# Distracted Driver Detection

## Business understanding:

- From a report from the CDC motor vehicle safety division, one in five car accidents is caused by a distracted driver. This has just become like drunken driving. Almost 425,000 people injured and 3,000 people killed by distracted driving every year.
- We are aiming to improve the alarming statistics caused by the distracted driving, by testing whether dashboard cameras can automatically detect drivers engaging in distracted behaviors. State Farm insurance company has given us a dataset of 2D dashboard camera images, and we need to develop an algorithm to detect and classify driver's behaviour and check if they are driving attentively or not. We are trying to implement deep learning arhitecture to create models and predict and classify them by training them on the given datset.

**The 10 classes to predict 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

## Data Understanding

We have 3 data files:

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

# Data Processing

**Splitting of data**

- Splitting data based on subject IDs.

- We have manually selected the following subject IDs to be our test data
  - p066
  - p035
  - p002
  - p041

**Image Augmentation**

- normalized the pixel value so that it ranges from 0 to 1
- Shifting the image vertically
- Flipping the image on x as well as y axis
- Rotating the image by 20 degree

**Appending class name with image name to make it suitable for image data generator**

# EDA

- Performing preliminary exploratory data analysis to see the quality of the data such as whether the data is balanced(approximately equal number of datapoints from each classes) or not.

```
count of datapoint in each Classname
```

Out[17]:
```
c0    2489
c3    2346
c4    2326
c6    2325
c2    2317
c5    2312
c1    2267
c9    2129
c7    2002
c8    1911
Name: classname, dtype: int64
```

From above result we can see that most of the classes have around 2000 images, hence our data is balanced.

```
count of each subject ID
```

Out[18]:
```
p021    1237
p022    1233
p024    1226
p026    1196
p016    1078
p066    1034
p049    1011
p051     920
p014     876
p015     875
p035     848
p047     835
p081     823
p012     823
p064     820
p075     814
p061     809
p056     794
p050     790
p052     740
p002     725
p045     724
p039     651
p041     605
p042     591
p072     346
Name: subject, dtype: int64
```

Different people have different number of images. We will consider p066, p035, p002, p041 as our test data and rest as the training data.

```
dimension of test dataset (3212, 3)
```

```
dimension of train dataset (19212, 3)
```

```
The first 5 rows of the test dataframe
```

Out[16]:

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

# Modeling

```
Found 14409 validated image filenames belonging to 10 classes.
Found 4803 validated image filenames belonging to 10 classes.
Found 3212 validated image filenames.
```

## Using CNN Model:

For this project we are using CNN for image classifcation. We are given a dataset of 2D dashboard camera images, so our first choice was CNN as it requires very little pre-processing, meaning that it can read 2D images by applying filters.

The hidden layers comprise convolutional layers, ReLU activation function, pooling layers, and fully connected layers, all of which play a crucial role.

Here we have an input layer, an output layer, and hidden layers, all of which help process and classify images. Architecture of CNN: Here we are using sequential model() as it helps to build a model layer by layer. add() function is used to add layers to our models. The Conv2D layers are convolution layers. It deals with our input images, which are seen as 2-dimensional matrices. We use padding = ‘same’ which ensures that the output has the same size as the input. This layer also has an input shape parameter. This is the shape of each input image, 256,256,3 with the 3 signifying that the images are RGB.

Activation Function adds non-linearity to the network. The activation function used here is ReLu activation function. It is efficient and easy for computation. With ReLu, all the negative values are removed, and the accuracy of the image is increased. Now we have the pooling layer. It helps in controlling the overfitting of the model. We are using max pooling. It is the process of merging. It helps to extracte the features that are highly important or that are dominant or high lighted in the image. It identifies the largest element, that is, it recognizes the maximum value in each area and thus the image becomes smaller as all the extra information is not considered by the model.

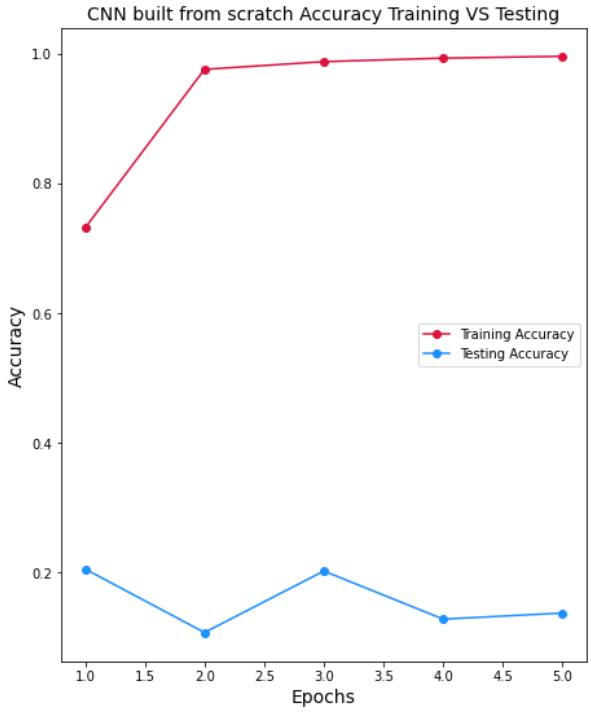
Then we have our drop out layer. It prevents the network from overfitting. It deactivates the nodes so that they don't have any effect on the output. The input to the dense layer should be in 1 dimensional array. So we use the flatten layer to convert the data in to 1 dimensional vector making it suitable for input to the next layer.

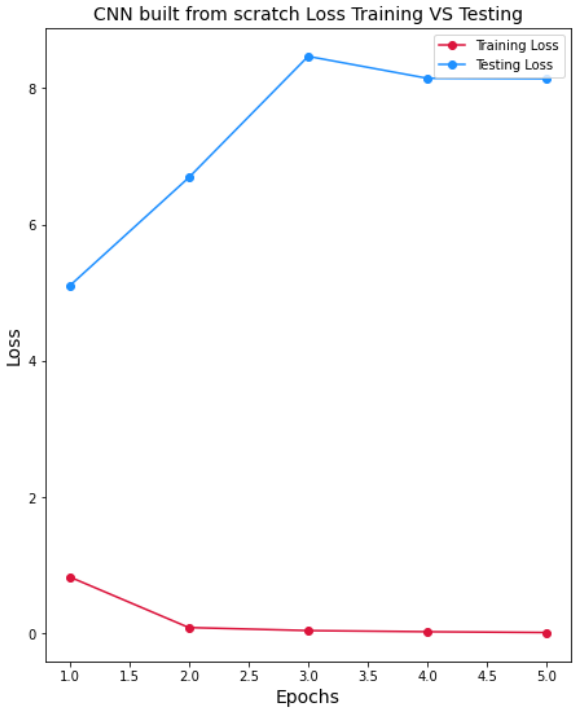
In dense layer each neuron receives input from all the neurons from the preceding layer. This layer can’t have a multidimensional array as an input so flatten layer is used before this layer to get the input in this layer as 1-D array. This dense layer has softmax activation function. Softmax converts a vector of value to a probability distribution. This function is used for multiclass classification problems. This function here has 10 units (10 neurons). Each of these neurons represent one class as c0, c1, c2 and so on till c9. All these 10 neurons return respective probabilities for the input image. The class which will have the highest probability will be considered as the output for that image. Then we are configuring the model for training using model.compile(). We are using optimizers to get results faster. We are using RMSprop optimizer ((Root Mean Squared Propagation- it implements RMSprop algorithm). It decreases the number of function evaluations and improves speed and performance for training a specific model. Here we are considering a learning rate of 0.0001. It helps in configuring our network. When model weights are updated, it helps to control change to the model for the estimated error. Decay helps to slowly reduce or

decay the learning rate so that at the end of the training a local minimum is reached. We have a loss function = Categorical cross entropy, that is used when we have a multiclass classification problem. Note: SoftMax activation function is recommended to use, when we are using categorical cross entropy function.

## Convolutional Neural Network from Scratch

- Using the early stopping helps to stop a training when a monitored metric has stopped improving. We will be monitoring the loss on the validation dataset to end the training.
- min_delta is the minimum change in the monitored quantity to be considered as an improvement. Any change less than min_delta, will count as no improvement.
- Patience specifies number of epochs after which the training will be stopped if there is no improvement.

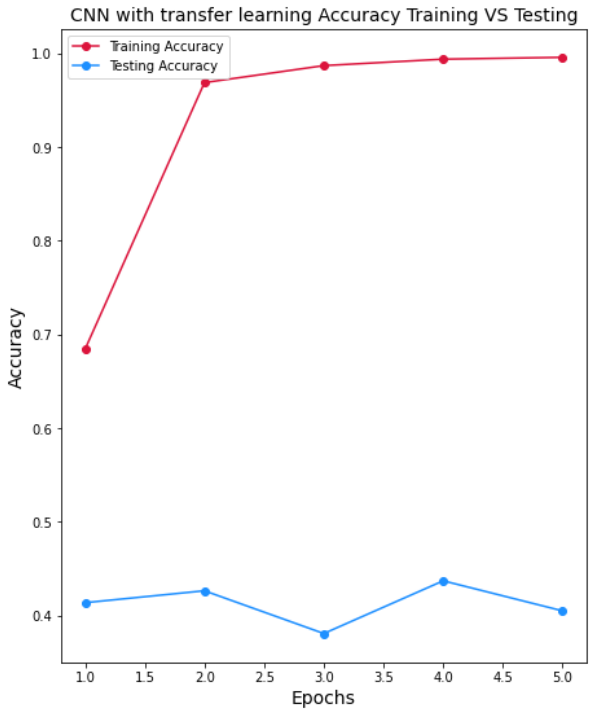


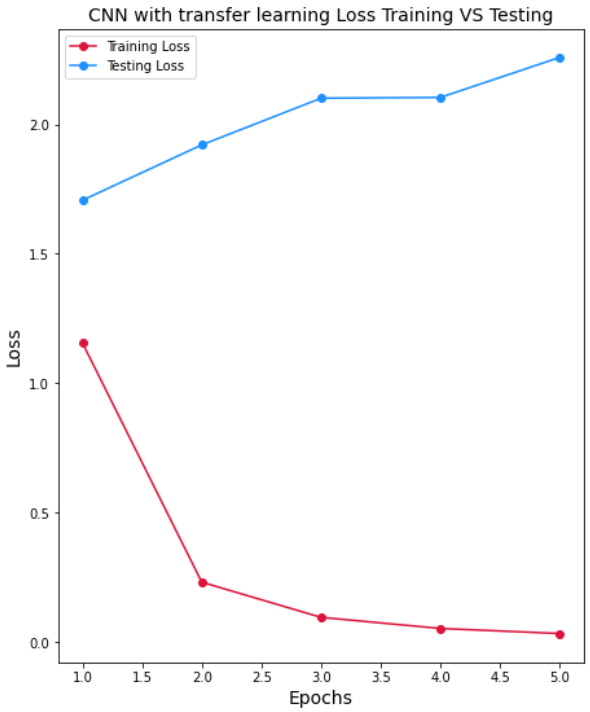




## Convolutional Neural Network with Transfer Learning using VGG16

- Here we will be using transfer learning. It is method to take a model that is already trained on a large dataset and then we tranfer its knowledge to a smalller dataset.
- It improves the efficiency when we are training new models. Here we will be using VGG16 pre trained model, this was trained on imagenet dataset.
- The Initializers are used to set the initial random weight for the layers
- we are using he normal initializer because it works really well with non linear activation function and here we are using ReLu activation function.

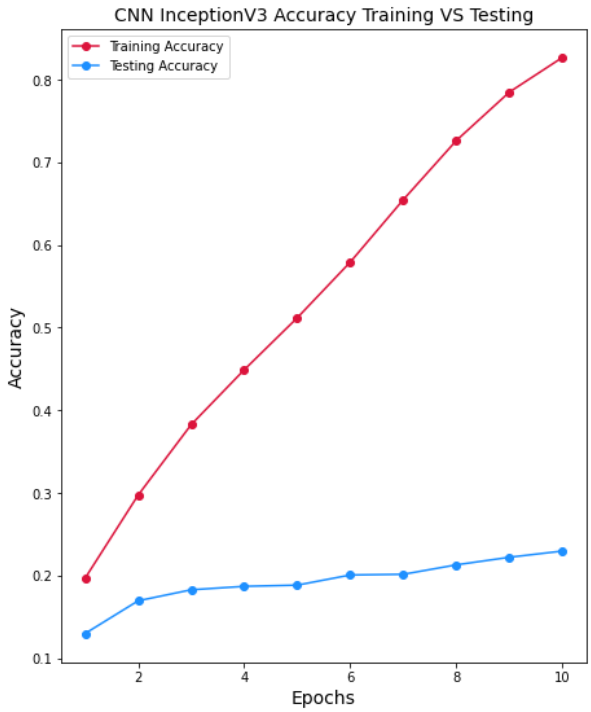
- We have specified input layer as false so we need to specify the input shape. Here we have input shape as (256,256,3) - that is the width , height and 3 input channels.
- Maxpooling is used for feature extraction
- kernel size here specifies the height and the width of the sliding window and strides specifies the amount by which the filter shifts.
- we are using Adam optimizer(Adaptive Moment Estimation). It is appropriate for non-stationary objects. Adam optimizer involves a combination of two gradient descent methodologies: 1) Momentum, 2)Root Mean Square Propagation (RMSP). It takes up the positive attributes from these two methods and builds a more optimized gradient decsent.

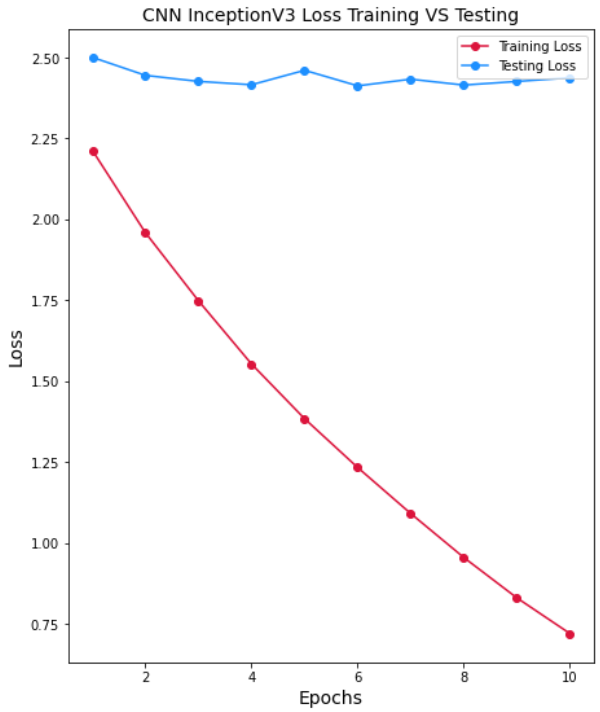



## Convolutional Neural Network with Transfer Learning using InceptionV3

- Inception V3 is a well known image recognition model, and it has been used on Imagenet Dataset and has been shown to have achieved greater than 78% accuracy.
- As we see from the last plot( VGG16), where we used adam optimizer, which is over shooting and over fitting the model
- so in this model we will use SGD optimizer which gradually moves towards local minima
- GlorotNormal initializer - it is very similar to he initializer but it also considers fan in and fan out, where
- Fan-in defines the maximum number of inputs that a system can accept, and,
- Fan-out defines the maximum number of inputs that the output of a system can feed to other systems.




# Evaluation and Selection of Model

```
The test accuracy of the CNN model built from scratch is 0.47540473225404734
```

```
The test accuracy of the VGG16 model is 0.6755915317559154
```

```
The test accuracy of the VGG16 model is 0.4364881693648817
```

***Summary:***

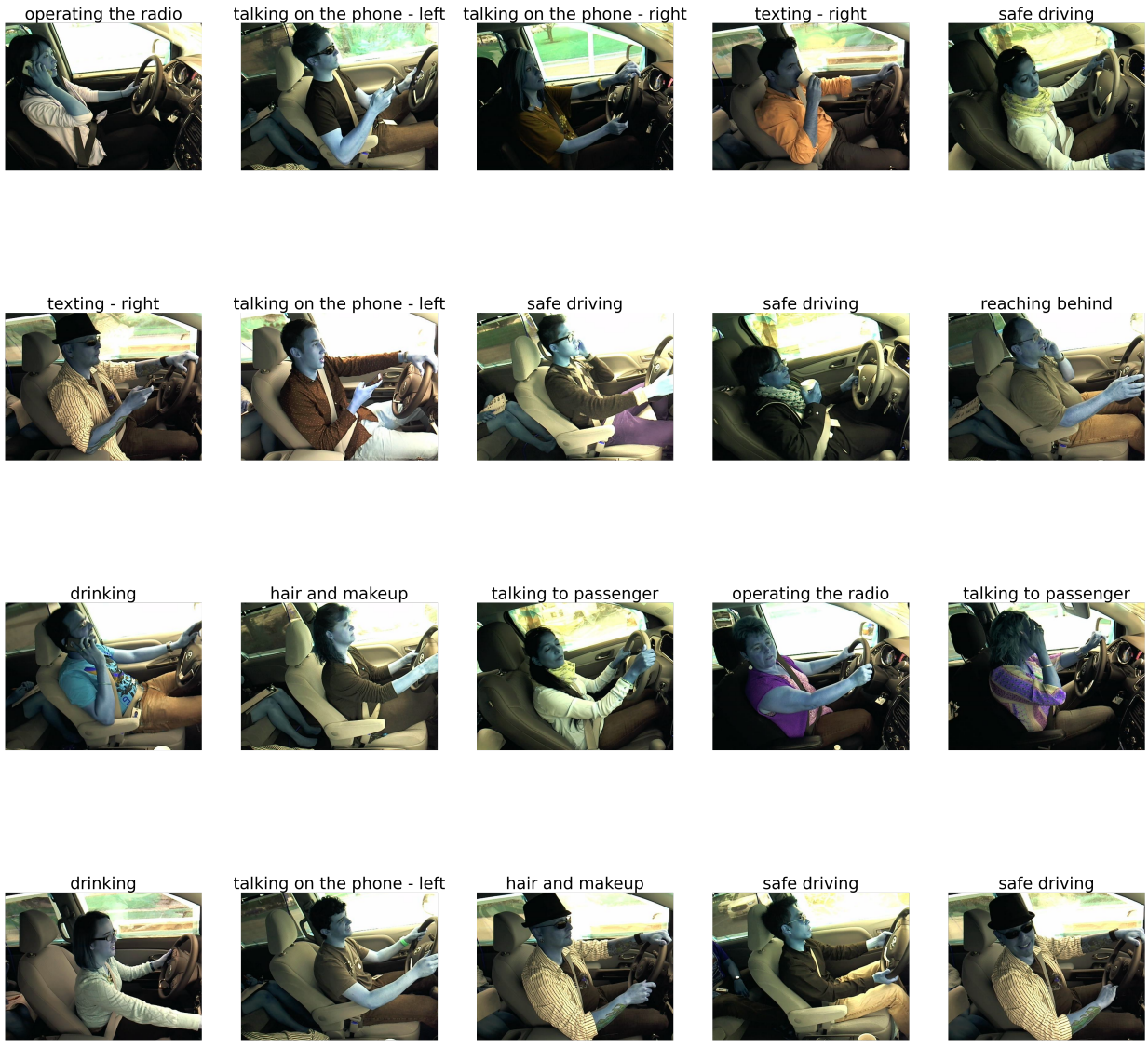
Out[45]:

| | Models | Accuracy |
|---|---|---|
| **0** | CNN from scratch | 47.540473 |
| **1** | CNN using VGG16 | 67.559153 |
| **2** | CNN using InceptionV3 | 43.648817 |

- Here we see that the CNN model built using VGG16 gives the best accuracy among all. The accuray is 67.55%
- Now we will test this model on test images (which were provided in the test folder). We will randomly select 20 images out of all the test images to test on.

## Evaluating final model on test images (from test folder)

### Making prediction on test dataset using VGG16 model




# Conclusion and Future Scope:

- After evaluating all the three models, i.e, the convolutional neural network that we built from scratch, the transfer learning models based on VGG16 and Inception V3, we see that VGG16 works compartively best among all of them with an accuracy of 67.55%.
- One of the reasons, why VGG16 is working better than other models is because we have limited amount of data here. Its very difficulat to build a deep network on this liited amount of data, else our model will overfit.
- If we build this model from scratch still we won't be able to tune all the parameters because we don't have enough image data.

- VGG16 works really good on ImageNet data because the weights of that model is very optimized.
- We can try various hyperparameter tuning like trying different combinations. We can try various kernel size, different strides, padding and work to improve the model performance.

# References


- https://medium.com/@abbz.hamil/distracted-driver-detection-d26f42905f87 (https://medium.com/@abbz.hamil/distracted-driver-detection-d26f42905f87)
- https://ai.plainenglish.io/distracted-driver-detection-using-machine-and-deep-learning-techniques-1ba7e7ce0225 (https://ai.plainenglish.io/distracted-driver-detection-using-machine-and-deep-learning-techniques-1ba7e7ce0225)
- https://towardsdatascience.com/distracted-driver-detection-using-deep-learning-e893715e02a4 (https://towardsdatascience.com/distracted-driver-detection-using-deep-learning-e893715e02a4)
- https://www.upgrad.com/blog/using-convolutional-neural-network-for-image-classification/#:~:text=The%20reason%20CNN%20is%20so,image%20classification%20using% (https://www.upgrad.com/blog/using-convolutional-neural-network-for-image-classification/#:~:text=The%20reason%20CNN%20is%20so,image%20classification%20using%
- https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python (https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python)
- https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480#:~:text=Flattening%20is%20converting%20the%20data,called%20a%20fully%2D (https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480#:~:text=Flattening%20is%20converting%20the%20data,called%20a%20fully%2D
- https://vijayabhaskar96.medium.com/tutorial-on-keras-flow-from-dataframe-1fd4493d237c (https://vijayabhaskar96.medium.com/tutorial-on-keras-flow-from-dataframe-1fd4493d237c)
- https://keras.io/api/applications/vgg/ (https://keras.io/api/applications/vgg/)
- https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator (https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator