Driver's Distraction Detection using Deep Learning and Computer Vision

Quick Savajiyani, Vatsal Ketan Patel
A20451378, A20458061
Computer Science, Illinois Institute of Technology
Computer Science, Illinois Institute of Technology

***Abstract*- Automobile crashes have tremendously increased these days and first the foremost cause for motor vehicle accidents and incidents is Driver's distraction. Driving requires an intensive amount of concentration otherwise the results can be fatal. Distracted driving is the activity that takes drivers attention from the road and these activities may be texting, calling, or any other distraction line grooming, eating and operating the music system of the car. Yet, most motor vehicles have no system in place to assist the driver when he/she is feeling drowsy, fatigued or distracted. In this project we aim to develop a system which detects the driver's distraction by using deep learning and computer vision. Here we are going to use deep learning techniques for predicting sample images by its different categories to help in the better detection and reduce false positives and negatives.**

I. INTRODUCTION

For most people these days driving is a daily activity which does not require much thought or consideration, but the truth is that driving requires a lot of concentration and if the driver is distracted it can lead to a fatal accident. The reality is that each day in the United States, approximately 9 people are killed and more than 1000 injured in the crashes that involve distracted driving. Distracted driving is where drivers are doing other activities which diverts their attention from the road and are not concentrating. These activities not only include texting or talking on the phone but it also includes other activities like eating and drinking, talking to other people in the car, feeling sleepy or drowsy, adjusting the audio system or setting up the GPS system all comes under distracted driving. [1]

Several studies show that there are 3 main types of distractions:
1. Visual distractions (driver's eyes off the road) - This is the type of distraction where the driver takes his eyes off the road which involves as simple as checking the navigation system.
2. Manual distractions (driver's hands off the wheel) - This is the type of distraction where the driver removes one or both of his/her hands from the steering wheel. It involves
3. Cognitive distractions (driver's mind off the driving task) - This is the type of distraction which causes the driver's mind and focus to divert from the road to something happening in the car.

A study suggests that 2841 people died of distracted driving in the year 2018 and most of them were teens and the people with young children or pets in the car. Not all cars have technology to detect driver distraction. In the world of machine learning and computer vision there are a lot of technological advancements taking place from the health industry to detect different diseases to the automotive transportation all at a very high pace. These developing fields have changed the global scenarios and have paved ways for technologies that we had never thought of a few years ago. The autonomous driving industry is in the boom these days and the coming future. Imagine a world where accidents can be avoided and very less traffic all these is possible because of the boom in the above-mentioned fields. In this project we are trying to develop a model which can detect driver distraction using deep learning techniques. The model will be able to detect the driver's hands, eyes and face movements or any other activities the driver is doing while driving i.e. Talking on the phone, texting, grooming, eating, or feeling sleepy. [2]

II. Dataset Description

Here we are going to use the Statefarm dataset which has images taken from the 2D camera mounted on the car's dashboard. The dataset has images of the driver doing something in the car i.e. texting, eating, talking on the phone, makeup, reaching behind, etc. The dataset has 22,400 training samples with equal distribution among the classes and 79,700 unlabeled test samples. The train and text split are on the drivers such that one driver can only appear on either the train or the test set. Here the metadata such as creation dates has been removed as this is a computer vision problem.

There are 10 different classes of images which are stated below:

- 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

The dataset folder has different files in it
.

1. imgs.zip - zipped folder of all (train/test) images
2. sample_submission.csv - a sample submission file in the correct format
3. driver_imgs_list.csv - a list of training images, their subject (driver) id, and class id [3]

III. Implementation Details

Program Design Issues

1. The dataset is huge with 22,400 train samples and 79,700 test samples and thus training and testing on this dataset was quite resource intensive and led to crashing of the program all together several times.
   - ➢ The training time was highly dependent on the input image size thus we reduced the input image size to 150 instead of 224 without scarifying the performance of the model.
2. Also choosing the best model for the task at hand was quite difficult.
   - ➢ We tuned the hyperparameters for every single model we used by running on multiple machines available to us.

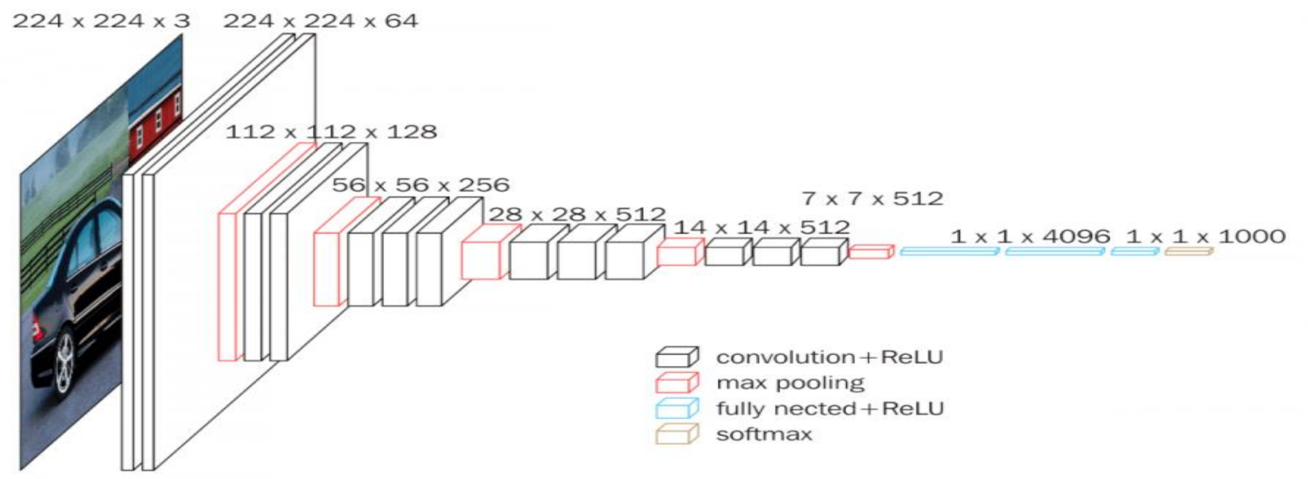
IV. PROPOSED SOLUTION

In this project we have used 3 pre trained models - VGG16, Mobile Net, Resnet50 and CNN.

VGG 16:

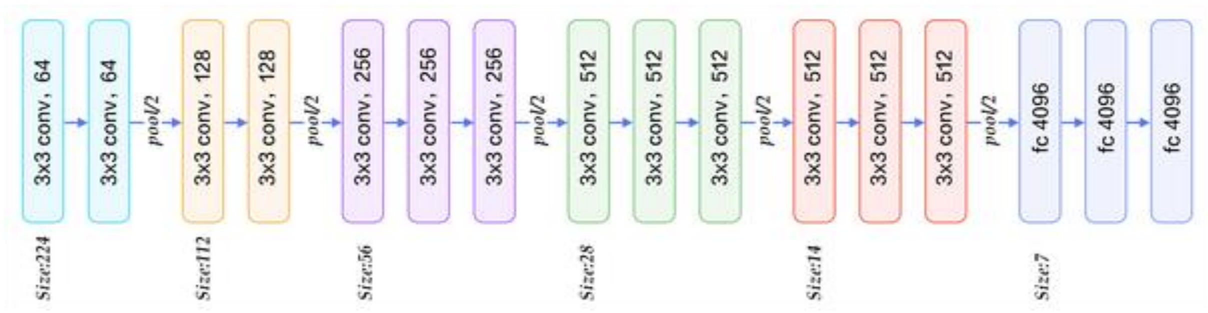
VGG-16 is a convolution neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford which has 16-layer The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous models submitted to the ILSVRC-2014 competition as it excels in Image based classification. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU.

The VGG-16 model architecture is as follows:



LAYERS IN THE VGG 16 MODEL:

1. CONVOLUTION USING 64 FILTERS
2. CONVOLUTION USING 64 FILTERS + MAX POOLING
3. CONVOLUTION USING 128 FILTERS
4. CONVOLUTION USING 128 FILTERS + MAX POOLING
5. CONVOLUTION USING 256 FILTERS
6. CONVOLUTION USING 256 FILTERS
7. CONVOLUTION USING 256 FILTERS + MAX POOLING
8. CONVOLUTION USING 512 FILTERS
9. CONVOLUTION USING 512 FILTERS
10. CONVOLUTION USING 512 FILTERS + MAX POOLING
11. CONVOLUTION USING 512 FILTERS
12. CONVOLUTION USING 512 FILTERS
13. CONVOLUTION USING 512 FILTERS + MAX POOLING
14. FULLY CONNECTED WITH 4096 NODES
15. FULLY CONNECTED WITH 4096 NODES
16. OUTPUT LAYER WITH SOFTMAX ACTIVATION WITH 1000 NODES [4]

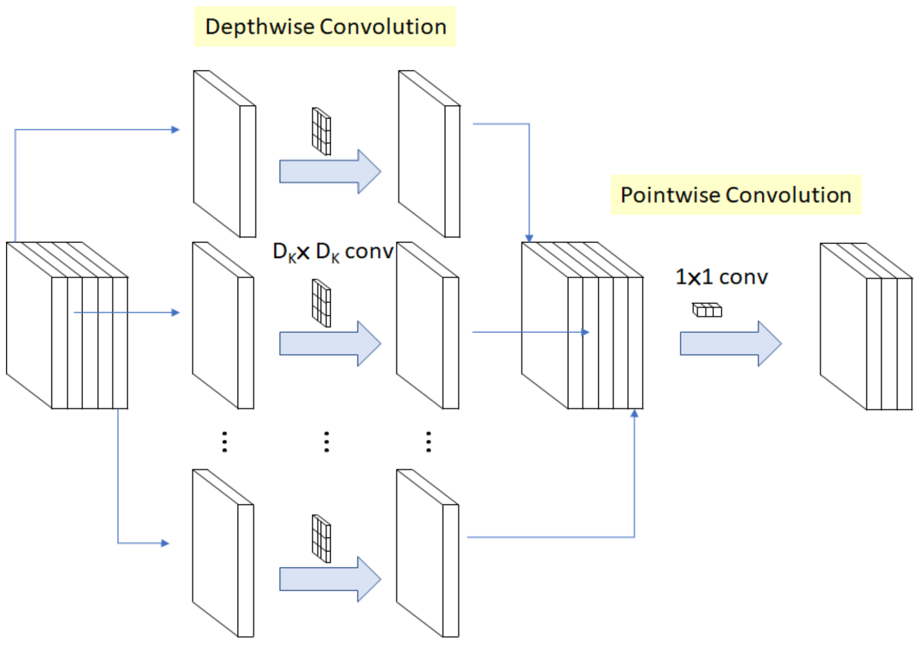


MOBILE NETS:

Mobilenets is a CNN architecture model for image classification and mobile vision developed by google. The main reason why mobile net is very special is that it requires very less computation power to run or apply transfer learning. This makes it a perfect fit for Mobile devices, embedded systems and computers without GPU or low computational efficiency with compromising significantly with the accuracy of the results. Mobilnet is also best suited for web browsers as browsers have limitations over computation, graphic processing and storage.

MobileNets for mobile and embedded vision is based on a streamlined architecture that uses depth wise separable convolutions to build light weight deep neural networks. Here the width and resolution can be tuned to tradeoff between latency and accuracy. The depthwise separable convolution is used to reduce the model size and complexity. It has smaller model size (i.e. Less number of parameters) and Lesser Complexity (i.e. Fewer Additions and multiplications). There are 2 parameters in the MobileNets Width Multiplier α and Resolution Multiplier ρ to easily tune the model.

Depthwise separable convolution is a depthwise convolution followed by a pointwise convolution. In MobileNet, the depthwise convolution applies a single filter to each input channel and later on the pointwise convolution applies a 1×11×1 convolution to combine the outputs from the depthwise convolution.



The cost of Depthwise separable convolution:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

Depthwise Separable Convolution Cost: Depthwise Convolution Cost (Left), Pointwise Convolution Cost (Right)

Where M is the Number of input channels, N is the Number of output channels, DK is Kernel size, and DF is Feature map size.

The cost of standard convolution is:

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$$

Standard Convolution Cost

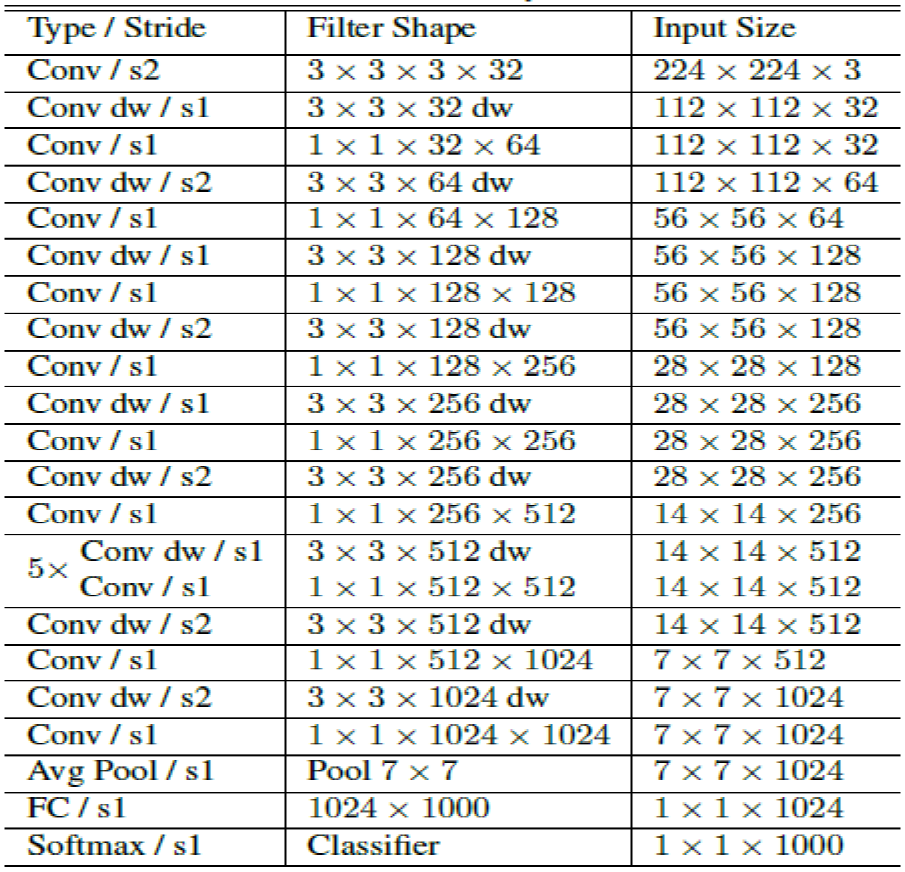
Thus, the computation reduction is:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F}$$
$$= \frac{1}{N} + \frac{1}{D_K^2}$$

Depthwise Separable Convolution Cost / Standard Convolution Cost [5]

We can achieve 8 to 9 times less computation when DK×DK is 3×3, but with only a small reduction in accuracy.

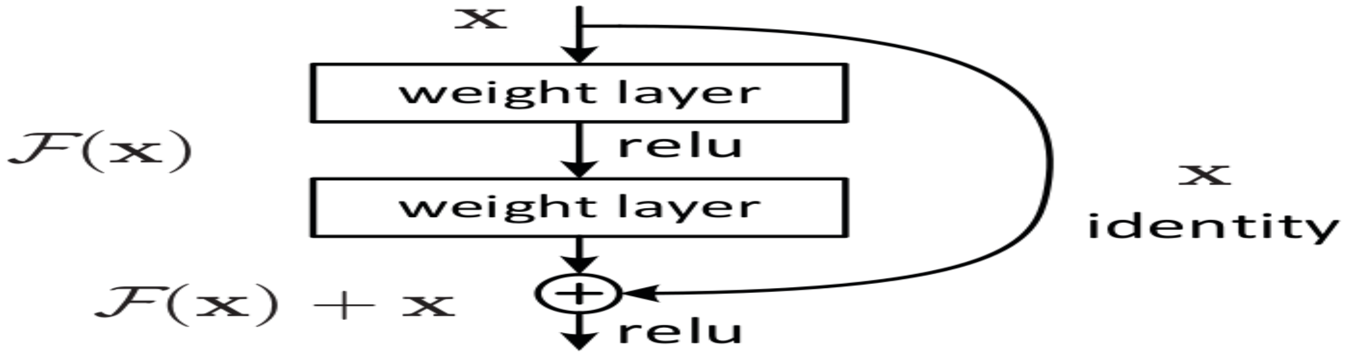
Table 1. MobileNet Body Architecture



Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
5× Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

<u>RESNET 50:</u>

ResNet is the short form of Residual Network. In this network architecture there are 50 layers and it is based on residual learning. Generally, in a deep convolutional network layers are stacked and trained for a specific task on a specific dataset and the network learns several features. But in Residual learning the case is different, here in residual learning instead of learning new features the model learns residual. Residual is the subtraction of features learned from input of that layer. ResNet does this by directly connecting input of the Nth layer to some (N + X)th layer; this is called shortcut connections. Training this type of network is easier compared to the normal deep convolutional network. It also solves the problem of degrading accuracy. In a simple deep neural network there is a problem of vanishing gradients as the model back propagates the gradients gets smaller and smaller and this can make learning intractable and these problems are overcome by ResNet 50. [6]



Shortcut Connection

Convolution Neural Network:

Here we are using 6 convolution layers and 2 dense layers.

1st layer and 2nd layers – 32 filters (3 x 3)
3rd, 4th and 5th layers – 64 filters (3 x 3)
6th layer – 128 filters (3 x 3)
7th layer – 128 nodes
Output layer – 10 nodes.

V. Results Obtained

The Table below shows some of the different models that we tried and tuned on our dataset.

MODELS USED	IMAGE SIZE	OPTIMIZER	EPOCHS	EPOCH FOR BEST RESULT	TRAIN ACC	TRAIN LOSS	VAL ACC	VALIDATION LOSS
CNN	**150**	**ADAM**	**20**	**18**	**0.9032**	**0.2935**	**0.6382**	**1.6969**
VGG16 -NO EXTRA LAYERS	**200**	**SGD**	**15**	**11**	**0.9636**	**0.1219**	**0.8774**	**0.4837**
VGG16 -NO EXTRA LAYERS	**150**	**SGD**	**15**	**11**	**0.9624**	**0.1259**	**0.8765**	**0.4391**
VGG16 – WITH EXTRA LAYERS	**150**	**SGD**	**15**	**10**	**0.9926**	**0.0280**	**0.8414**	**0.5252**
MOBILENETS NO EXTRA LAYERS	**200**	**SGD**	**15**	**12**	**0.9766**	**0.0847**	**0.7637**	**0.8469**
MOBILENETS NO EXTRA LAYERS	**150**	**SGD**	**15**	**8**	**0.9576**	**0.1413**	**0.7527**	**0.7092**
RESNET50 NO EXTRA LAYERS	**150**	**SGD**	**15**	**10**	**0.9852**	**0.0506**	**0.7342**	**0.7176**

- For this project we tried a lot of different combinations of models like VGG16 – with /without extra layers, MobileNets with / without extra layers, Resnet50 with/without extra layers, and CNN. We also changed the image size and shape from the default (224 x 224 x 3) to (200 x 200 x 3) and (150 x 150 x 3). We noted all the observations by running the dataset on all the combinations and we found out that there is not much difference between the performance of VGG16 with no extra layers and with the image size 200 and 150. The same can be observed in the table above.

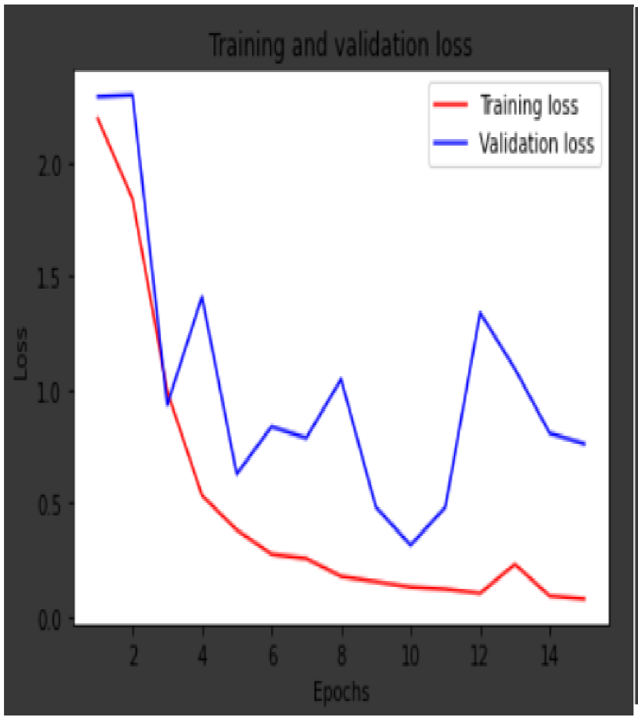
- We also tried using MobileNets but its accuracy was compromised as it is suitable for less complex models while with ResNet the model is highly overfitting.

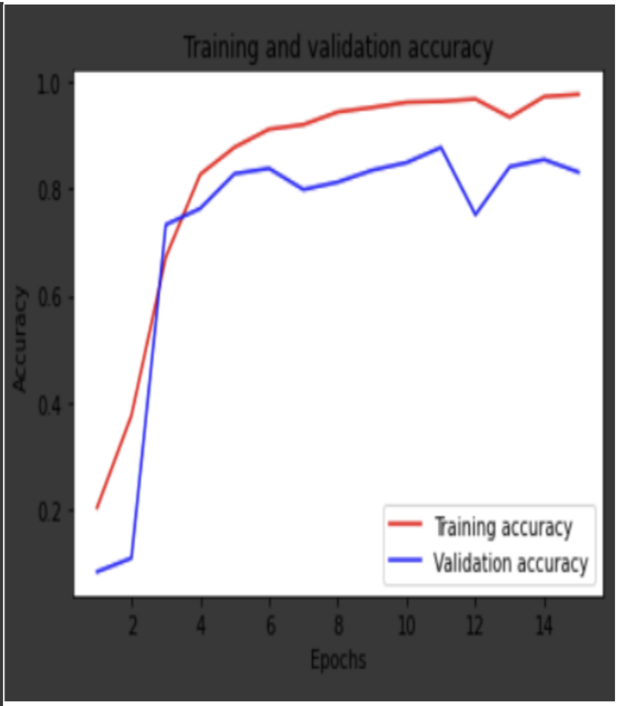
- So, for our model VGG 16 – without extra layers and with input image size 150 performs the best, as it takes less run time compared to the VGG 16 – without extra layers and with input image size 200. The one with input image size 150 takes 90 seconds for one epoch while the one with input image size 200 takes 180 seconds.

- VGG16 with extra layers overfits the model and the normal CNN does not provide accurate performance. So, for out model VGG 16 – with no layers with input image size 150 performs the best.

VGG16 – NO EXTRA LAYERS (200, 200)

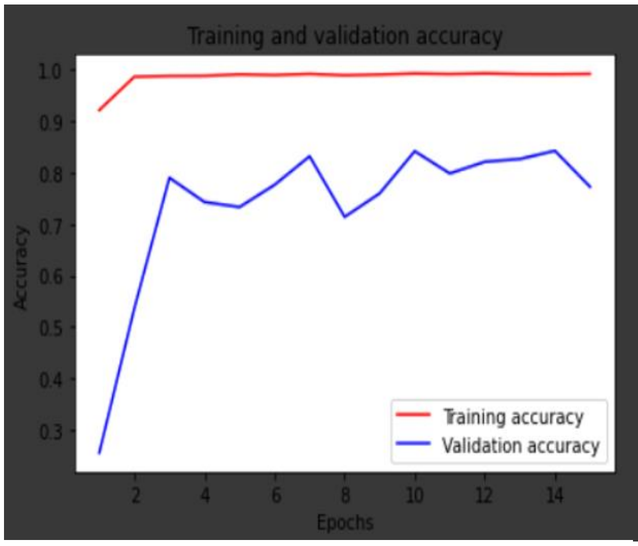
The graphs below show the training and validation losses and accuracies respectively as a function of the number of epochs:

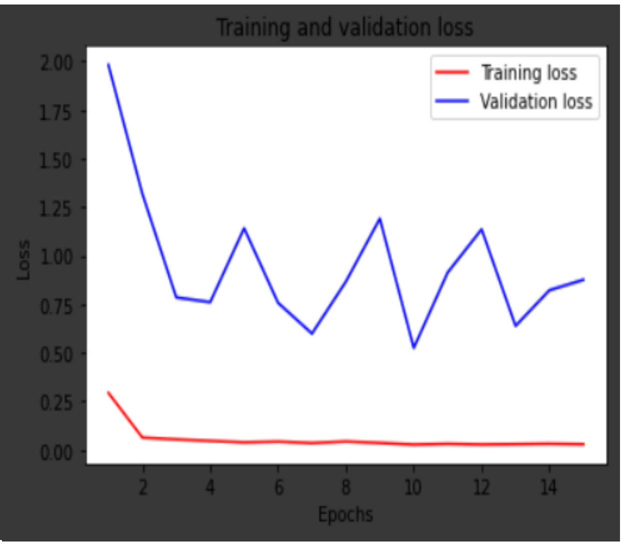




VGG16 – EXTRA LAYERS (200, 200)

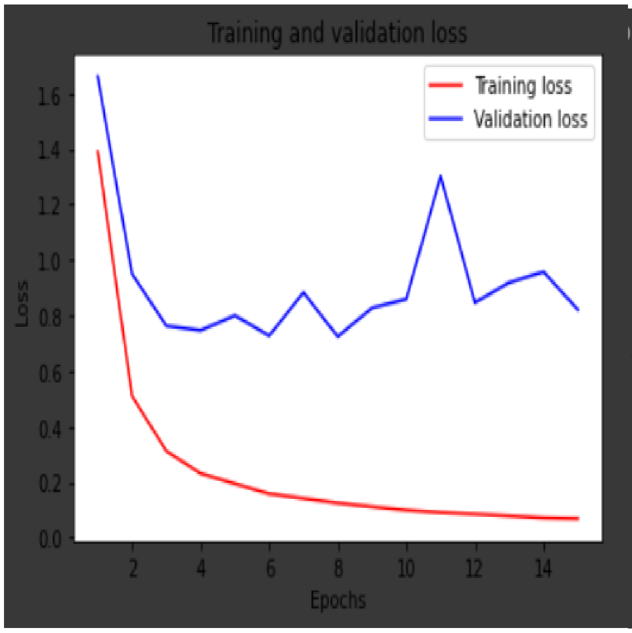
The graphs below show the training and validation losses and accuracies respectively as a function of the number of epochs:

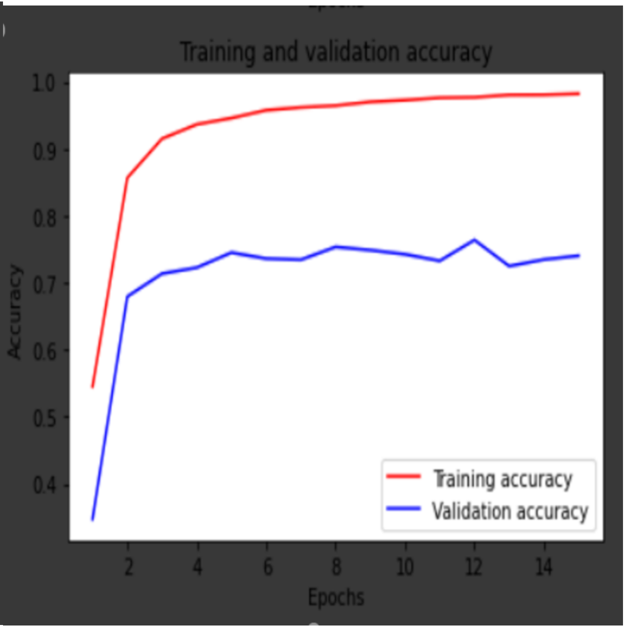




MobileNet – No Extra Layers (200, 200)

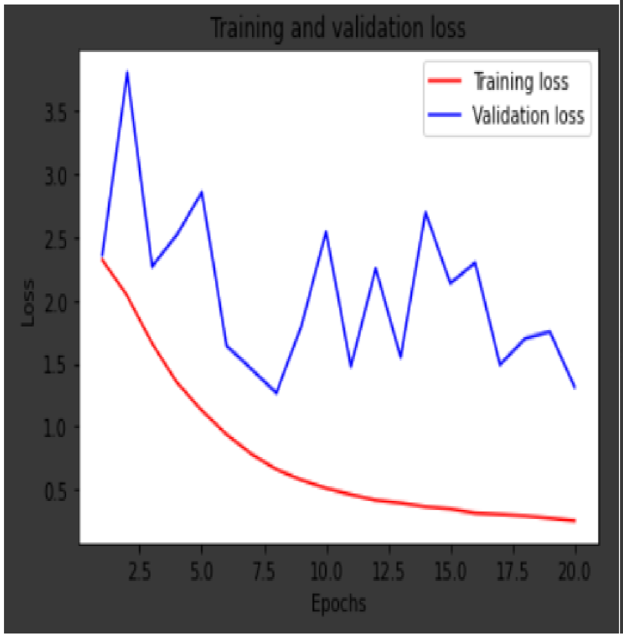
The graphs below show the training and validation losses and accuracies respectively as a function of the number of epochs:

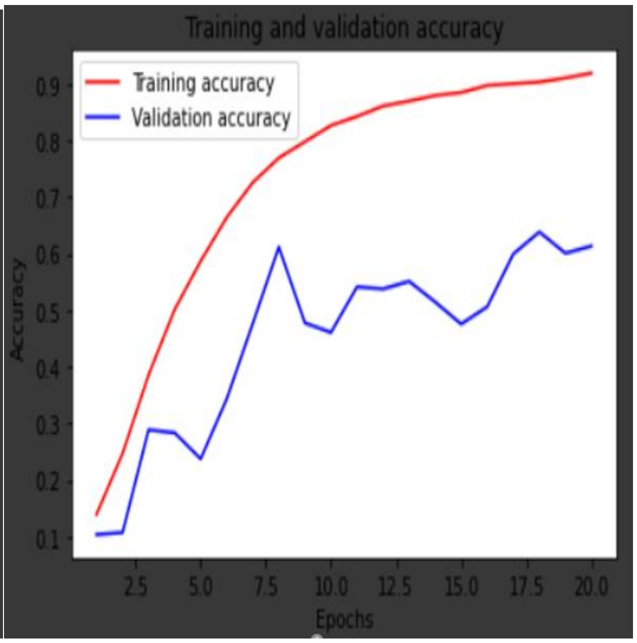




CNN (150, 150)

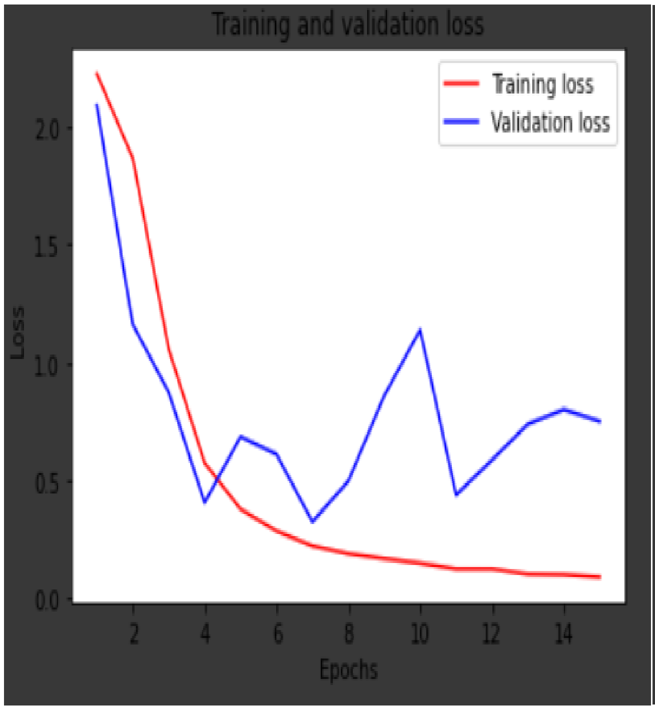
The graphs below show the training and validation losses and accuracies respectively as a function of the number of epochs:

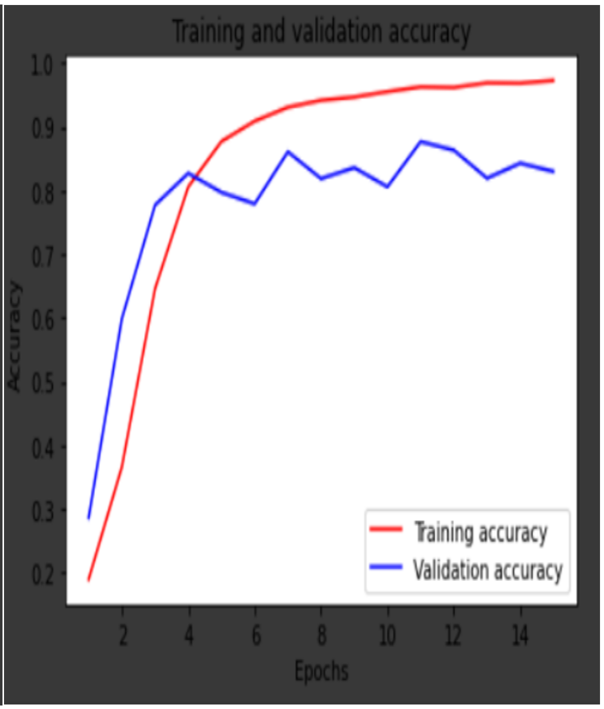




VGG16 (150, 150) – NO Extra Layers

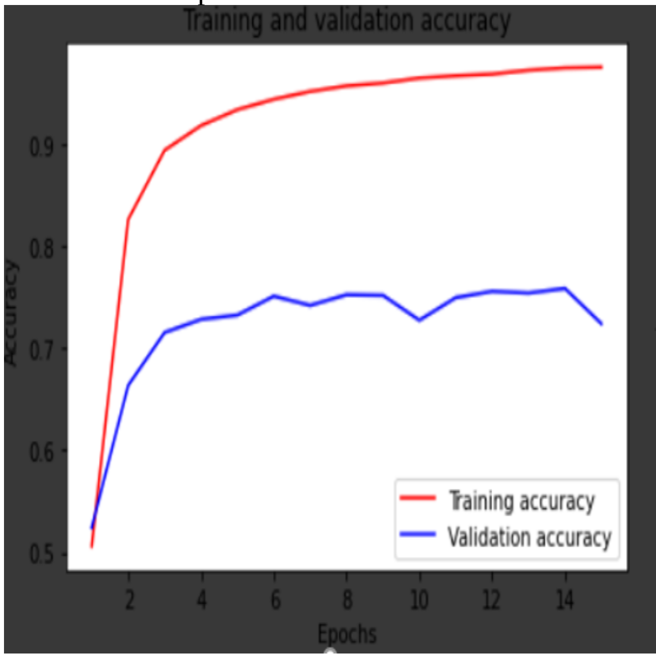
The graphs below show the training and validation losses and accuracies respectively as a function of the number of epochs:

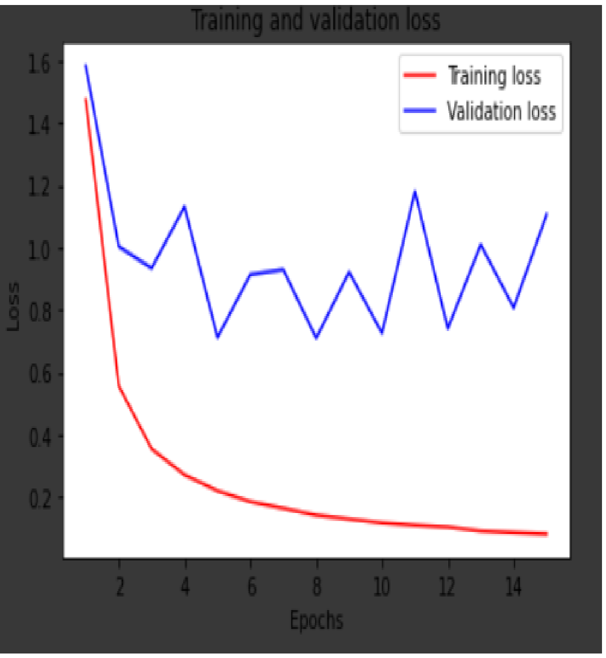




MOBILENET (150, 150)

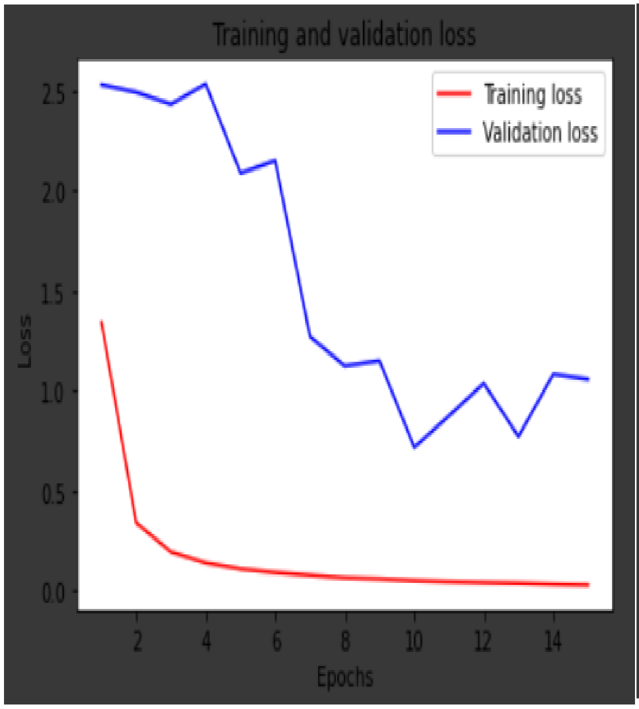
The graphs below show the training and validation losses and accuracies respectively as a function of the number of epochs:

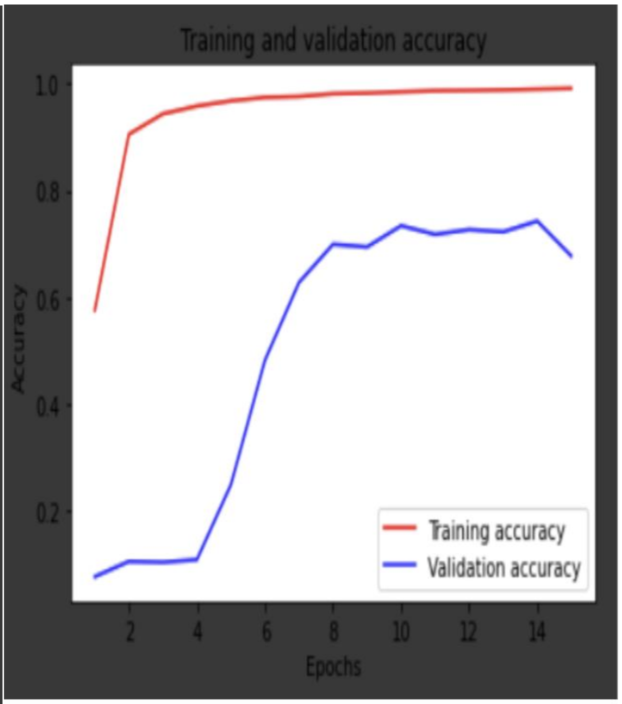




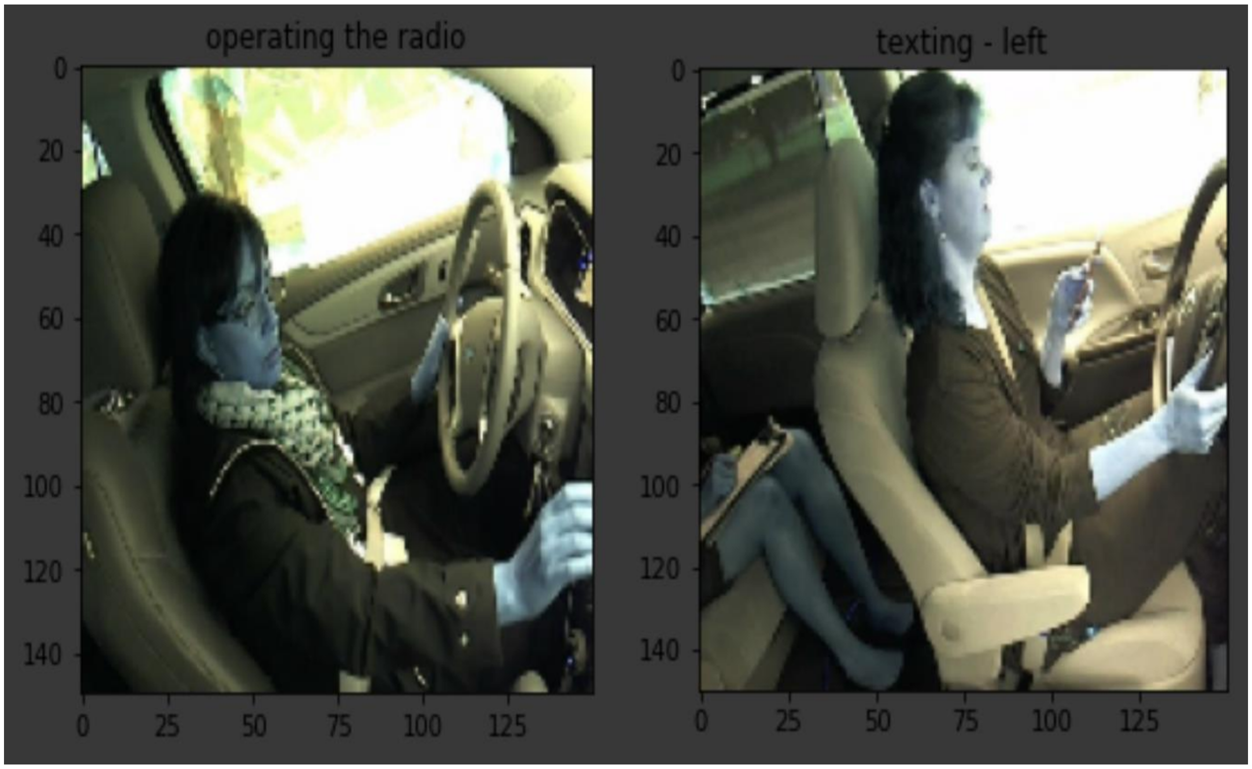
ResNet (150, 150) – No Extra Layers

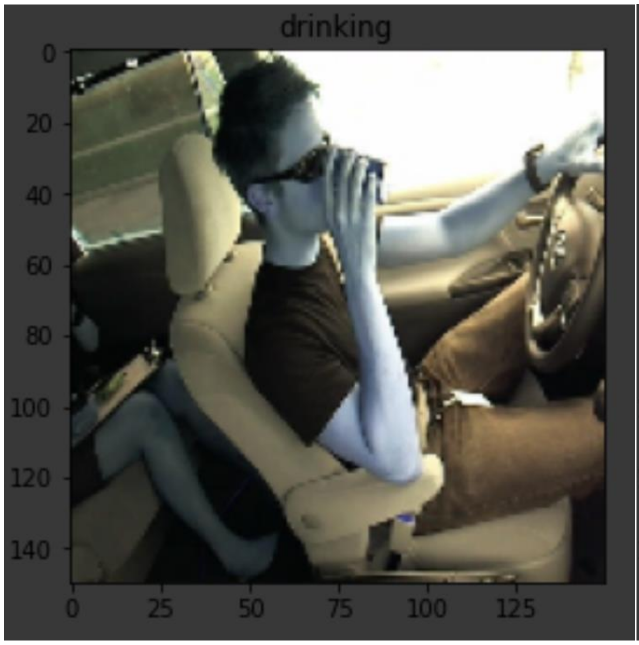
The graphs below show the training and validation losses and accuracies respectively as a function of the number of epochs:

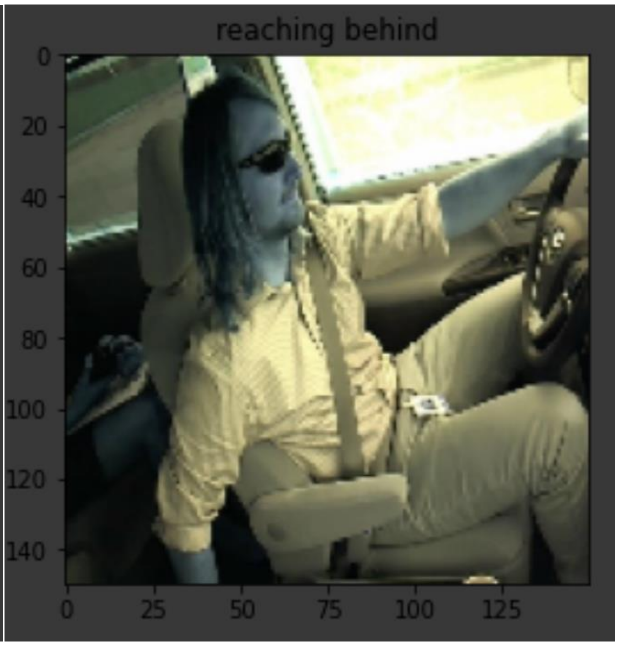


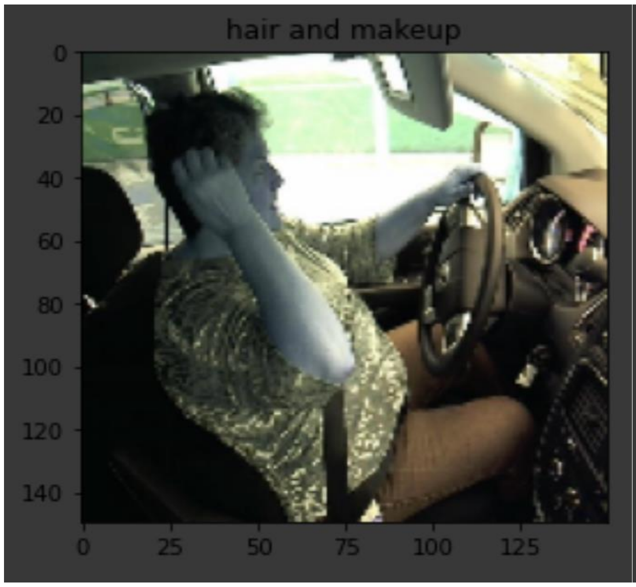


After using the VGG16 with no extra layers (input image size 150) on the test set below are some of the test results obtained belonging to different categories:

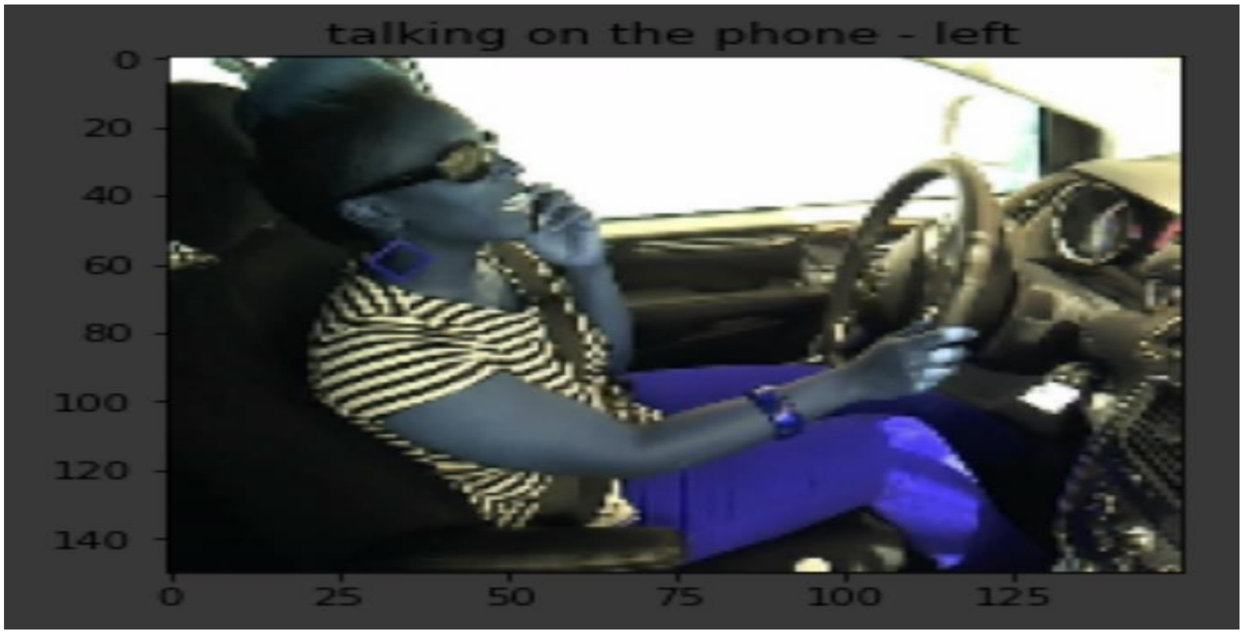



drinking
0
20
40
60
80
100
120
140
0
25
50
75
100
125


reaching behind
0
20
40
60
80
100
120
140
0
25
50
75
100
125


hair and makeup
0
20
40
60
80
100
120
140
0
25
50
75
100
125

talking on the phone - right
0
20
40
60
80
100
120
140
0
25
50
75
100
125


talking on the phone - left
0
20
40
60
80
100
120
140
0
25
50
75
100
125

VI. CONCLUSION

We successfully predicted all the 79700 test images by using VGG16 with no extra layers (input image size 150).

References

[1] "Distracted Driving," [Online]. Available: https://www.nhtsa.gov/risky-driving/distracted-driving.

[2] [Online]. Available: https://teensafe.com/100-distracted-driving-facts-statistics-for-2018/.

[3] "Kaggle - state-farm-distracted-driver-detection," [Online]. Available: https://www.kaggle.com/c/state-farm-distracted-driver-detection/data.

[4] "VGG16," [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/.

[5] MobileNets. [Online]. Available: https://medium.com/analytics-vidhya/image-classification-using-mobilenet-in-the-browser-b69f2f57abf.

[6] "Resnet50," [Online]. Available: https://www.quora.com/What-is-the-deep-neural-network-known-as-%E2%80%9CResNet-50%E2%80%9D.

[7] D. U. A. S. Kusuma.S, "Driver Distraction Detection using Deep Learning and Computer Vision," 2019.

[8] C. Streiffer, "DarNet: A Deep Learning Solution for Distracted Driving Detection," *DarNet: A Deep Learning Solution for Distracted Driving Detection ACM. 978-1-4503-5200-0/17/12DOI: 10.1145/3154448.3154452*, 2017.