Distracted Driver Image Classification for Auto Insurance

IST 718 Project Writeup

Gregory Haben, Ashraf Wan, Victor Yamaykin

Summary

1. **Need for speed when evaluating a customer's driving behavior**
2. **Experimented with supervised and unsupervised machine learning**
3. **Recommend supervised learning with image real-time heatmapping**

Specification

The project's objective is to create a real-time predictive model for identifying distracted driving behaviors using image classification.

Our team expects to address the following questions:

- What are some of the ways of image augmentation to improve the accuracy and speed of the model?
- Will the model be able to predict images taken from different angles?

Observations About the Data

The data set was created by State Farm for a Kaggle competition: https://www.kaggle.com/competitions/state-farm-distracted-driver-detection/data

The data source provided above accesses the 3 files below:

- Training
- Testing
- Final submission example

The three data files include the subject ID, class ID, and the images.

The dataset has a slight imbalance in the number of images for each category:

We can address this imbalance by sampling the same size from each category.

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

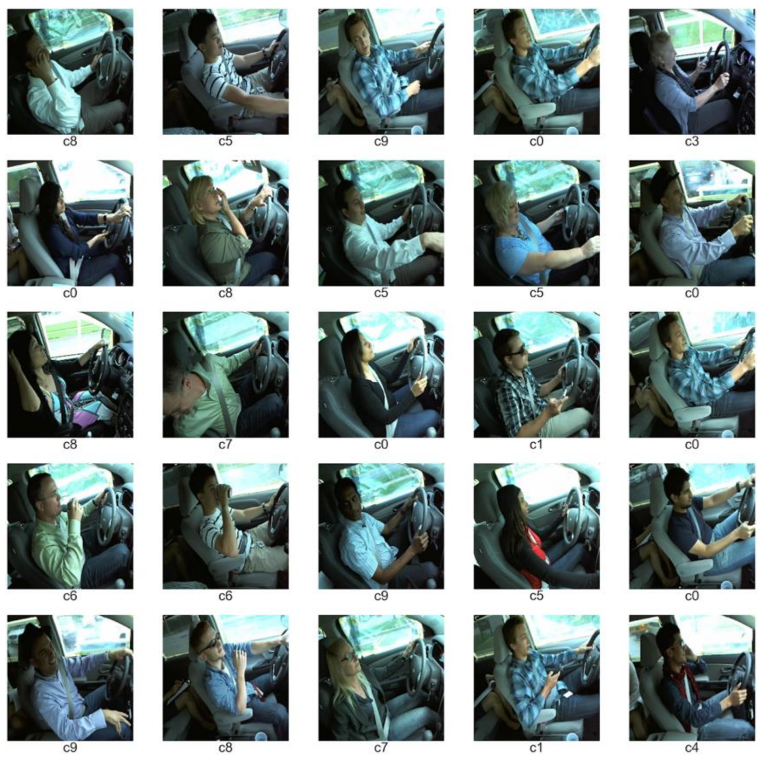
c7: reaching behind

c8: hair and makeup

c9: talking to passenger

We have the labels for the data as folder names. We need to associate the image with the label to train the models. By transforming the images into arrays with Numpy, we can then normalize the values for each pixel to standardize the data.

Example images



Examples of the images in gray scale to reduce the training time:

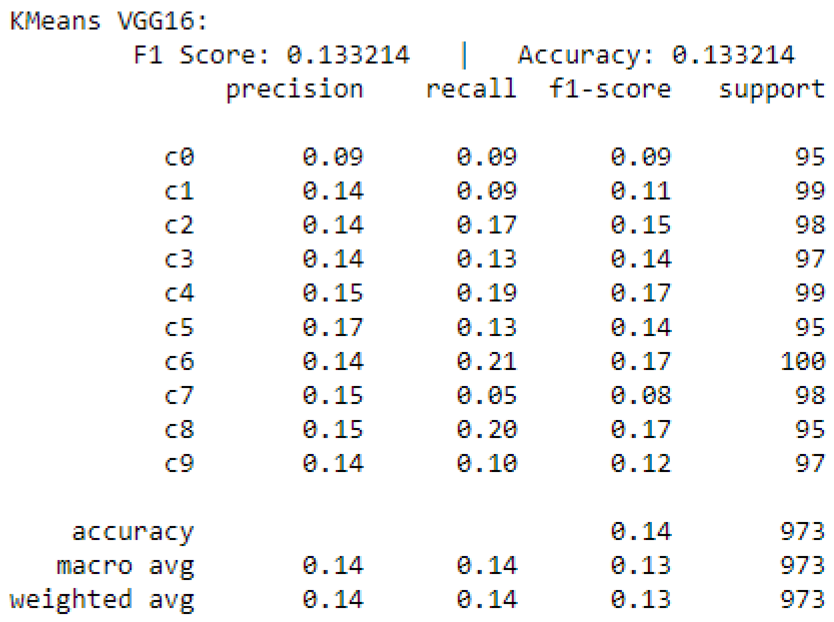
We can take this process a step further with image augmentation.

The Gradient-based Class Activation Mapping (CAM) method will help with drawing heat maps on the images. We explored ways to use the trained model to pinpoint the source of the distraction on the heat map, but we were unable to reuse the trained to draw the heat maps.

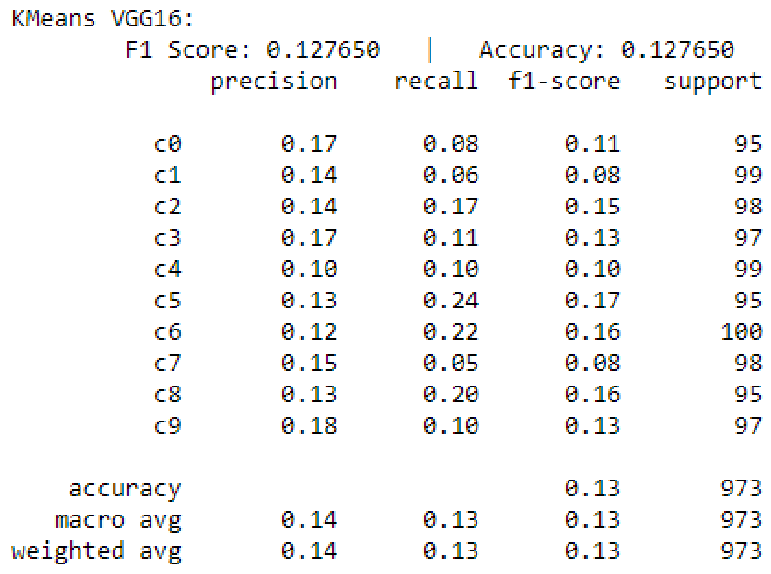
Analysis

We wanted to try to see if the dataset was compatible with any of the unsupervised learning, so we opted to run K-Means clustering. Our reasoning for this is that we theorized that it would be possible for us to cluster the images based on the categories. We ran two different K-Means models, using a sample size for training and testing, with one set with the default images, while the other set with the images with heatmapping. For both tests, principal component analysis (PCA) was just to make the process of the data faster.

The result for the first k-means model using the default images was 13.3% accuracy, which is very low, with category ‘c0’ being the least precise which isn’t good since the model wasn’t able to precisely predict safe driving.


```
KMeans VGG16:
        F1 Score: 0.133214    |   Accuracy: 0.133214
              precision    recall  f1-score   support

          c0       0.09      0.09      0.09        95
          c1       0.14      0.09      0.11        99
          c2       0.14      0.17      0.15        98
          c3       0.14      0.13      0.14        97
          c4       0.15      0.19      0.17        99
          c5       0.17      0.13      0.14        95
          c6       0.14      0.21      0.17       100
          c7       0.15      0.05      0.08        98
          c8       0.15      0.20      0.17        95
          c9       0.14      0.10      0.12        97

    accuracy                           0.14       973
   macro avg       0.14      0.14      0.13       973
weighted avg       0.14      0.14      0.13       973
```

The result for the second k-means model using the images with heatmapping was 12.77%, which is lower than the first model. However, this test was able to be more precise at predicting safe driving compared to the first model. This model had the lowest precision percentage for ‘c4’, which is talking on the phone using left hand.

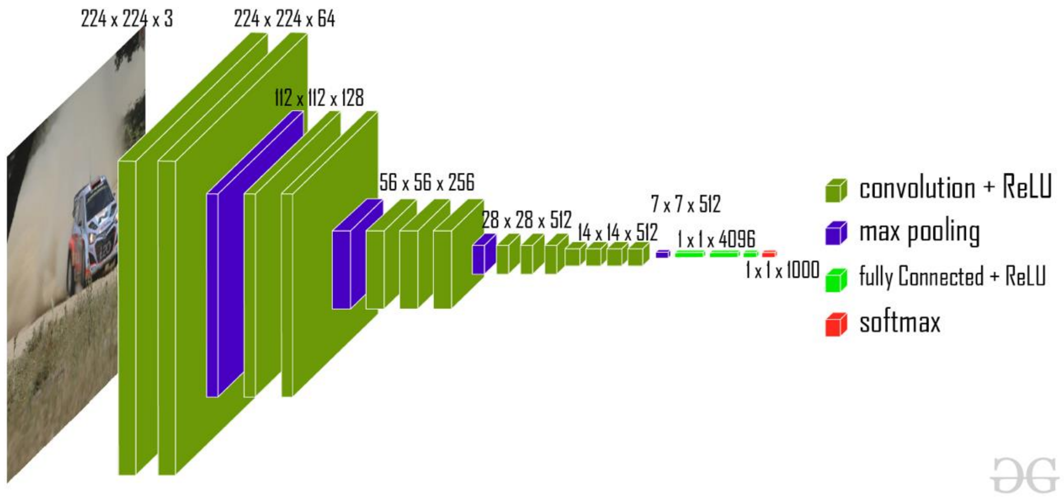


```
KMeans VGG16:
        F1 Score: 0.127650    |   Accuracy: 0.127650
              precision    recall  f1-score   support

          c0       0.17      0.08      0.11        95
          c1       0.14      0.06      0.08        99
          c2       0.14      0.17      0.15        98
          c3       0.17      0.11      0.13        97
          c4       0.10      0.10      0.10        99
          c5       0.13      0.24      0.17        95
          c6       0.12      0.22      0.16       100
          c7       0.15      0.05      0.08        98
          c8       0.13      0.20      0.16        95
          c9       0.18      0.10      0.13        97

    accuracy                           0.13       973
   macro avg       0.14      0.13      0.13       973
weighted avg       0.14      0.13      0.13       973
```

At first, we thought that the reason for the accuracy score to be that low was because of the small sample size we used. So, we increased the sample size and discovered that the accuracy score got lower the larger the sample size was. That is the reason why we kept the sample size to 100 images per category.

Based on our test, we do not recommend using K-Means clustering for predictions unless more image augmentation is done to isolate the action of the drivers.

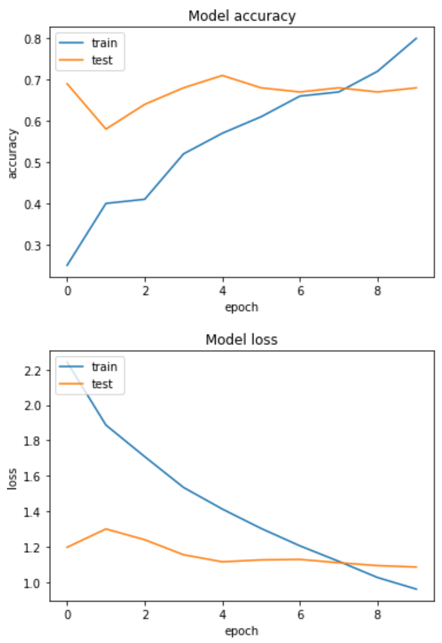
We hypothesized that a supervised learning approach would be more effective. We chose a Convolutional Neural Network (CNN) known as VGG16 because it has proven to be adept at image classification tasks with more than two classes.

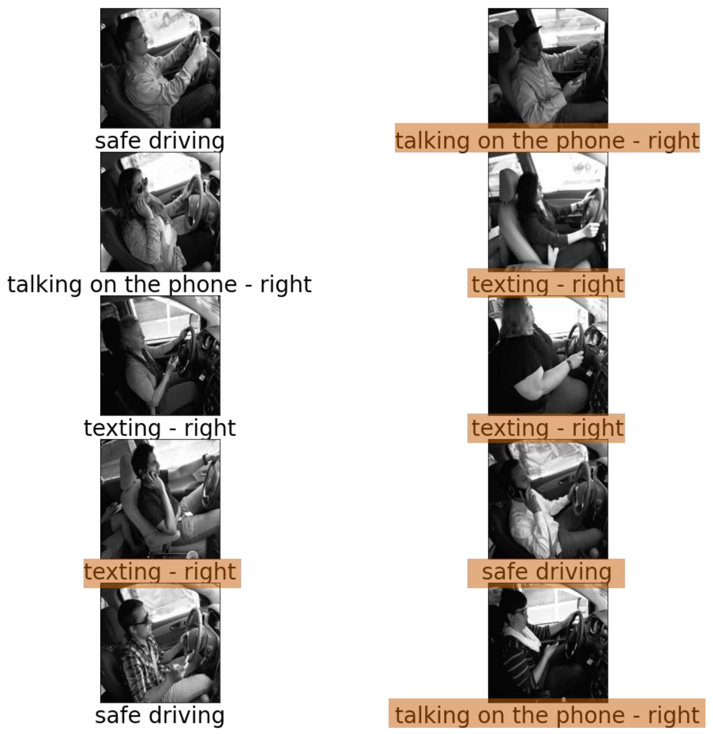


First, we ran the model with the smaller subset of data of 100 images in each class. The training took about five minutes and accuracy was around 70%. We repeated the same training using the images with the heatmapping and achieved a higher accuracy of around 80%. However, it is known that the model was overfit and was unable to predict well on the test images.

Overfit ~70%

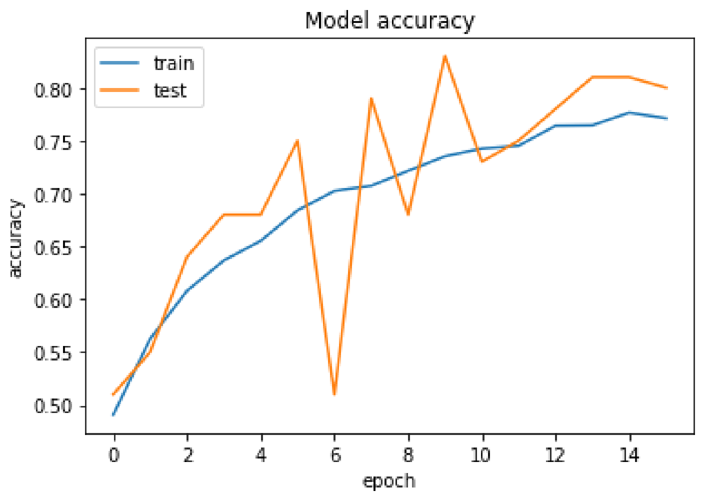
Test result - **40% accurate**

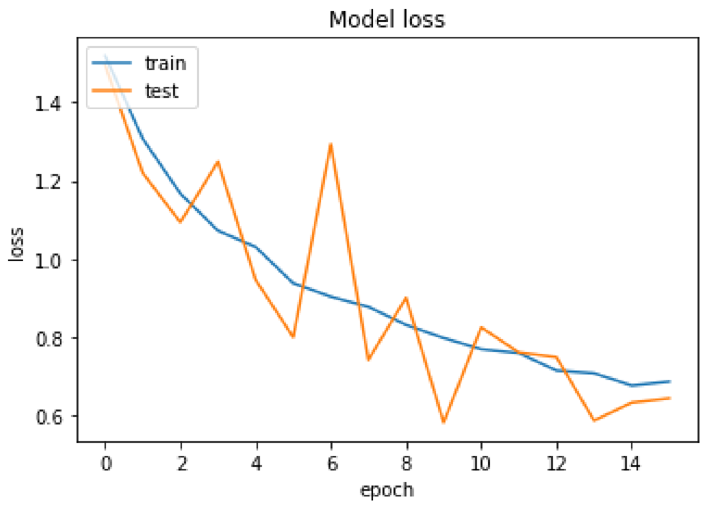




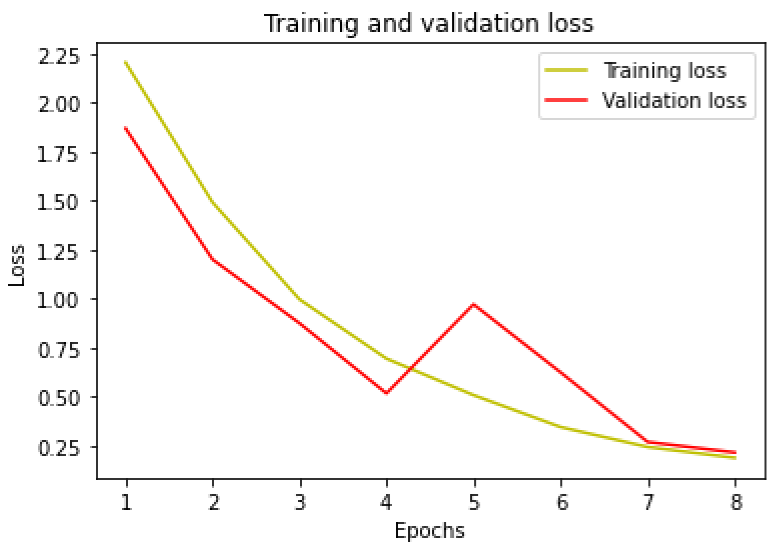
Then, we repeated the training with the full set of training data. It took significantly longer to train, around an hour and a half even with Google CoLab runtime set to GPU.

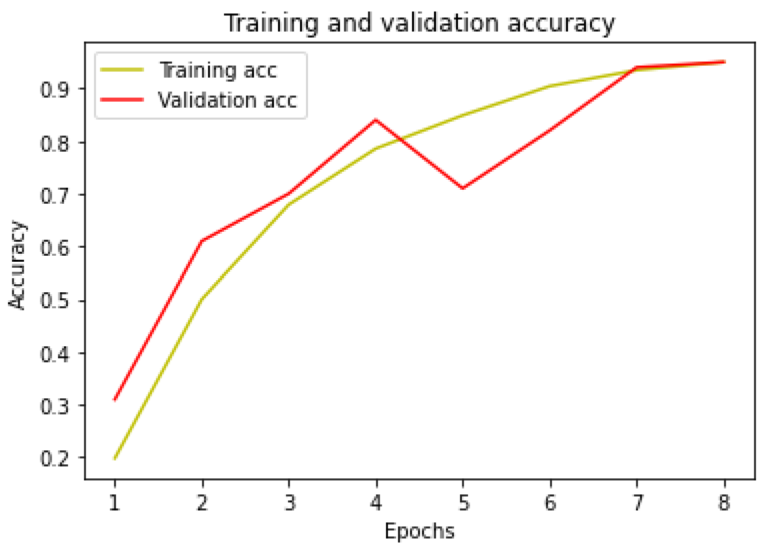
However, the accuracy was higher (~80%) and the loss (0.64) was much lower than the previous trained model using only the subset of images.





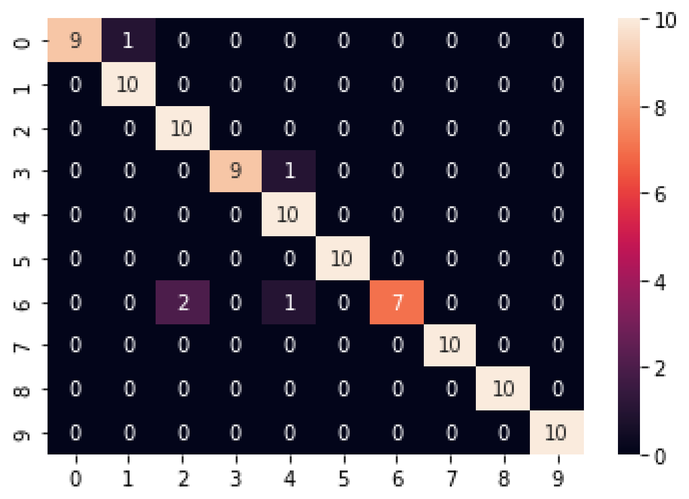
A second CNN model was created to test accuracy using a 'rmsprop' optimizer and 'sigmoid' activation method. From the training and validations loss and accuracy charts we can determine the optimal epoch runs is at 4. Each epoch takes an average of 560 seconds (about 9 and a half minutes) to iterate through.





Next, a confusion matrix was created to visualize accuracy of trained labels across the x axis and sampled predicted labels across the y axis. 96 of the 100 images was correctly predicted.

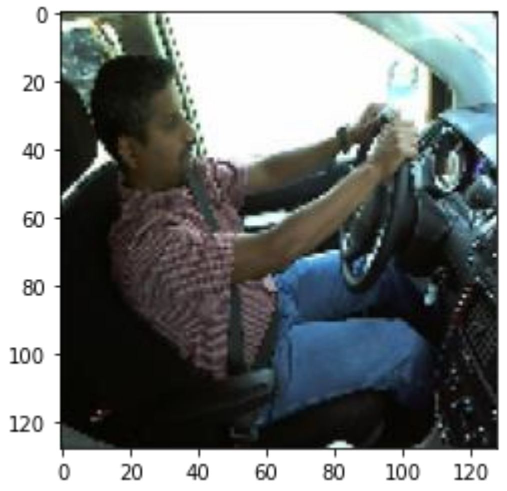
CNN Confusion Matrix



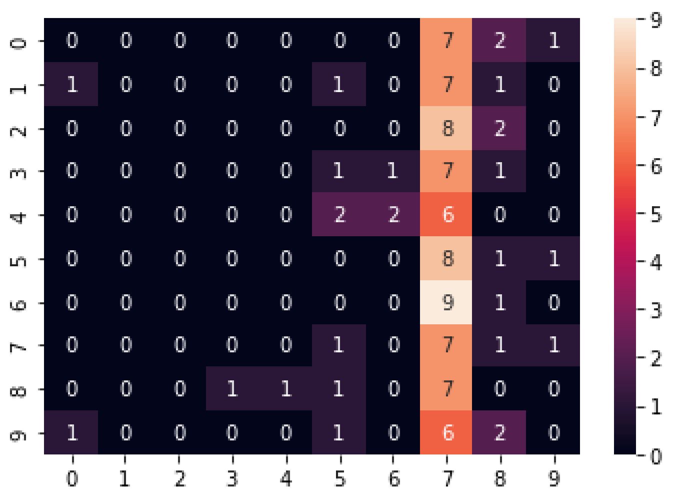
	0	1	2	3	4	5	6	7	8	9
0	9	1	0	0	0	0	0	0	0	0
1	0	10	0	0	0	0	0	0	0	0
2	0	0	10	0	0	0	0	0	0	0
3	0	0	0	9	1	0	0	0	0	0
4	0	0	0	0	10	0	0	0	0	0
5	0	0	0	0	0	10	0	0	0	0
6	0	0	2	0	1	0	7	0	0	0
7	0	0	0	0	0	0	0	10	0	0
8	0	0	0	0	0	0	0	0	10	0
9	0	0	0	0	0	0	0	0	0	10

```
We further analyzed the actual images which showed the predicted labels against
the actual trained labels.
```

```
The prediction for this image is:  c0
The actual label for this image is:  c0
```

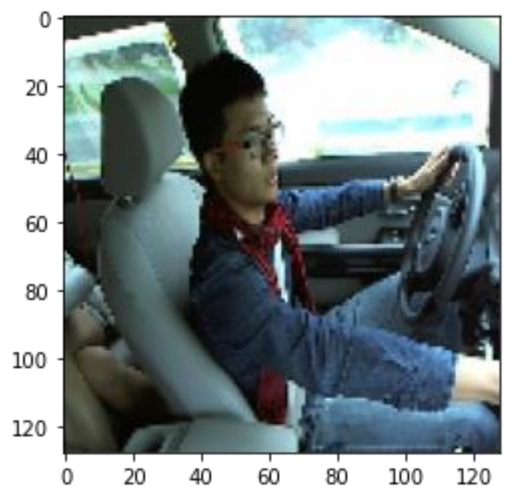


Our last model used for analysis was a random forest. Keeping the same prediction images, optimizer, and activation as the last CNN model we analyzed the results again in a confusion matrix. The random forest attempted to bin the images in the training label 7. Here we can see that it is less accurate with 72 of the 100 images being correctly predicted.



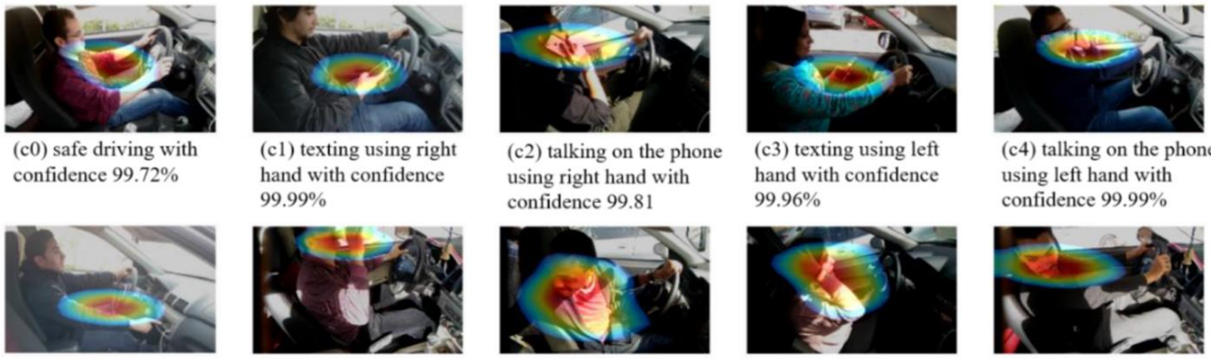
Again, we randomly sampled the index and were able to show the images along with their prediction and actual rained labels.

```
The prediction for this image is:  c5
The actual label for this image is:  c5
```



Recommendation:

We recommend that State Farm and other auto insurance companies invest in Gradient Class Activation Mapping for more accurate results as shown here:



(c0) safe driving with confidence 99.72%
(c1) texting using right hand with confidence 99.99%
(c2) talking on the phone using right hand with confidence 99.81
(c3) texting using left hand with confidence 99.96%
(c4) talking on the phone using left hand with confidence 99.99%
(c5) operating the radio with confidence 99.94%
(c6) drinking with confidence 99.63%
(c7) reaching behind with confidence 100.00%
(c8) hair and makeup with confidence 99.99%
(c9) talking to passenger with confidence 99.99%

The CNN model will help with identifying the most important parts of the image that identify the type of distraction present. It will be essential to prepare the model to encounter the driver in different lighting conditions and from different angles. One way to achieve this training is with NVIDIA Omniverse where the model could be set up inside a 3D simulation within a changing environment. That way different situations can be presented without needing to stage the photos. We hope that this level of detail will help to identify when drivers have low attention and get them back on track.

References

Bhattiprolu, S. (2020, September 14). *Python_for_microscopists/158_classification_cnn_rf.py at master · bnsreenu/python_for_microscopists*. GitHub. Retrieved June 26, 2022, from https://github.com/bnsreenu/python_for_microscopists/blob/master/158_classification_CNN_RF.py

Chetoui, Mohamed. 2019. "Gradient-Weighted Class Activation Mapping - Grad-CAM-." Medium. Medium. March 14, 2019. https://medium.com/@mohamedchetoui/grad-cam-gradient-weighted-class-activation-mapping-ffd72742243a.

"State Farm Distracted Driver Detection | Kaggle." *Kaggle.com*, 2016, www.kaggle.com/competitions/state-farm-distracted-driver-detection/data.

Selvaraju, Ramprasaath R., Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2020. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization." *International Journal of Computer Vision* 128 (2): 336–59. https://doi.org/10.1007/s11263-019-01228-7.

Team, Keras. 2020. "Keras Documentation: Grad-CAM Class Activation Visualization." Keras.io. 2020. https://keras.io/examples/vision/grad_cam/.

“VGG-16 | CNN Model - GeeksforGeeks.” 2020. GeeksforGeeks. February 26, 2020. https://www.geeksforgeeks.org/vgg-16-cnn-model/.

“Your First Deep Learning Project in Python with Keras Step-By-Step.” Machine Learning Mastery. June 17, 2022. https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/.