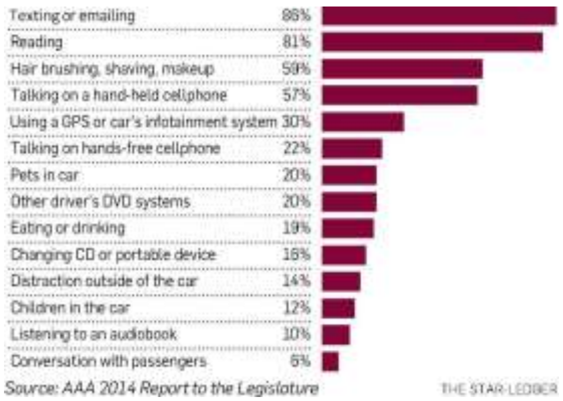
State Farm Distracted Driver Identification

Aditya Lahiri, Ali Akbar Shafi, Rishi Laddha, and Shirish Pandagare

***Abstract* - The objective of this project was to design a classifier that can identify from images of drivers whether they are driving the car safely or not. The project provided a training and a testing set for designing the classifier. We used the training set to create a convolutional neural network to classify these images into safe or one of the nine distracted driving categories. Our best model achieved a multi-class logloss score of 1.274 and a mean validation accuracy of 95.34%**

I. Introduction

The world is witnessing a rapid transformation in the way we commute and transport ourselves. The wheels combined with cellular technologies often result in fatalities rater than making our cars more user friendly. Moreover, as the number of vehicles on road increases, the fatal accidents increase too. Road traffic accidents take massive toll of lives beside causing huge economic damages. Every year about 1.2 million people fall victim for road accidents all over the world and we lose about $158 billion annually *(Organisation, 2009).* Distracted driving is one of the prime reasons for these accidents. Around 425,000 people get injured and 3,000 people die by distracted drivers every year. *(Motor Vehicle Safety, 2017).*



Source: AAA 2014 Report to the Legislature

THE STAR-LEDGER

Figure 1 Driving Distraction poll (AAA report to the legislature, 2014)

It is a common myth that humans are good at multi-tasking. For humans multi-tasking is usually just a task switching activity which causes cognitive tunneling and the reaction time decreases exponentially as number of parallel activities increase. A meta-analysis of several studies reveals that cell phone usage decreases the reaction time by 0.25 seconds. Other than cell phone use, drivers can be distracted by several external sources and events outside/inside the vehicle.

Several organizations such as *The National Highway Traffic Safety Administration* (NHTSA) work to reduce the occurrence of distracted driving and raise awareness of its dangers. Driver distraction is a specific type of driver inattention. Distraction occurs when drivers divert their attention from the driving task to focus on some other activity. Oftentimes, discussions regarding distracted driving center around cell phone usage and texting, but distracted driving can also include other activities like eating, talking to other passengers, or adjusting the radio or climate controls. The goal of this project is to use deep learning techniques to accurately detect any other activity along with driving and eventually classify them as a distracted/safe driver.

II. Dataset

With the aim of distracted driver distraction, State Farm has provided a public dataset consisting of several images of drivers. This proceeding uses this dataset from Kaggle to analyze the driver's posture and classify it between 10 different classes defined in the dataset.

Table 1 possible class labels

C0	Safe Driving	**C5**	operating the radio
C1	Texting - right	**C6**	drinking
C2	Talking on the phone - right	**C7**	reaching behind
C3	Texting - Left	**C8**	hair and makeup
C4	talking on the phone - left	**C9**	talking to passenger

The dataset consisted of full colored images of drivers operating their vehicles. The training set consisted of 20187 images and 79726 of test images. Each image had a dimension of 640 by 480. The training images are labeled as "safe" or one of the nine categories of "distracted driving" and on the other hand the testing images are unlabeled. Table 1 describes the various labels according to which the training data is categorized, and Fig 2 shows each of the instances of the various categories from the training set.

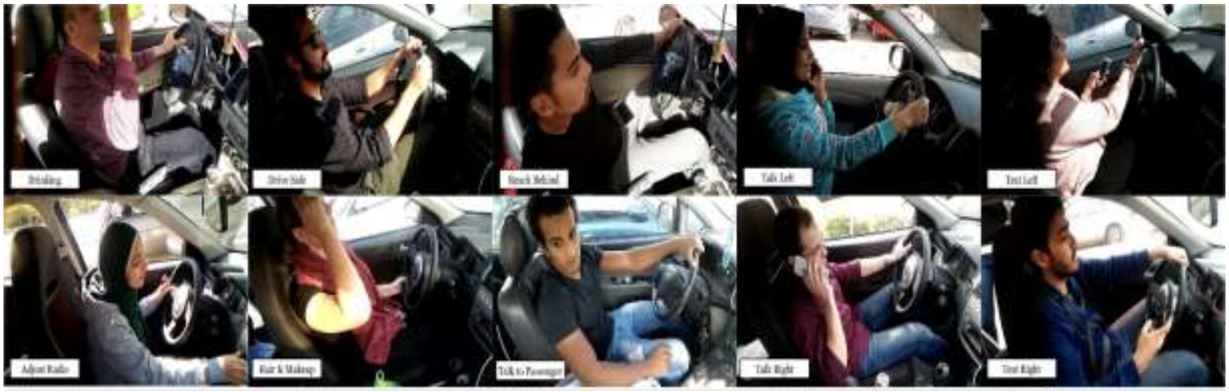


Fig 2: Drivers belonging to various categories

Ten percent of the training images were randomly sampled and separated for the purpose of validation. Image preprocessing was required in order to feed the data in the neural networks. Each model has own input size constraint, hence image resizing and pixel values normalization was performed beforehand.

III. Evaluation Metric

The dataset was provided by State Farm on Kaggle as a competition and the performance metric designed by Kaggle was *multiclass logarithmic loss* which is calculated as follows:

$$\log loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \log(p_{ij})$$

where N is the number of images in the test set, M is the number of image class labels, log is the natural logarithm, Y_{ij} is 1 if observation i belongs to class j and 0 otherwise, and P_{ij} is the predicted probability that observation i belongs to class j. In, addition to logarithmic loss we also used the mean validation classification accuracy to judge the performance of our model.

IV. Methodology

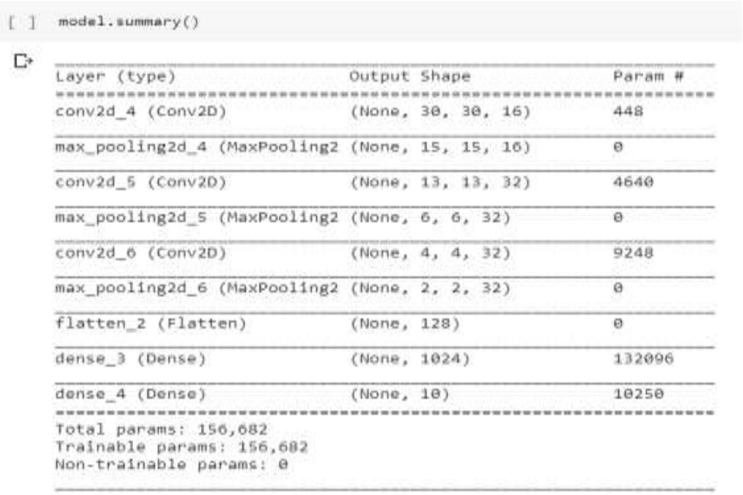
The dataset for this proceeding had images that can be classified in 10 distinguished classes. The Baseline model that could be used to train and then classify could potentially be a fully connected neural network. For the purpose of best accuracy, we utilized the Convolutional Neural Network which in contemporary literature has been identified as one of the ideal paradigms for the image classification problems. The CNN identifies and discovers complex feature relations by back-propagation.

We mainly worked on 3 different deep learning frame-work to achieve minimum log-loss and maximum accuracy. All these models are improved versions of the previous model. We first started with a simple 3 convolutional Network and 3 Max-pooling networks, followed by the pre-trained VGG-16 model and VGG-16 stacked with the max pooling and combination of 3 dense layer as our last model. The following is the description of the models.

Model A:

In this model, we have used a stack of convolutional layer, pooling layer and Fully-connected layer. The architecture of this model can be seen in Fig 3. The input size of the image is taken as 32 x 32 which significantly reduced the computation. The images were preprocessed by resizing it to 32 x 32 and normalizing the pixel values, to feed it in the model.

The architecture for this model is a stack of 3 pairs of convolutional layers followed by a pooling layer, a fully connected layer and softmax layer respectively. Multiple filters are used at each convolutional layer for different type of feature extraction. Once the architecture was defined, we complied the model by using ‘catergorical_crossentropy’ as a loss function because it is a multiclass problem. We used ‘*Adam’*, a stochastic optimizer to optimize the model parameter. Once model’s architecture was defined and complied, we have trained our model on 90% of training data and validated it in each epoch with the remaining 10% of the training dataset. Epoch of 50 and batch size of 64 was used to fit the model. Using this model, we were able to achieve of mean validation accuracy of *92.04%* and Kaggle score of *2.121*.



```
[ ]  model.summary()

     _________________________________________________________________
     Layer (type)                 Output Shape              Param #
     =================================================================
     conv2d_4 (Conv2D)            (None, 30, 30, 16)        448
     _________________________________________________________________
     max_pooling2d_4 (MaxPooling2 (None, 15, 15, 16)        0
     _________________________________________________________________
     conv2d_5 (Conv2D)            (None, 13, 13, 32)        4640
     _________________________________________________________________
     max_pooling2d_5 (MaxPooling2 (None, 6, 6, 32)          0
     _________________________________________________________________
     conv2d_6 (Conv2D)            (None, 4, 4, 32)          9248
     _________________________________________________________________
     max_pooling2d_6 (MaxPooling2 (None, 2, 2, 32)          0
     _________________________________________________________________
     flatten_2 (Flatten)          (None, 128)               0
     _________________________________________________________________
     dense_3 (Dense)              (None, 1024)              132096
     _________________________________________________________________
     dense_4 (Dense)              (None, 10)                10250
     =================================================================
     Total params: 156,682
     Trainable params: 156,682
     Non-trainable params: 0
     _________________________________________________________________
```

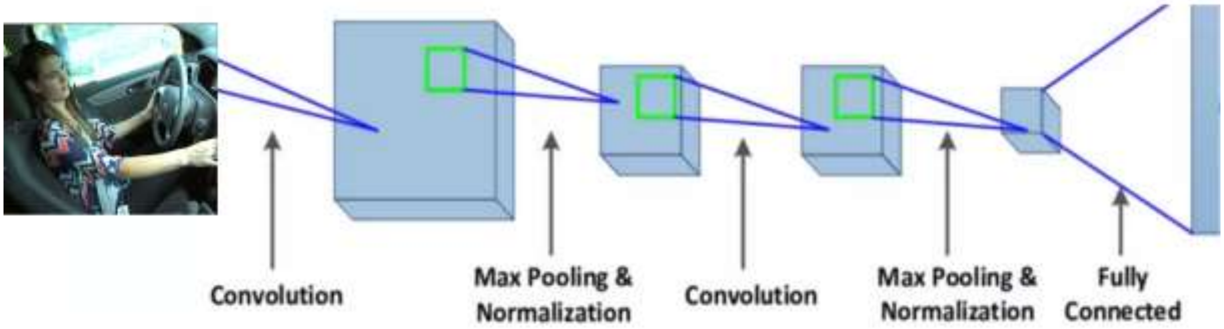
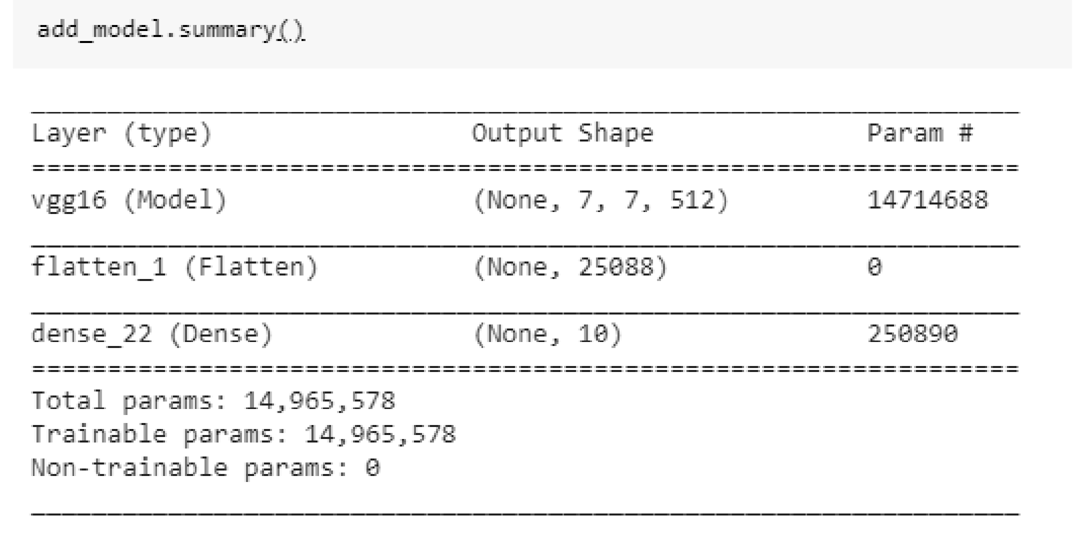


Figure 3: Model A (top) with 3 convolutional layers and 3 max pooling layers arrange alternatively followed by a densely connected layer. Schema for CNN (bottom)

Model B:

To further expand the horizon of the modeling, we used pre-trained VGG-16 convolutional neural network. University of Oxford Visual geometry Group (VGG) developed a 16-layer CNN called VGG-16. We used a pre-trained VGG16 model which is trained over thousands of images from ImageNet and has performed well in classifying 1000 different classes. However, we were not sure how this model will perform in detecting the action performed by the same object. It was also observed that, before training this pre-trained model with our training dataset, it was able to detect seat belt from the training images.

VGG-16 is stack of convolutional, max pooling and fully connected layers. The detail architecture is displayed in the Fig 4. The architecture of this model consists of pre-trained VGG-16 followed by fully a connected layer. The model was compiled using stochastic gradient descent optimizer with the learning rate of 0.0001. As the input for this model is 244 x 244, the images were accordingly resized and normalized. The model was trained on the training dataset and have achieved mean validation accuracy of *93.26%* and Kaggle score of *1.875*.



```
add_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688  
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0         
_________________________________________________________________
dense_22 (Dense)             (None, 10)                250890    
=================================================================
Total params: 14,965,578
Trainable params: 14,965,578
Non-trainable params: 0
_________________________________________________________________
```

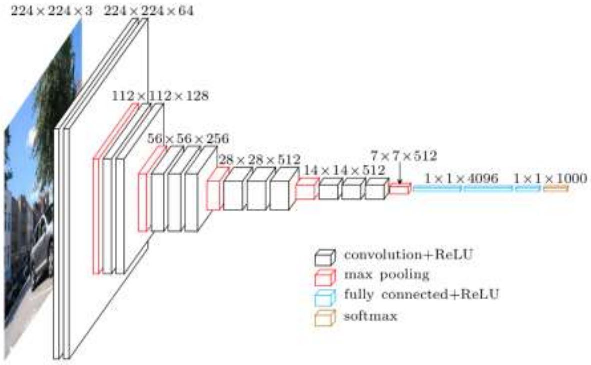
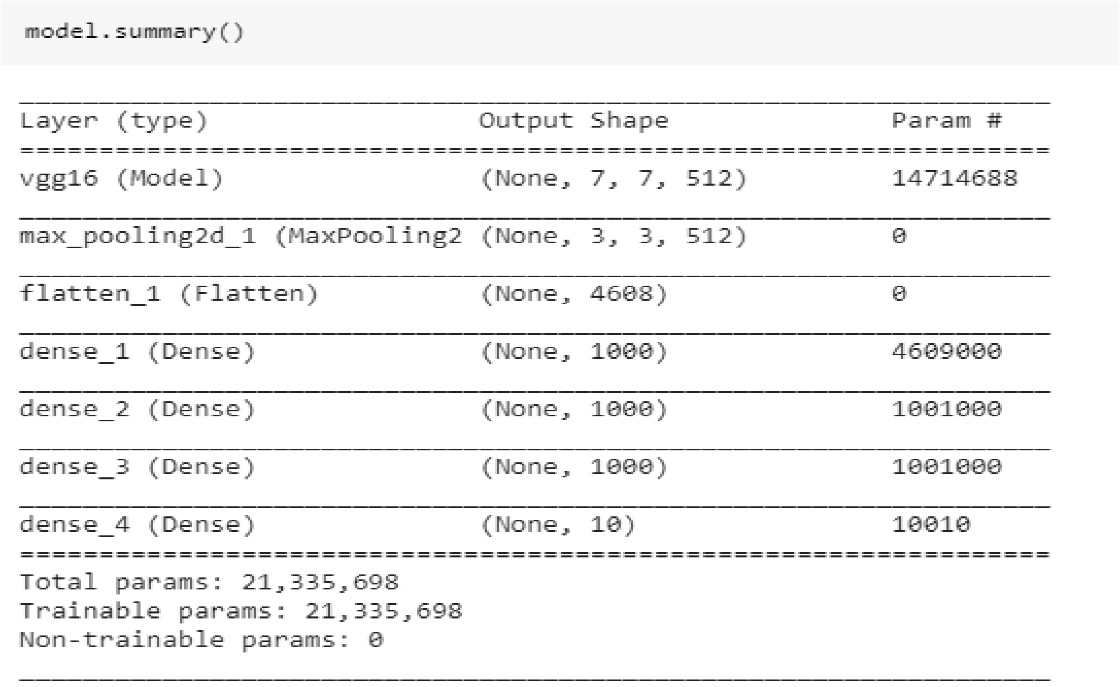


Figure 4: Model B architecture (top) and VGG-16 architecture (bottom)

Model C:

Looking at the performance of VGG-16 in the previous model, we decided to increase the layers in the next model to extract more features. This model has a pre-trained VGG-16 followed by a Max-pooling layers to reduce the parameters followed by the 3 fully connected dense layer. The model has the same input size of 244x244. After the model architecture was decided, model was compiled with the stochastic gradient descent optimizer with the learning rate of 0.0001. The model was trained on the training dataset and validated at each epoch with the 10 % of the validation data from the training set. The epoch for the model was 50 with the batch size of 20. The model was tested and has performed the best among all three of the models with the mean validation accuracy of *95.35%* and Kaggle score of *1.274*.



```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              4609000
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_3 (Dense)              (None, 1000)              1001000
_________________________________________________________________
dense_4 (Dense)              (None, 10)                10010
=================================================================
Total params: 21,335,698
Trainable params: 21,335,698
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: Model C, VGG-16 stacked with pooling layer and 3 fully connected dense layer.

VI. Model accuracy and performance

The following table lists the mean validation accuracy and logloss.

Table 2 Model Performance

Model	Testing Logloss	Mean Validation Accuracy
Model A	2.121	92.04%
Model B	1.875	93.26%
Model C	1.274	95.34%

Based on our experimental analysis we found that the model C has performed the best among all three models. Pre-trained VGG-16 model stacked with pooling layer and 3 dense layers provided us with the highest validation accuracy and lowest Log-loss.

Model C is an improved version of Model B with extra fully connected layers compared to VGG 16, however, Model C had a layer of L2 regularization at the end of its fully connected layers. These models were trained for 50 epochs and the average run time was about 45 minutes for training and 10 minutes for testing. These models were tested on Google's Collaboratory platform using its GPU services.

VII. Discussion

The models we designed significantly underperformed compared to some of the models that were implemented by other developers on Kaggle. The best performing model on Kaggle had a logloss score of 0.08 which is much smaller compared to logloss obtained by our best performing classifier Model C. Our objective was to increase the Mean Validation Accuracy and thereby reduce the testing logloss. However, it seems that our models were overfitting significantly, even after implementing regularization (Model C). A complex model VGG16 did not produce a better result because of high flexibility neither did a very simple model (Model A) because of high bias. The models which we designed (Model A) were significantly less complex because we wanted to make sure that trained neural network can predict images very quickly. However, it didn't perform well as compared to models with a greater number of layers. We can deduce form this that larger number of layers may extract more features, however it comes at the cost of computation and high variability probably due to overfitting. In the future we would like to get a better understanding of how every layer of CNN works, how are the weights being adjusted for each of the features. Another area which we would like to optimize our models is by converting the RGB images to grayscale and reducing the space and time complexity.

VIII. References

[1] *AAA report to the legislature.* (2014).

[2] Bíža, O. (2017). *State of Deep Learning in Computer Vision*. Retrieved from tech.showmax.com

[3] Motor Vehicle Safety. (2017). *Motor Vehicle Safety*. Retrieved from Centers for Disease Control and Prevention: *https://www.cdc.gov/motorvehiclesafety/distracted_driving/*

[4] Organisation, W. H. (2009). *World Health Organisation*. Retrieved from Global status report on road safety: time for: *http://apps.who.int/iris/bitstre*

[5] Safety, T. D. (2017). *Texas Driver handbook.*

[6] Simonyan, K. a. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition.* Retrieved from Arxiv.org: *https://arxiv.org/abs/1409.1556*