DISTRACTED DRIVER DETECTION

1. Problem Statement:

Road accidents are caused a lot in the past and in the present due to various distractions that a driver can be exposed to. The distractions could be due to the passengers, music, gadgets, drinks, etc. The distraction of the driver results in accidents which not only affects him/her but also the passengers and the other people on the road. Thus, there exists a need to monitor the driver and alert him/her when he/she has lost concentration in driving. Sensors can be used for such monitoring purposes. The widely used sensor is the dash cam. The system can have a machine learning model running which can help detect the distracted drivers through the images captured by the dash cam. This system should have less number of false negatives possible to be entitled as a safe driver monitoriing system.

I have desined a pipeline to classify the input dash cam images of drivers into 10 different classes.

2. Dataset:

The model uses StateFarm Disttracted Driver Detection dataset. It contains 22424 images along with their class labels. The dimension of each image is 480x640. They are RGB images. Figure I shows an image from the training dataset.

Figure I

3. Model :

3.1 Architecture :

Layer	**Parameter**	**Output Shape**
Regression Network		
conv2d	32 filters	1,32,150,150
conv2d	64 filters	1,64,150,150
conv2d	64 filters	1,64,150,150
flatten	64 units	1,64*150*150
dense	64 units	1,64
dense	64 units	1,64
dense	64 units	1,32
dense	32 units	1,8

Extract 4 crops using the 8 co-ordinates that the above trained regression network outputs.
20x20 pixels around each co-ordinate are cropped from the respective images. The crops of respective images are stacked.
Using these images, the classification network shown below is trained.

Classification Network

ZeroPadding	1,1	20,122,122,3
conv2d	64 filters	20,120,120,64
Activation	—	20,120,120,64
conv2d	64 filters	20,118,118,64
Activation	—	20,118,118,64
MaxPool	2,2	20,59,59,64
ZeroPadding	1,1	20,61,61,64
conv2d	64 filters	20,59,59,64
Activation	—	20,59,59,64
ZeroPadding	1,1	20,61,61,64
conv2d	64 filters	20,59,59,64
Activation	—	20,59,59,64
MaxPool	2,2	20,29,29,64
ZeroPadding	1,1	20,31,31,64
conv2d	64 filters	20,29,29,64
Activation	—	20,29,29,64
ZeroPadding	1,1	20,31,31,64
conv2d	64 filters	20,29,29,64
Activation	—	20,29,29,64
ZeroPadding	1,1	20,31,31,64
conv2d	64 filters	20,29,29,64
Activation	—	20,29,29,64
MaxPool	2,2	20,14,14,64
ZeroPadding	1,1	20,16,16,64
conv2d	64 filters	20,14,14,64
Activation	—	20,14,14,64
ZeroPadding	1,1	20,16,16,64
conv2d	64 filters	20,14,14,64
Activation	—	20,14,14,64
ZeroPadding	1,1	20,16,16,64
conv2d	64 filters	20,14,14,64
Activation	—	20,14,14,64
MaxPool	2,2	20,7,7,64
Dropout	0.2	20,7,7,64
ZeroPadding	1,1	20,9,9,64
conv2d	64 filters	20,7,7,64
Activation	—	20,7,7,64
ZeroPadding	1,1	20,9,9,64
conv2d	64 filters	20,7,7,64
Activation	—	20,7,7,64
ZeroPadding	1,1	20,9,9,64

conv2d	64 filters	20,7,7,64
Activation	—	20,7,7,64
MaxPool	2,2	20,3,3,64
Dropout	0.3	20,3,3,64
flatten	—	20, 576
dense	4096 units	20,4096
Activation	—	20,4096
dropout	0.5	20,4096
dense	4096 units	20,4096
Activation	—	20,4096
dropout	0.5	20,4096
dense	10 units	20,10

The last layer outputs the probabilities for 10 classes.

By using keras.predict_classes() the output class is generated.

3.2 Hyper parameters :

3.2.1 Hyper parameters chosen for tuning :

Epochs = 40, 80, 120

Batch size = 32, 60, 80

3.2.2 Tuned hyper parameter values :

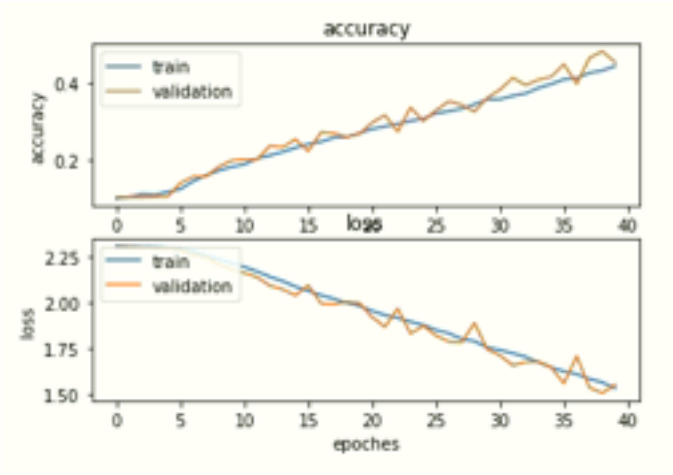
Epochs = 120

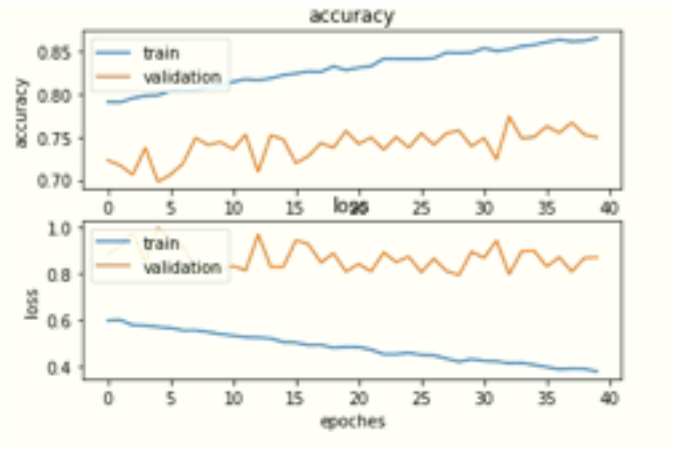
Batch size = 80

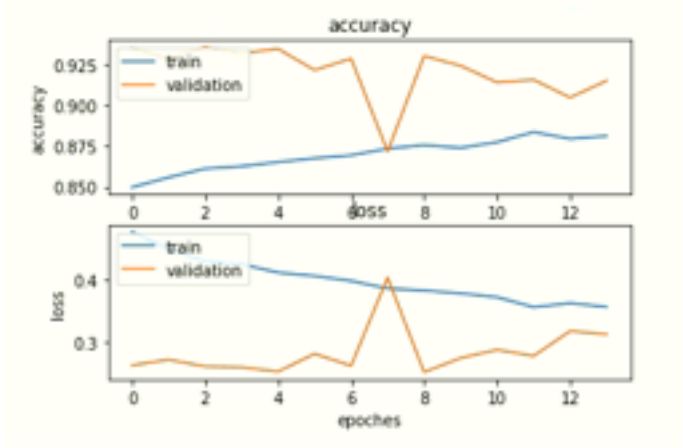
3.3 Training Curves :

3.3.1 Regression network + Classification network

Validation Accuracy : 93.5%

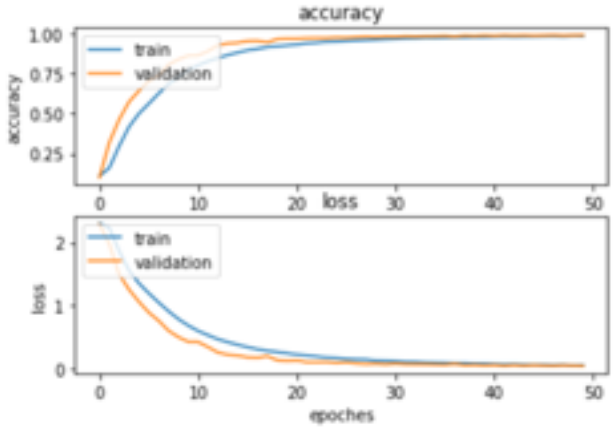






3.3.2 Standalone Classification network

Validation Accuracy : 99.02%



3.4 Implications in the regression network + classification network architecture :

The stacked images were only of 120x30 pixel dimension. They were resized to 120x120 pixel dimension to feed them into the classification network model that I designed. Thus, the input to the classification network model had a portion of black pixels which did not contain any useful data. The network had to learn to omit the trivial black pixels and to learn only those pixels that were extracted from the original image. This learning process made the gradient descent to reach local minima in 114 epochs which is slower when compared to the learning process of feeding the entire image as input to the classification network.

Though the learning process was slower, the classification network that learnt from the feature extracted data is robust than the conventional classification networks.

The dropout layers are added between pooling layers and dense layers to increase the robustness of the network by makning the learning process more generic. Various ratios are used for the dropout layers.

4. Project Files :

4.1 Github Link :

https://github.com/Shruthi-Sampathkumar/Distracted-Driver-Detection

4.2 Regression Model Link :
https://drive.google.com/file/d/1YK_be5plQZee7WDt9BaR-mU7ZSgWSbAJ/view?usp=sharing

4.3 Classification model Link :
https://drive.google.com/file/d/15EExJn8i4V49JRWeEhQ2tFfwyYNd4hJb/view?usp=sharing

4.4 Standalone Classification model Link :
https://drive.google.com/file/d/1ZWVdUvOtHvhRGBN-uuRPgKgMt9zlqQX5/view?usp=sharing

4.5 GUI Tutorial Link :
https://youtu.be/KeFj3yKZl0k

4.6 Model to be used in GUI:
https://drive.google.com/file/d/15EExJn8i4V49JRWeEhQ2tFfwyYNd4hJb/view?usp=sharing