Distracted Driver Detection

Predict the likelihood of what the driver is doing in each picture ?

Agenda

- Business problem
- Approach
- Data / Data wrangling
- Deep Learning Models
- Training & Predictive modeling
- Conclusion
- Future Scope of work

Business Problem

According to the CDC motor vehicle safety division, one in five car accidents is caused by a distracted driver. Sadly, this translates to 425,000 people injured and 3,000 people killed by distracted driving every year.

State Farm hopes to improve these alarming statistics, and better insure their customers, by testing whether dashboard cameras can automatically detect drivers engaging in distracted behaviors. Given a dataset of 2D dashboard camera images, State Farm is challenging Kagglers to classify each driver's behavior. Are they driving attentively, wearing their seatbelt, or taking a selfie with their friends in the backseat?

Dataset contains driver images, each taken in a car with a driver doing something in the car (texting, eating, talking on the phone, makeup, reaching behind, etc).

Goal is to predict the likelihood of what the driver is doing in each picture.

Data Wrangling : Source - Kaggle

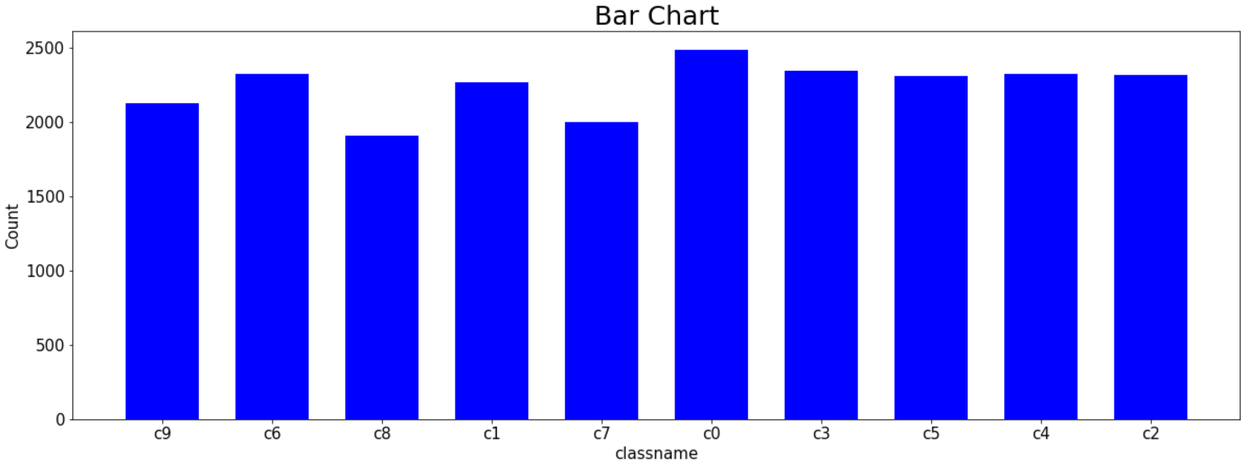
Image Counts

c9	2129
c6	2325
c2	2317
c3	2346
c4	2326
c8	1911
c0	2489
c7	2002
c5	2312
c1	2267

Dataset Description:

- **Image Size** - 480 X 640 pixels
- **Training Images count** - 22424 images
- **Image type** - RGB
- **Image field of view** - Dashboard images with view of Driver and passenger
- **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
- **Loss** - multi-class logarithmic loss

Data Wrangling : Source - Kaggle



Balanced training set for 10 classes

Safe driving
(c0)

Texting with right hand
(c1)

Doing hair & makeup
(c8)

Predictive Modeling -Splitting Test & Train

- Out of the main training dataset, a certain percentage is kept untrained to test the model's performance.
- Training set and validation set are split in following percentages: 80% : 20%.
- On the Validation set, the target labels are hidden, until the performance is evaluated.

Predictive Modeling - Preprocessing

Preprocessing data set before creating ML modeling:

Much of the image manipulation had to be done manually and prior to the machine learning process, since it did not fit into memory using the limited hardware at our disposal.

As a result, the images had to be reduced significantly from 640 x 480 to 64 x 64 and grayscale as depicted in Figure.

Predictive Modeling

Models to be used- Using Convolutional Neural Networks and Transfer Learning (VGG16)

Framework - Keras version

Loss: Categorical Cross Entropy

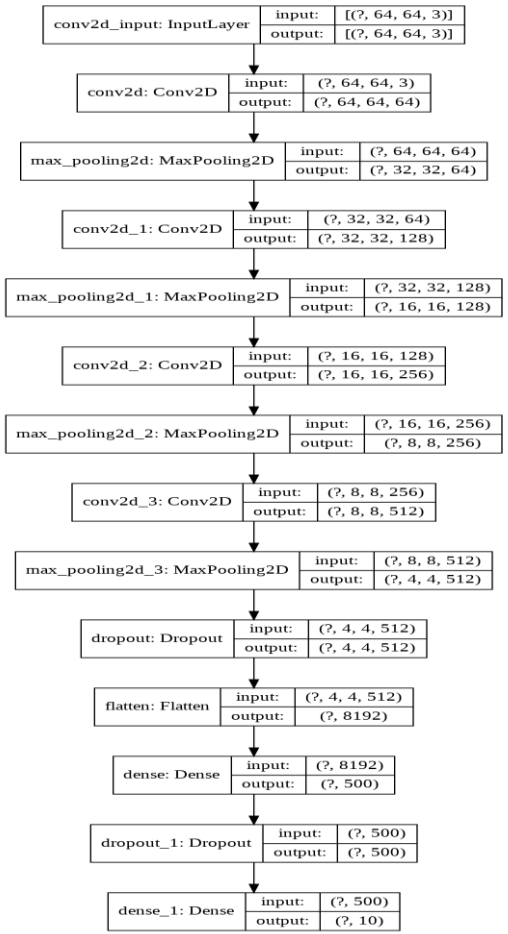
Metrics to be used: Accuracy, Precision, Recall, F1 score & Heatmap (Validation Train Predict Vs Validation Target)

Optimizer: rmsprop

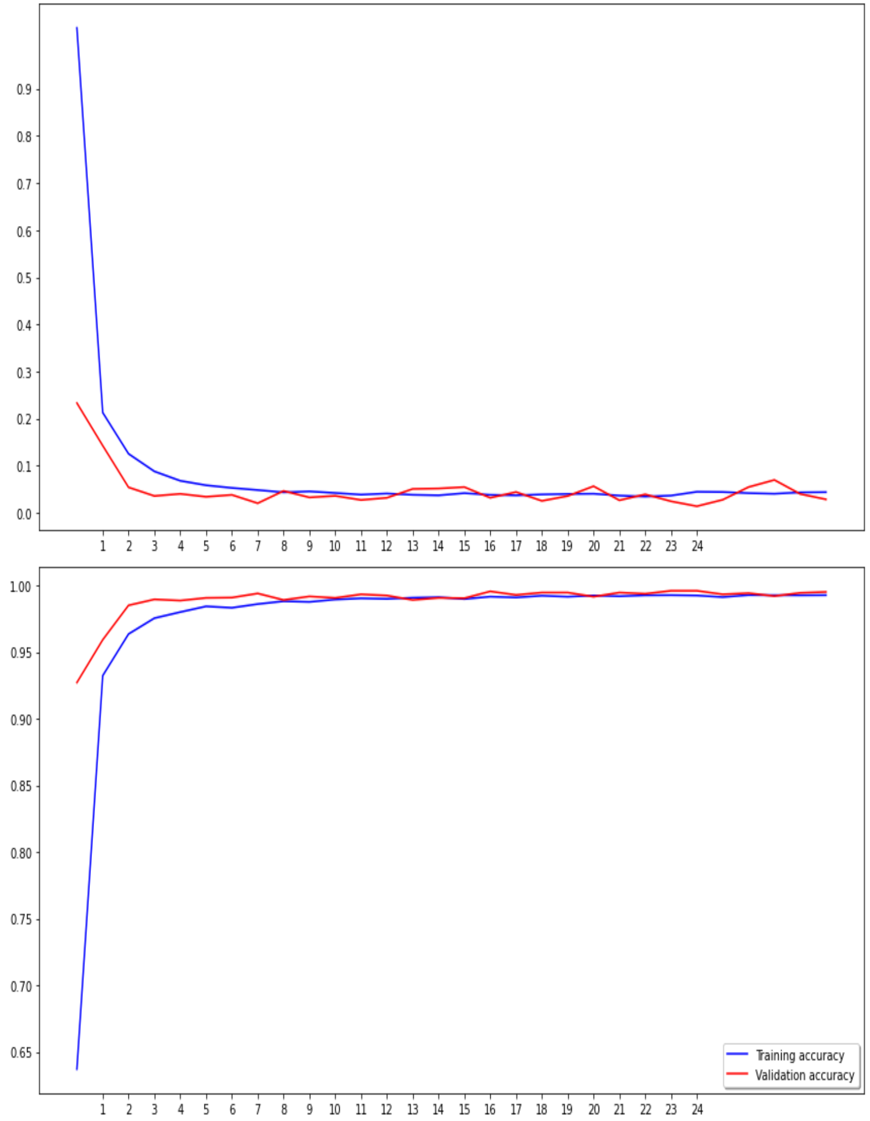
Checkpoint: Model checkpoint by monitoring ‘val_accuracy’ and storing best weights

Output activation type: softmax

Predictive Modeling - CNN

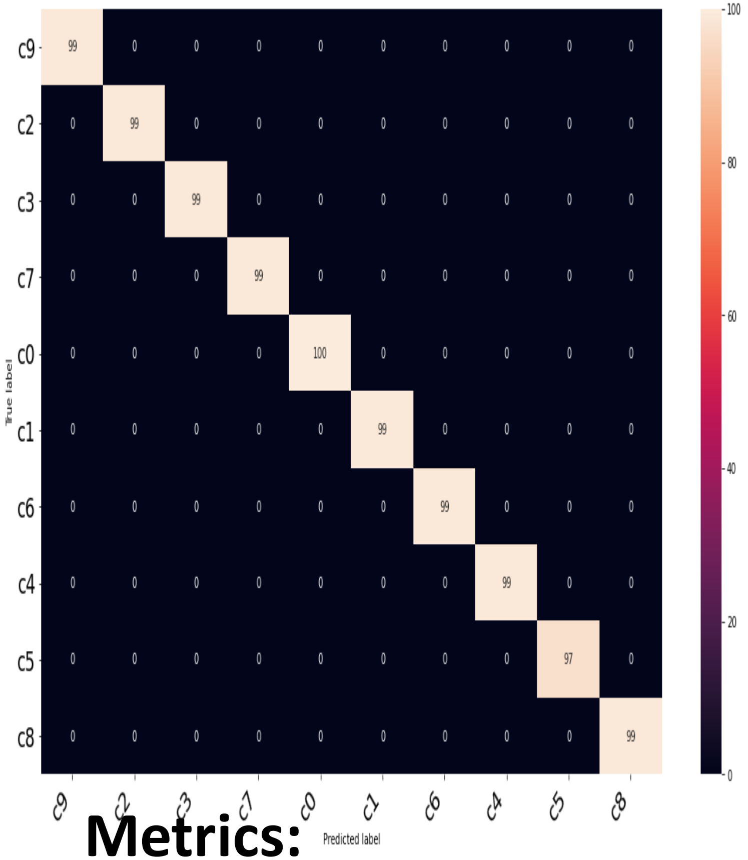


Layer layout



Loss Vs Accuracy for Training & Validation

Heatmap



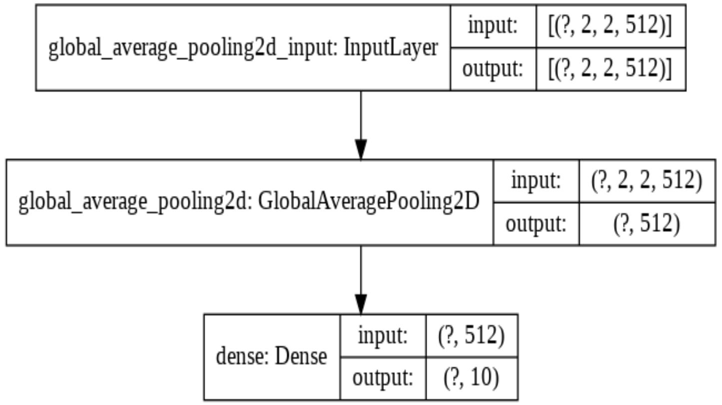
Epchs & Batch size used:

Epoch -30 & Batch size – 40

Metrics:

```
Accuracy: 0.995095
Precision: 0.995100
Recall: 0.995095
F1 score: 0.995088
```

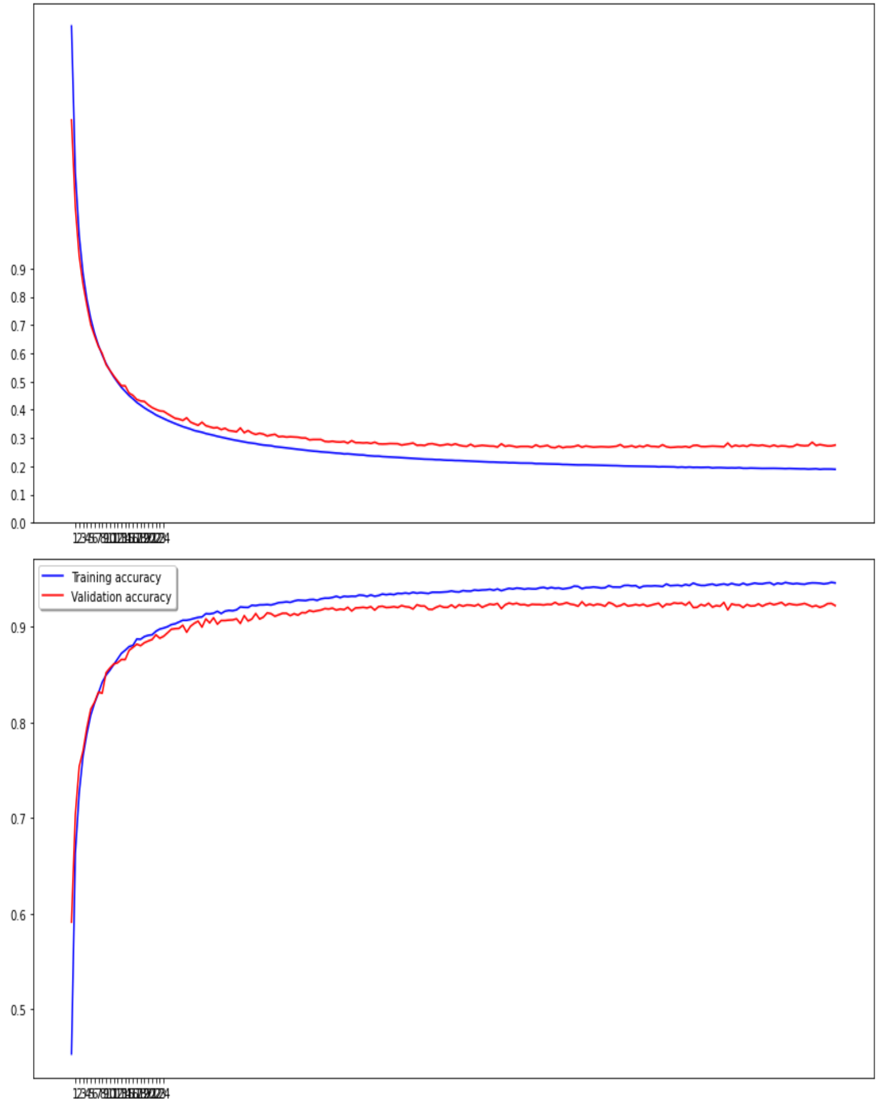
Predictive Modeling – VGG16 Base



Layer layout

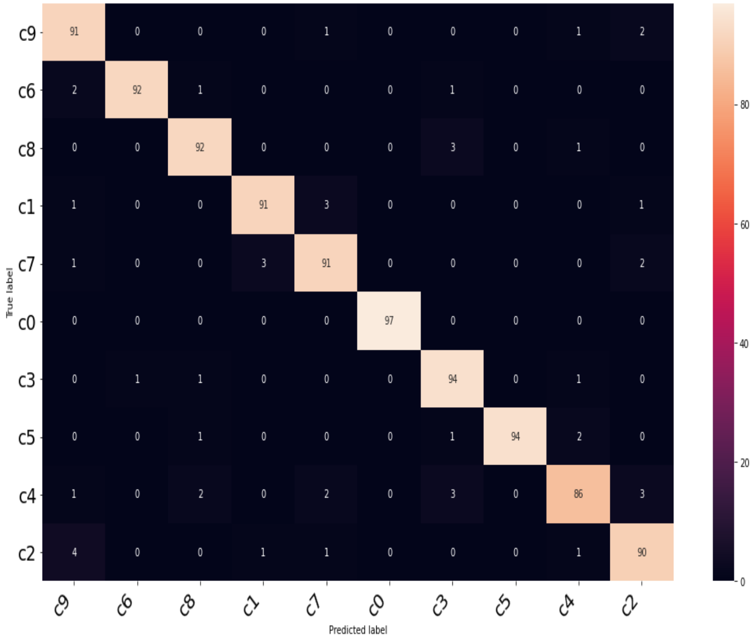
Epchs & Batch size used:

Epoch -200& Batch size – 15



Loss Vs Accuracy for Training & Validation

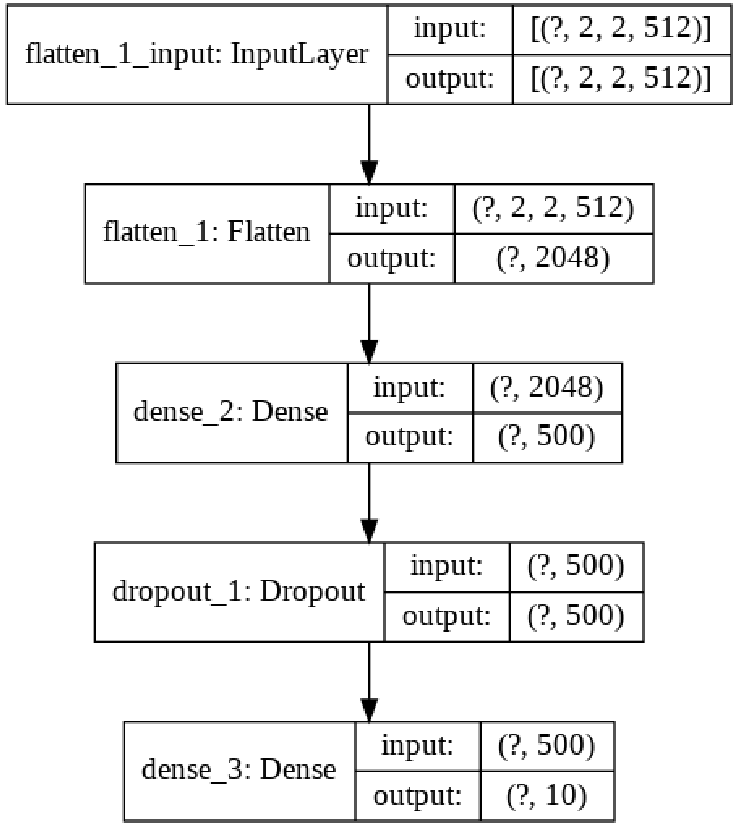
Heatmap



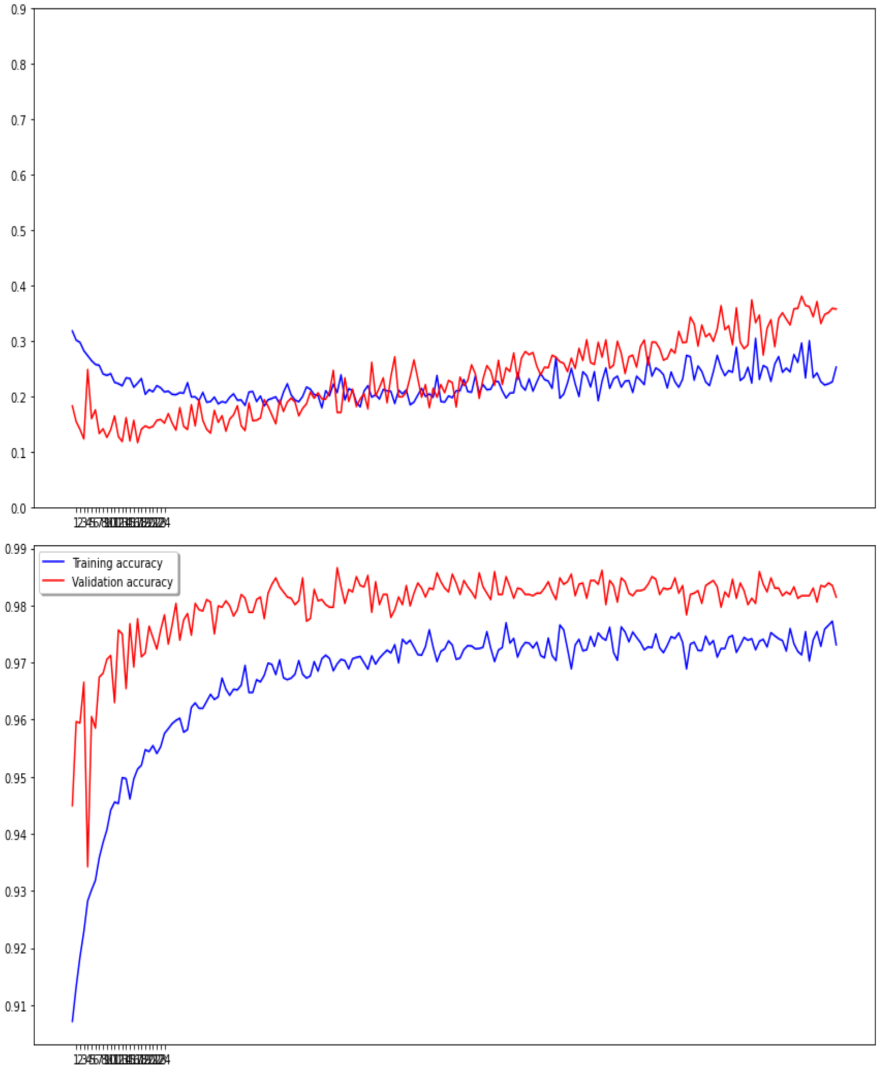
Metrics:

Accuracy: 0.922408
Precision: 0.923282
Recall: 0.922408
F1 score: 0.922602

Predictive Modeling – VGG16 Feature Extraction

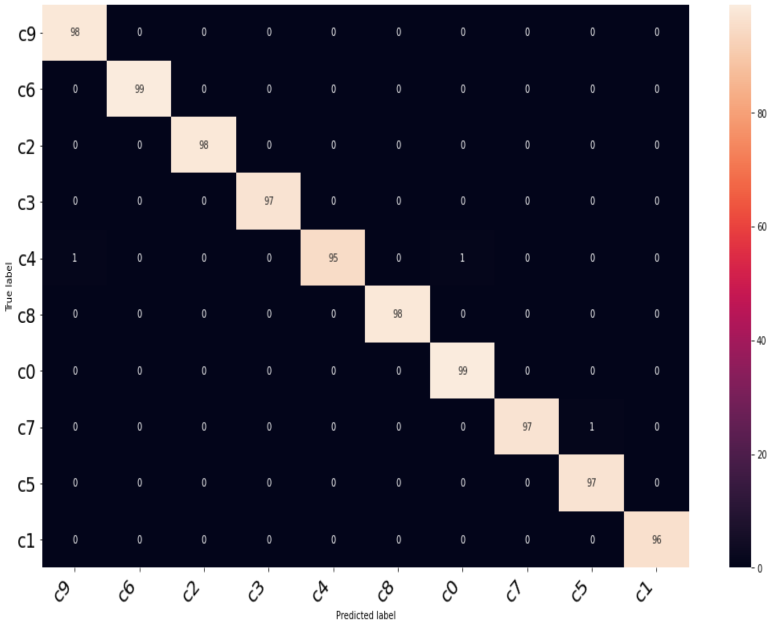


Layer layout



Loss Vs Accuracy for Training & Validation

Heatmap



Epchs & Batch size used:

Epoch -200& Batch size – 15

Metrics:

```
Accuracy: 0.981494
Precision: 0.981659
Recall: 0.981494
F1 score: 0.981507
```

Conclusion

This work has looked at solving the detection of distracted drivers through images obtained from the State Farm Distracted Driver Detection competition on Kaggle.

By using a pre-trained VGG16 network, extracting the bottleneck features, and retraining a new set of fully connected layers, the model was able to achieve 98.15% accuracy on test data.

Despite given the task of classifying very specific classes, the model is evidently able to accomplish that with great success.

Further evaluation revealed that the most miss-labeled class was reaching behind, often confused with the driver talking on their phone with the right hand.

Overall, the model has proven to be very effective at predicting distracted drivers, and will hopefully, one day, aid in preventing further injuries and deaths resulting from distracted driving

Future Scope of Work

- Using ImageDataGenerator instead of dimension converter
- Fine tune a few of the lower layers of the VGG16 network by freezing them and retraining the remaining ones.
- Using Restnet model for training & predicting