Driver's Distraction Detection using Deep Learning and Computer Vision

Quick Savajiyani (A20451378)

Vatsal Ketan Patel (A20458061)

Introduction

- Automobile crashes have tremendously increased these days and first the foremost cause for motor vehicle accidents and incidents is Driver's distraction.
- 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.

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 in the dataset 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

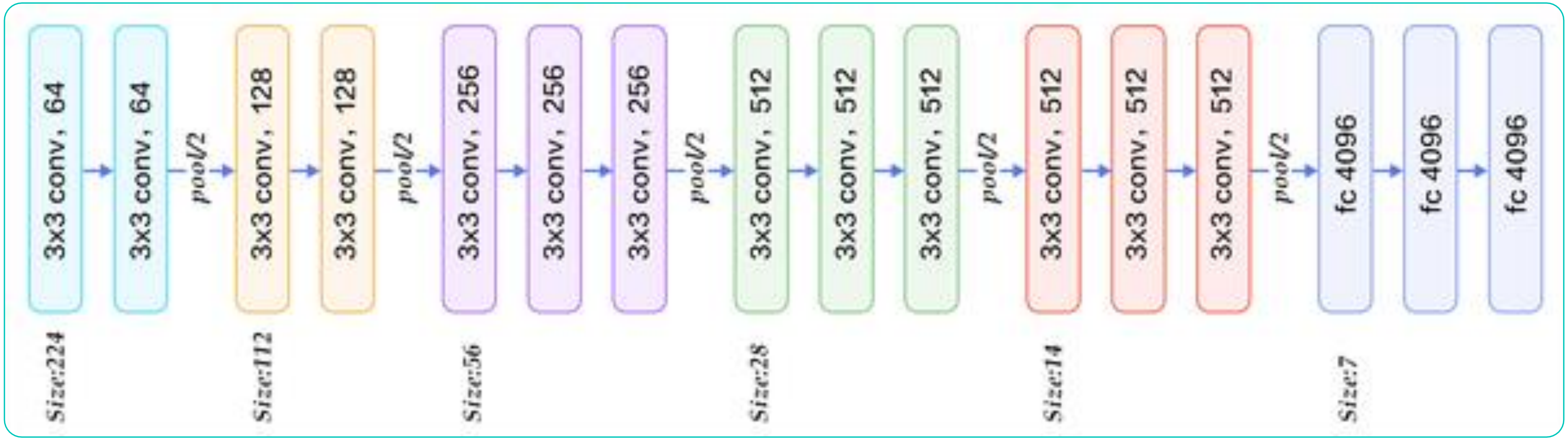
Implementation Details

- 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.
- 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.

PROPOSED SOLUTION

In this project we have trained our dataset on 3 well known pretrained models.

1. VGG16
2. MobileNets
3. Resnet50
4. Convolution Neural Network



VGG 16

- VGG-16 is a convolution neural network model which has 16-layer.
- The above image shows its layers

MobileNets Architecture

- 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.
- It is based on a streamlined architecture that uses depth wise separable convolutions to build light weight deep neural networks.
- It has smaller model size (i.e. Less number of parameters) and Lesser Complexity (i.e. Fewer Additions and multiplications).

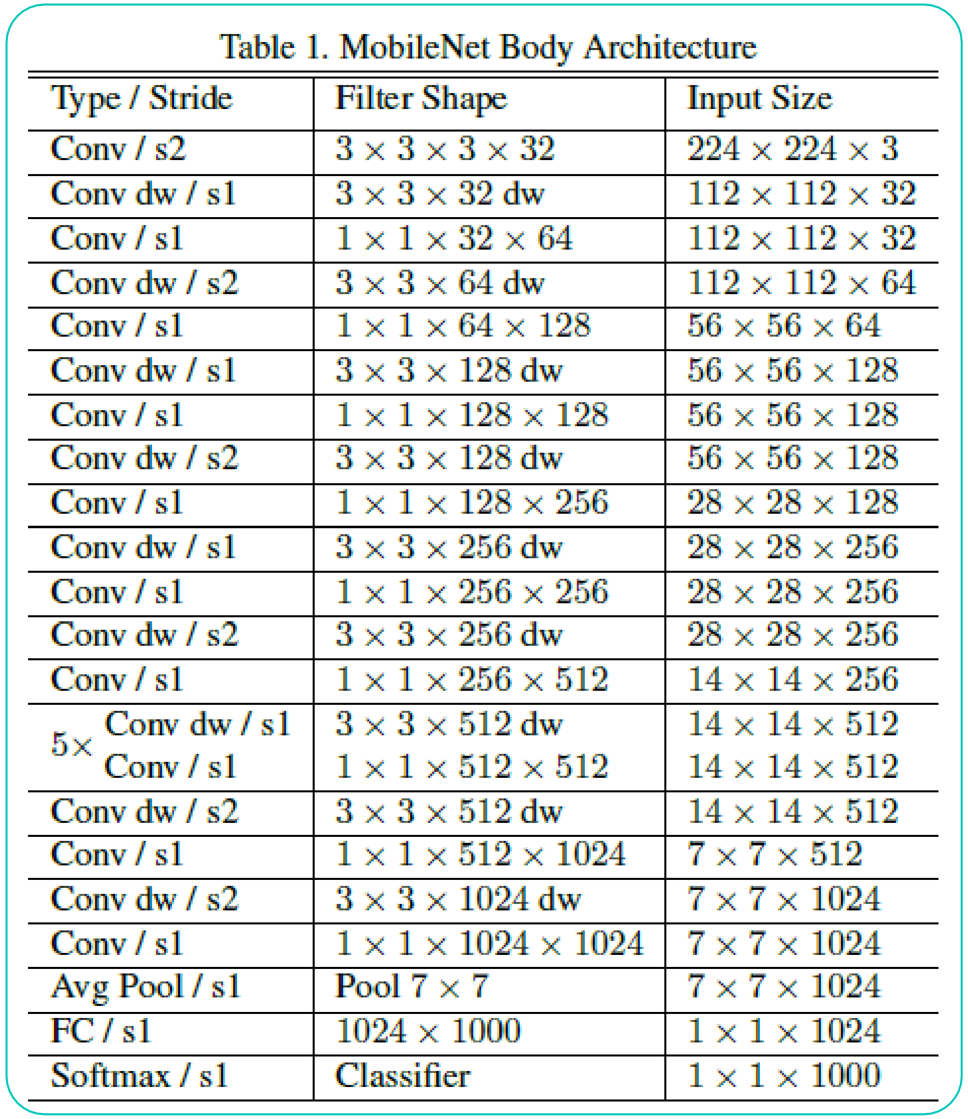
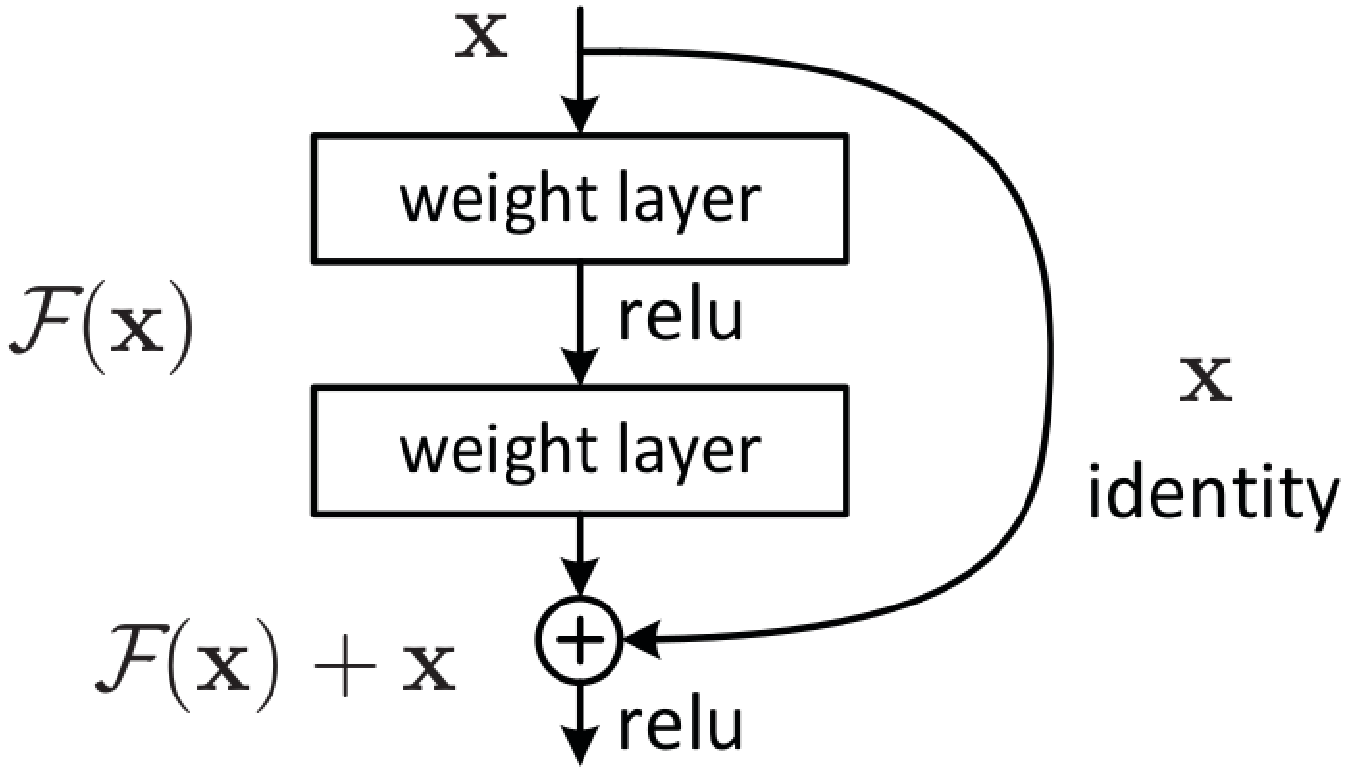


Table 1. MobileNet Body Architecture

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	$1 \times 1 \times 1000$

ResNet 50:

- 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.
- 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.
- 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.



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.

Results Obtained

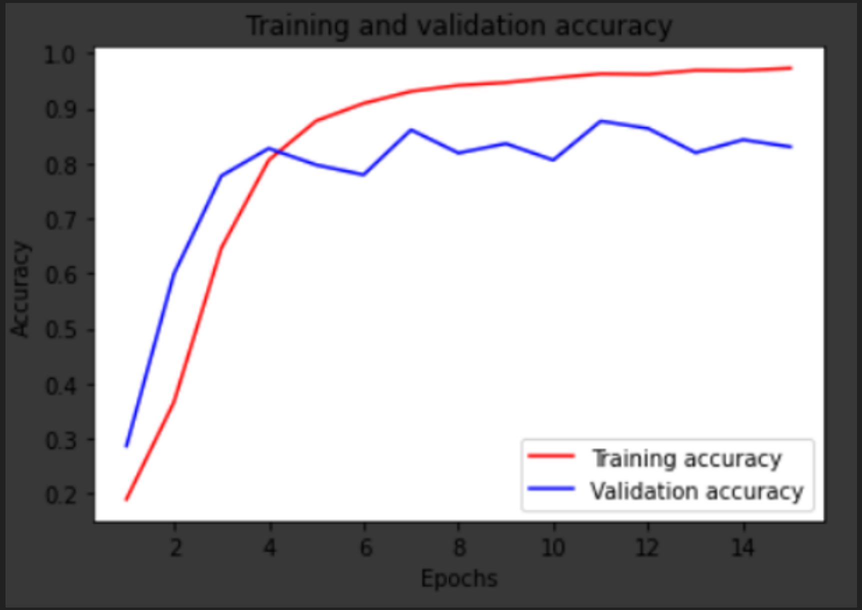
- For this project we tried a lot of different combinations of models like VGG16 – with /without extra layers, MobileNets with / without extra layers, Resnet50 with/without extra layers, and CNN. We also changed the image size and shape from the default (224 x 224 x 3) to (200 x 200 x 3) and (150 x 150 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.

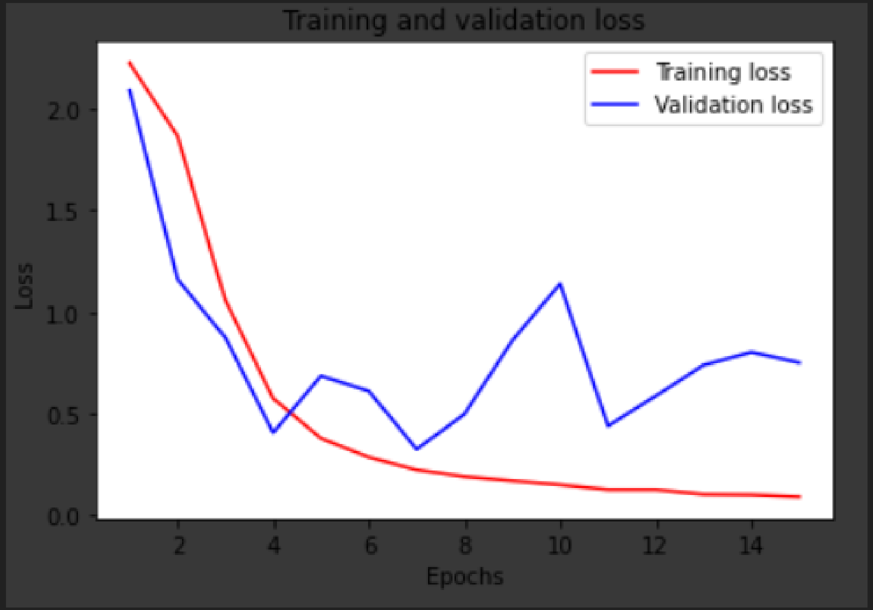
- We also tried using MobileNets but its accuracy was compromised as it is suitable for less complex models while with ResNet the model is highly overfitting.

- 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.

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





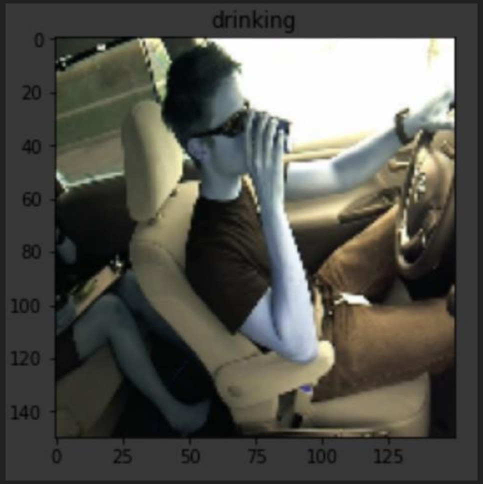
VGG16 (150, 150) – NO Extra Layers

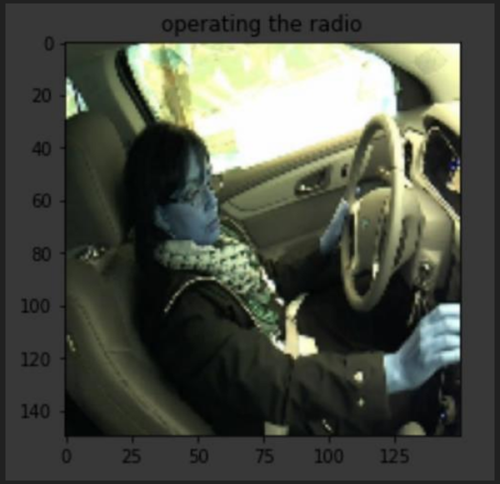
- The graphs above shows the training and validation losses and accuracies respectively as a function of the number of epochs

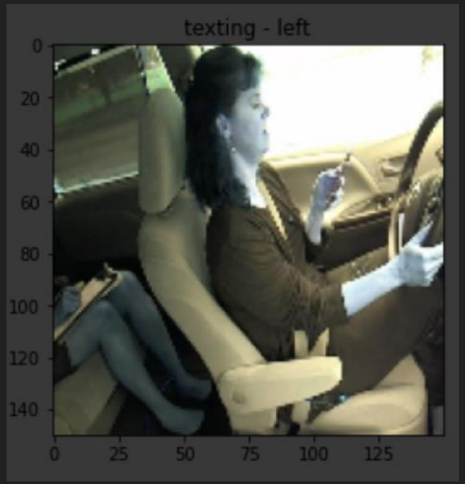
Test Results Belonging to different categories

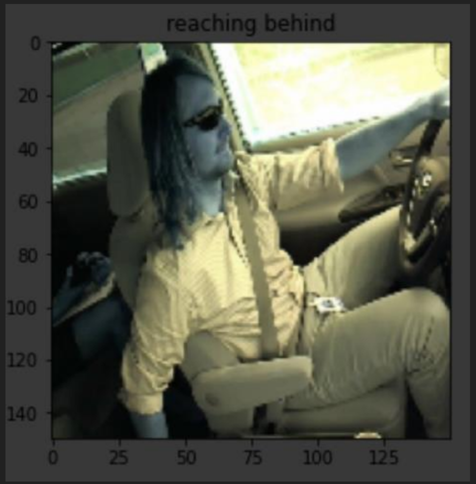
- After using the VGG16 with no extra layers (input image size 150) on the test set the coming slides shows the test results obtained belonging to different categories:

Test Results Belonging to different categories

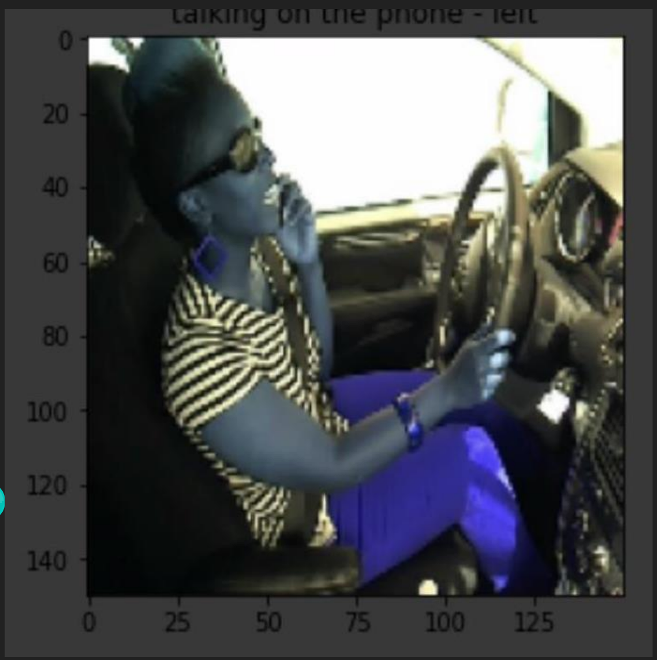


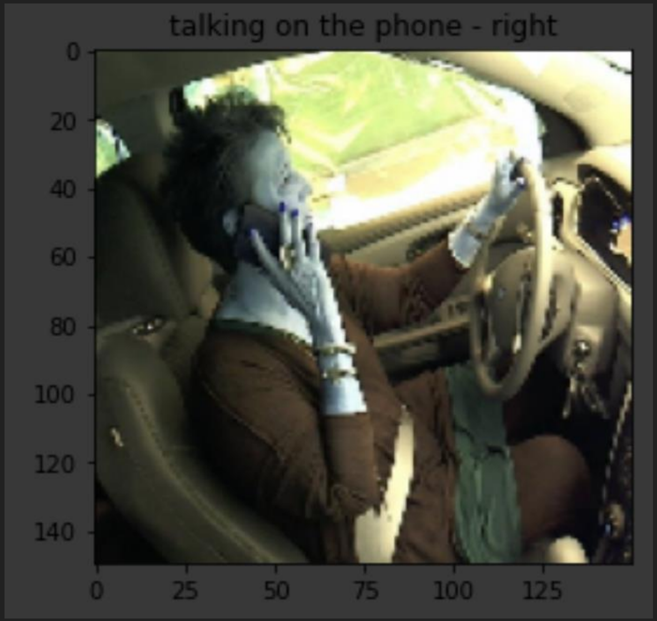


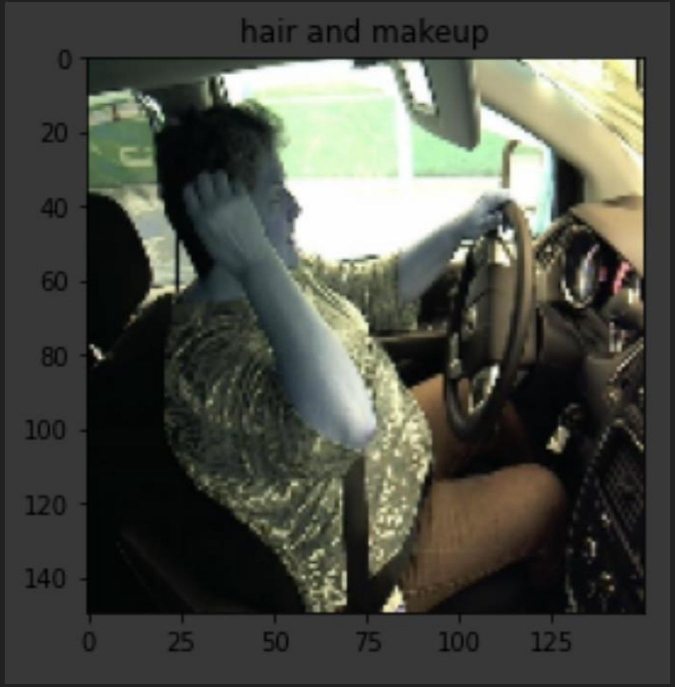




Test Results Belonging to different categories







Conclusion

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

https://drive.google.com/open?id=1jx78VZRGd8o6q
CrawYLJ5Yt5w_lne_cG

References

- https://neurohive.io/en/popular-networks/vgg16/
- https://medium.com/analytics-vidhya/image-classification-using-mobilenet-in-the-browser-b69f2f57abf
- https://www.quora.com/What-is-the-deep-neural-network-known-as-%E2%80%9CResNet-50%E2%80%9D
- https://www.kaggle.com/c/state-farm-distracted-driver-detection/data
- Driver Distraction Detection using Deep Learning and Computer Vision, Kusuma.S, Divya Udayan.J, Aashay Sachdeva, 2019,

https://ieeexplore-ieee-org.ezproxy.gl.iit.edu/document/8993260/references#references W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

Thank you