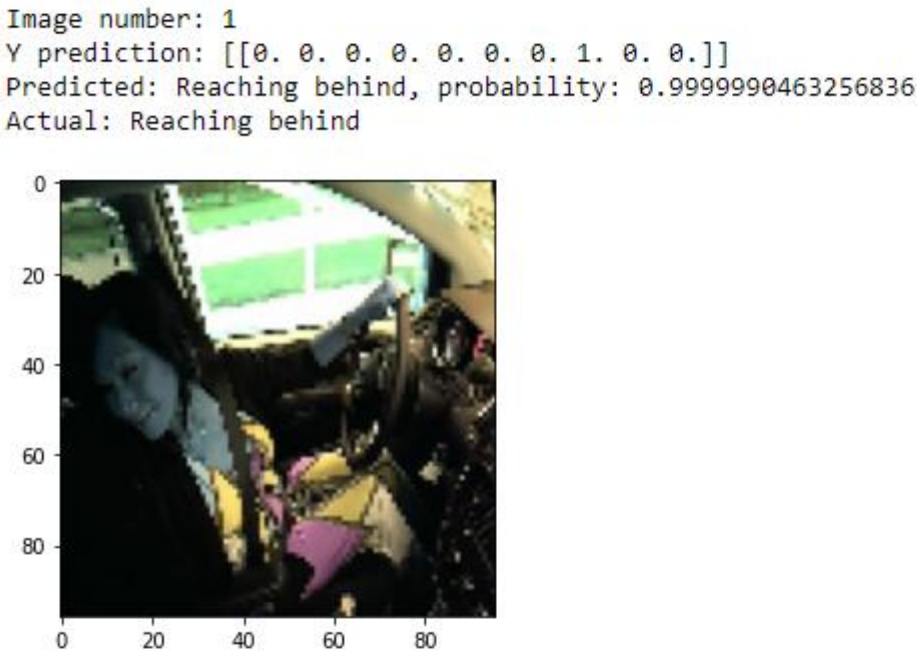
```
Image number: 1
Y prediction: [[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]]
Predicted: Reaching behind, probability: 0.9999995231628418
Actual: Reaching behind
```

Figure17: Image Test2.



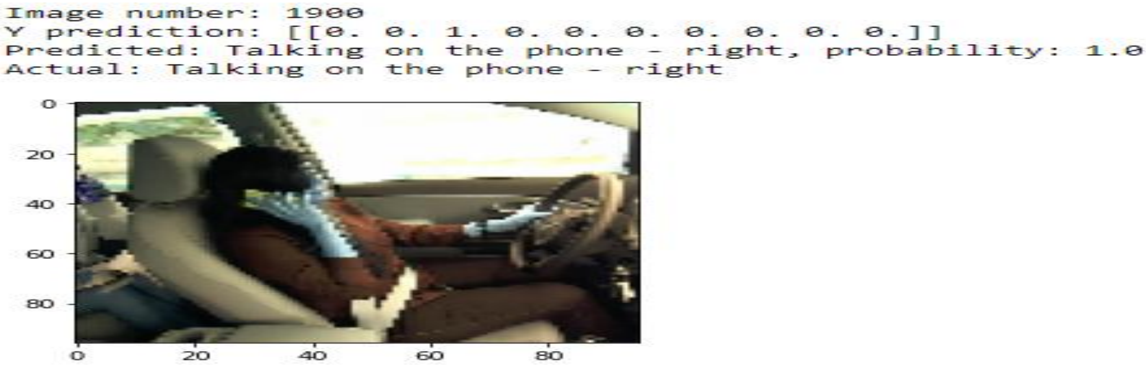
```
Image number: 3
Y prediction: [[0.001 0.    0.191 0.    0.001 0.002 0.786 0.002 0.016 0.   ]]
Predicted: Drinking, probability: 0.7864711880683899
Actual: Drinking
```

Figure18: Image Test3.



```
Image number: 1
Y prediction: [[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]]
Predicted: Reaching behind, probability: 0.999999463256836
Actual: Reaching behind
```

Figure19: Image Test4.



```
Image number: 1900
Y prediction: [[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]]
Predicted: Talking on the phone - right, probability: 1.0
Actual: Talking on the phone - right
```

Figure20: Image Test5.

References

(1) The prevalence of drowsiness as a risk factor among motorists involved in traffic accidents; A local study on a sample of drivers in the Kingdom of Saudi Arabiahttps://prod.kau.edu.sa/centers/spc/jkau/Data2/Review_Artical_ar.aspx?No=3574

(2) University Center for Sleep Medicine and Research at King Saud University College of Medicine.

(3) The prevalence of seatbelt and mobile phone use among drivers in Riyadh, Saudi Arabia: An observational study.https://www.sciencedirect.com/science/article/abs/pii/S002243751730823X?via%3Dihub

(4)A study conducted by the "MAS" women's insurance company.

(5) Ministry of Interior—Kingdom of Saudi Arabia. Available online:

(6)https://www.moi.gov.sa

(7)https://medium.com/@isalindgren313/transformations-scaling-and-normalization-420b2be12300

(8)https://iopscience.iop.org/article/10.1088/1742-6596/1518/1/012041/pdf

(9)https://iq.opengenus.org/vgg19-architecture/

(10)https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/