# Machine Learning Engineer Nanodegree

## Capstone Report

Michael Stevens

Octover 21th, 2018

## Problem Definition

### Proposal

Additional material for this project is available here, including the proposal.

https://github.com/mstevenscmu/capstone-project

The original proposal review is here: https://review.udacity.com/#!/reviews/1437790

### Project Overview

The StateFarm Kaggle Competition represents an important application of machine learning classification to image data. Distracted driving is a large contributor to property damage and personal injury. By adding a camera facing the operator of a motorized vehicle, it may be possible to recognize when the operator is distracted. This could provide an opportunity to add a safety system similar to the audible chime when the seatbelts are not in use. The identification of a distracted operator could be used to remind the operator, alert other drivers with a signal, alter insurance billing, or be used as an input into a self-driving auto-brake system.

This problem appears to be feasible to solve using the skills taught in the Udacity MLND. This problem has not only been the subject of the Kaggle competition, but also the subject of academic projects. Other research has shown the ability of deep neural networks to perform pose estimation on subjects. These projects show that this task is possible to learn using machine learning.

- Kagggle StateFarm Distracted Driver Detection
    - https://www.kaggle.com/c/state-farm-distracted-driver-detection
- End-to-End Deep Learning for Driver Distraction Recognition (Koesdwiady)
    - https://www.springer.com/cda/content/document/cda_downloaddocument/9783319598758-c2.pdf?SGWID=0-0-45-1608335-p180889205
- DarNet: A Deep Learning Solution for Distracted Driving Detection
    - https://users.cs.duke.edu/~cdstreif/static/darnet_presentation.pdf
- DeepPose: Human Pose Estimation via Deep Neural Networks (Toshev)

- https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42237.pdf

## Problem Statement

The problem is to classify images of automobile operators into one of ten possible classes. One of the classes represents normal driving and the other nine represent distracted driving. The classification algorithm to solve this problem will output probabilities for each possible class. The performance of the classification will be measured using accuracy and multi-class logarithmic loss.

This classification problem has several distracted classes that should be possible to classify with neural network deep learning systems, such as using a cell phone or reaching into the back seat. With a vast amount of training data it would be possible to train a network from scratch to classify novel images into the ten classes. With less labeled training data, it will require using transfer learning on bottleneck features of a pretrained neural network.

The full list of classes is the following.

- 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

## Metrics

The two metrics that will be used for this project are accuracy and multi-class log loss. Accuracy is an appropriate metric for understanding performance because the data set does not contain skewed classes. The multi-class log loss is a more sophisticated examination of the results and provides insights into how confident the predictions of class are.

This multi-class log loss metric is a standard for multi-class classifiers when the output is a predicted probability for each class. This is what the Kaggle competition originally used and is definitely suitable. Log loss takes into account the certainty result for the prediction. A standard accuracy measurement only accounts if the top prediction matched the actual label. In the case of a 10-class classification problem the actual class could be predicted with 0.1 + epsilon certainty or 1.0 certainty, but both would be counted the same by an accuracy score. Log-loss would rate the 1.0 certainty prediction much better than the 0.1 + epsilon prediction. A correct prediction of class A with 1.0 probability will score better on log loss than a prediction of A with 0.5, B with 0.25 and C with 0.25.

Log loss heavily penalizes mispredictions with a low predicted probability with extremely high loss values. Correct predictions with absolute certainty score close to zero. The undefined values of the

formula for zero and one certainties are compensated for by applying a minimum and maximum to ensure the p values are within (0+epsilon, 1-epsilon).

- http://wiki.fast.ai/index.php/Log_Loss
- http://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html#sklearn.metrics.log_loss
- http://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html

Consequences of misclassification depend on how the classifier would be used. If the system was used to sound a warning chime or apply brakes, users might be more accepting of a bias more towards classifying the image as safe driving to avoid unwanted chimes or braking. If the system was used as a prescreen for humans reviewing videos of driver behavior, such as monitoring a school bus driver, it might be preferable to allow more false positives. A human could review the video and determine if it was really distracted driving.

# Analysis

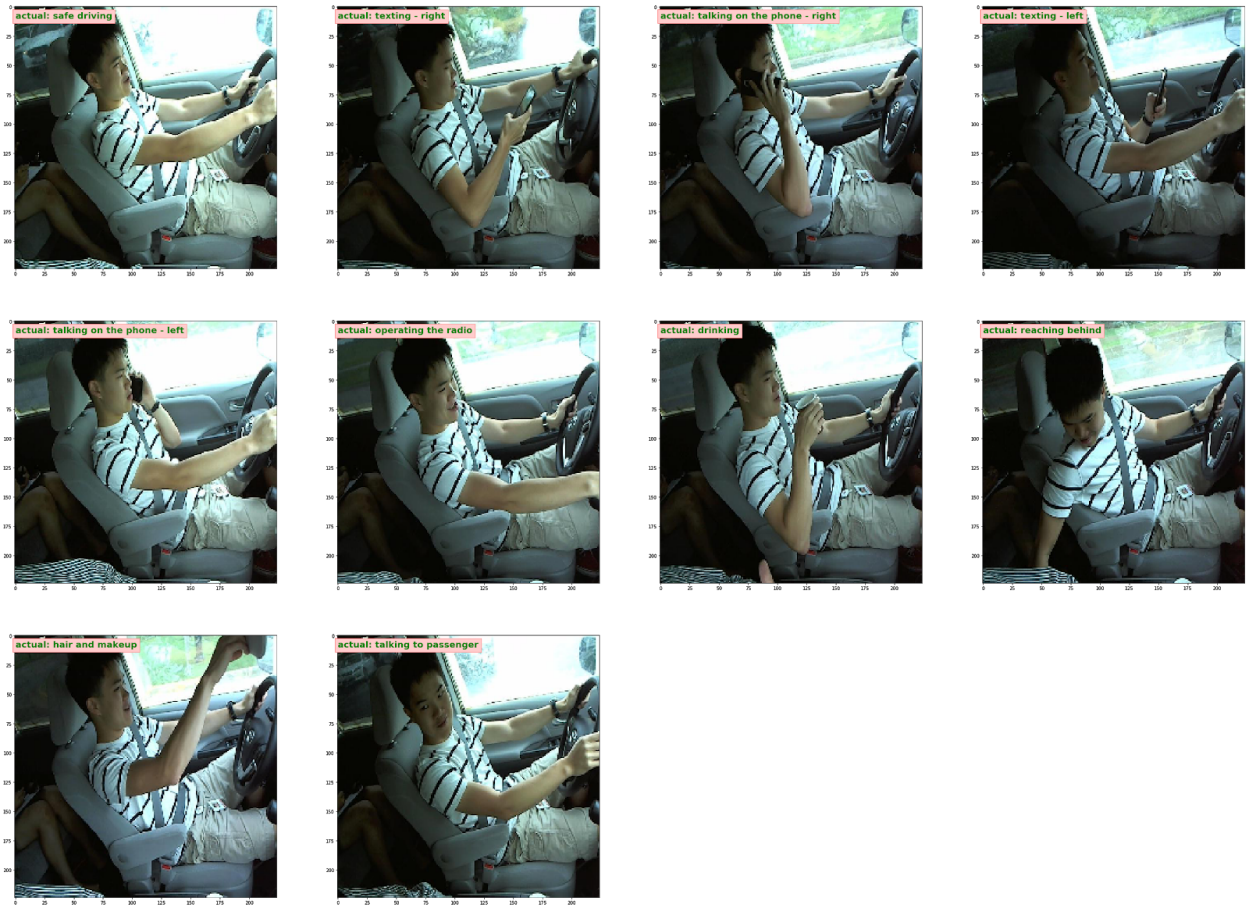
## Data Exploration

The dataset for this problem was acquired by StateFarm. The drivers captured represent a mix of genders, clothing, eyewear, ethnicity, and weights. Several vehicles are represented in the training dataset. A number of the images show the drivers wearing StateFarm badges, so they may have used their employees for a majority of their captures. The subjects captured were not actually operating the vehicle because the car was being towed behind a truck. In some images, a person with a clipboard is visible in the back seat. Each labeled class has between 1911 and 2489 images.

https://www.kaggle.com/c/state-farm-distracted-driver-detection/data

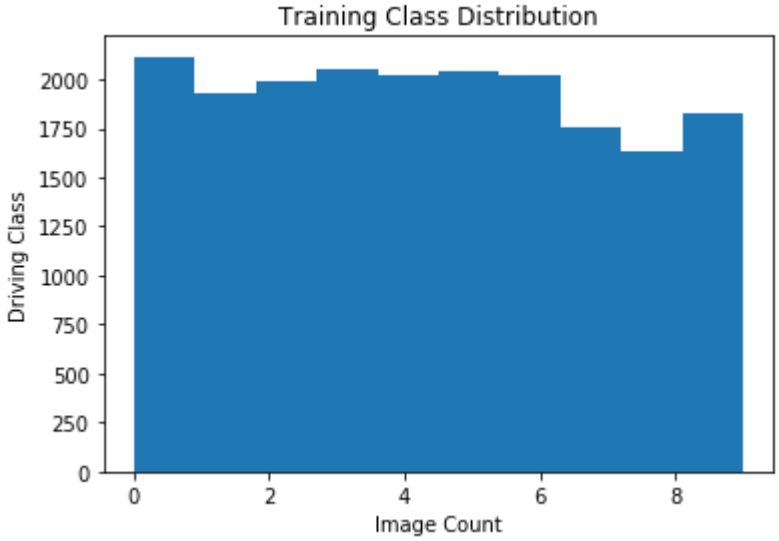
The training dataset is organized into ten folders, with one representing each class. Each image is a 640x480 pixel jpeg captured from a location near the passenger side A-pillar. A CSV file provides a driver identifier, class, and image name for each image in the training folder. A separate folder called test has 79,726 images, but no labels are provided as a part of the download, so they will not be used. Instead, the images in the test folder will be divided into train, validate, and test sets by splitting on the driver id.

Example images for each of the driving classes are shown here. Some clear potential challenges are already visible, such as very little obvious difference between operating the radio and normal driving. Hair/makeup and talking to passenger also have potential to be confused with checking in a mirror.




The 22,424 labeled images have a driver ID and driving behavioral class for time image. For this project, the labeled training data was split into train, validate, and test sets. Since the images represent captures from video, there are images that are very similar and would represent leakage across sets. To avoid this leakage, the sets are split by driver so that all images for a driver are only a member of one of the train, validate, or test classes. A total of 26 drivers are present in the dataset. 22 were used for training, 1 for validation, and 3 for test. This resulted in 19407 training images, 1034 validation images, and 1983 test images.

The distribution of labeled training data is evenly distributed across all of the driving classes. There is no presence of skewed classes in the labeled data, but a real world application would expect to see far more examples of normal driving than distracted driving.




a

Augmentation is frequently used to increase the amount of training data when building deep learning classifiers. Augmentation techniques of flipping and rotating images are very useful for training a classifier to recognize a cat in many possible orientations when only a single orientation is present. For the StateFarm competition the camera is always positioned in a fixed position, so flipped or rotated training images will not provide much value in training the network. Other augmentations such as lighting changes or non-affine transformations to simulate other camera lens may be useful.

## Algorithms and Techniques

Transfer learning using a pretrained Keras network was used for the project. My experience with using Resnet50 in prior MLND projects has shown that it has potential to perform well in multi-class image classification problems. The pretrained model was configured in Keras to not be trainable and additional layers appended to it that were trainable.

During training, additional callbacks were provided to the fit function to checkpoint the model at the best point and to stop early if no progress was being made in training.

The training was done for up to 20 epochs at a batch size of 20. Plots of the accuracy and log loss after each epoch were generated to understand the impact of additional epochs of training.

Predictions were made by selecting the class with the highest probability. This was necessary for accuracy computations. Log loss was computed using the raw probabilities predicted for each class.

## Benchmark

A benchmark model for distracted driving would be to always predict normal undistracted driving with complete certainty, 1.0, and all distracted driving classes as 0.0. The predictions will then be measured with accuracy and multi-class logarithmic loss. This benchmark should achieve slightly better than 10% accuracy, but a poor log-loss score. Beating this benchmark's log loss is a bare

minimum. The Kaggle competition also provides other benchmarks using the leaderboard with the log loss of the winning entries being lower than 0.10.

# Methodology

## Data Preprocessing

Several steps were required to prepare the data for usage with a transfer learning model. The 640x480 images were resized to the input dimensions of the pretrained neural networks. The RGB 0->255 values was also converted to a floating point format compatible with the neural network's input expectations.

The data was also divide into train, validate, and test sets by driver identifier to prevent leakage from train to test sets where a test image was just the next frame in a video of a training image. The similarity of some test images to each other because they are frames from a video could cause high accuracy and low loss for training.

A cursory scan of file sizes was done to make sure there were no outliers such as an empty image that compressed to an unusually small size.

Significant effort in preparing and cleaning the dataset was done by StateFarm to ensure suitability for the contest. In addition, when images were displayed for examination of data or printing of misclassifications, the labels were added to allow checking that the actual label was represented what was contained in the image. No errors were noticed during the runs.

## Implementation

The key steps to transfer learning with Resnet50 are the following

1) Resizing the images to match the Resnet50 input size of 224x244 from the 640x480 StateFarm dataset size
2) Create a Keras model using Resnet50 and the imagenet weights. Set all of the Resnet50 layers to be not trainable
3) Add additional Dropout, and Dense layers. The dropout layers were to reduce overfitting. The number of dense nodes was a balance between too few where the model would be high bias and too many that wouldn't be trainable with the amount of data. Too many dense layers would be too deep to train because of the vanishing gradient problem.
4) The Dense layers used ReLU activation, except for the final layer of 10 classes which used softmax.

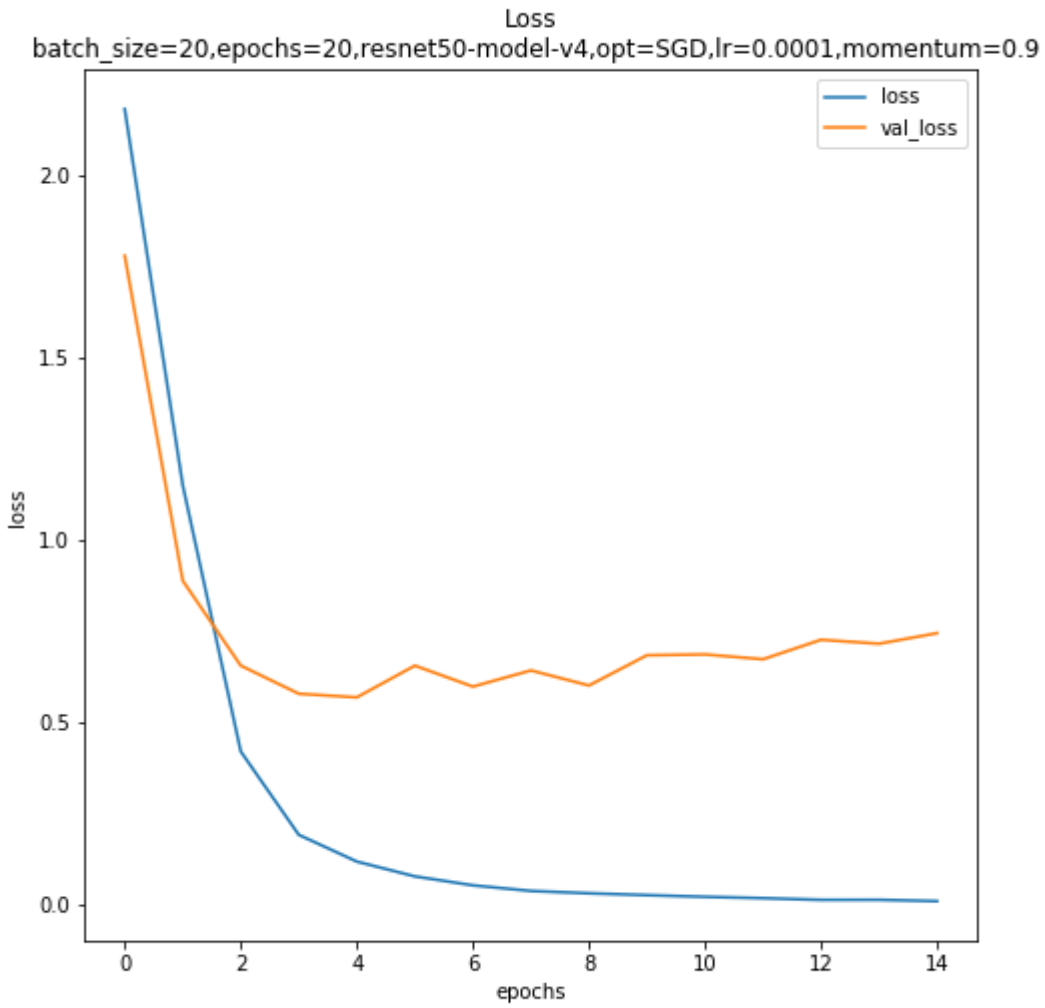
I attempted using both the SGD and Adam optimizers and obtained better results with the SGD and the parameters selected.

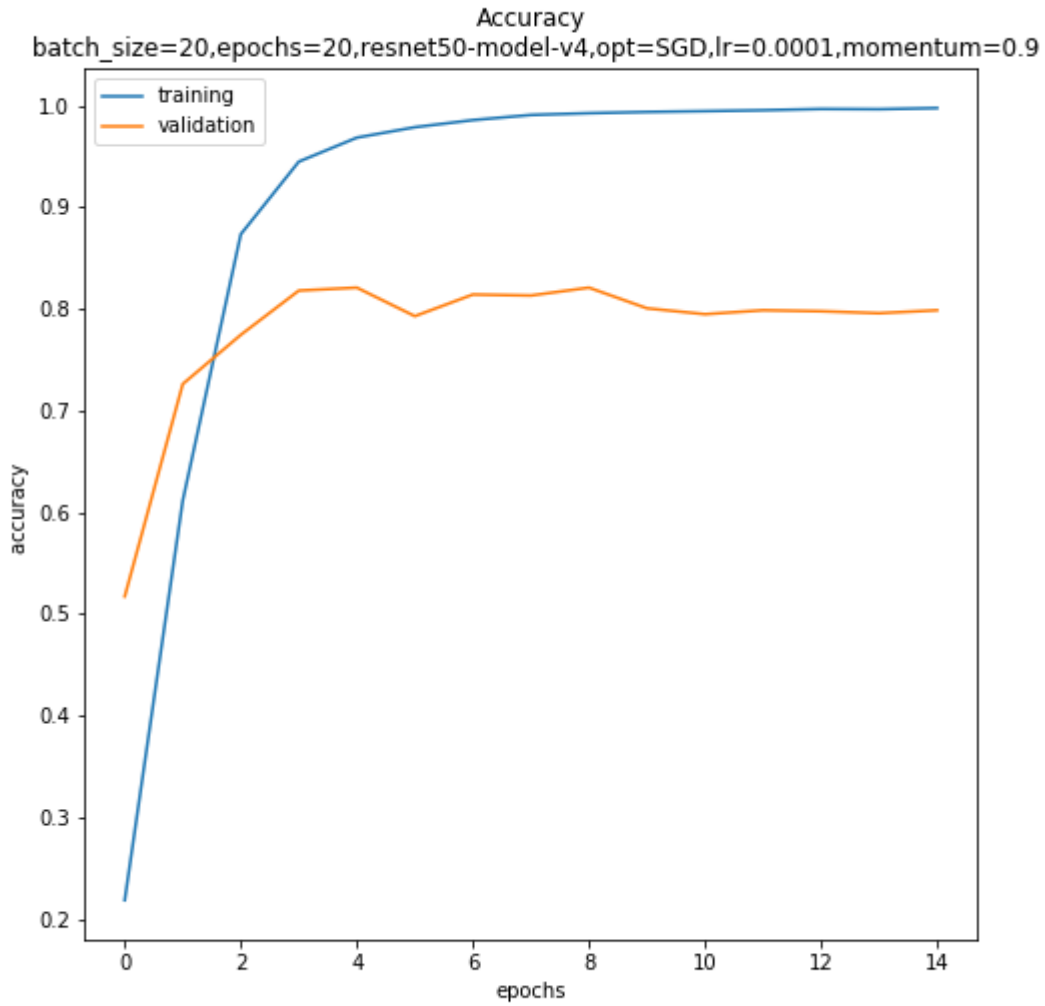
Plots of the accuracy and log_loss for train/validate were plotted per epoch and instrumental in understanding the progress of training. A confusion matrix was plotted using seaborn and was useful

for understanding patterns in misclassifications. Additionally, sample images from the set of misclassifications were displayed with the true and predicted classes.

## Refinement

The biggest implementation detail key to refinement was keeping track of the experiments in an orderly fashion. This was accomplished by having functions return configured variables for the optimizer, keras model, and the training parameters. The returned variables also had a stringified form that was used when saving the output images and added to the titles. Instead of modifying the existing functions, new functions could be created that would encapsulate the experiment in a read-only way. The results of the experiment were recorded with the description of the experiment parameters in the chart titles and the filename saved to the analysis/ directory.






The majority of the changes were to the model layered after the pretrained deep neural network. The changes to this section of the network were supported by the test/validation accuracy and log_loss graphs for training epochs. Parameters were changed one at a time to determine if they represented a positive or negative refinement to validation accuracy and log_loss.

Additional modifications to the optimizer, optimizer parameters, batch size, and maximum number of epochs was attempted. In many of the runs the early stopping callback prevented additional rounds from executing if the model wasn't improving.

Substantial improvements from the initial single dense layer, single epoch training were achieved through the modification process and guided by the graphs.

# Results

## Model Evaluation and Validation

The benchmark model achieved accuracy of 12.30% and a log loss of 30.28 with the test set. This was very close to the last-placed entrant into the StateFarm contest leaderboard, so perhaps.

The Resnet50 transfer learning model with two fully connected 1024 node layers achieved 67.62% accuracy and a log loss of 1.23 with the test set, which places it in the top 50% of leaderboard entries. https://www.kaggle.com/c/state-farm-distracted-driver-detection/leaderboard

The parameters used in this model were the following

- batch_size=50
- Optimizer=SGD with learning_rate=0.001 and momentum=0.9
- epochs=20

During training epochs, the accuracy hovered around 99% and log loss of 0.2. The validation accuracy was less stable, but was around 80% and validation log loss around 0.5. This is evidence of the model overfitting during training.

Additional epochs of training would not benefit without altering the model to reduce overfitting, such as reducing the number or depth of the dense layers or adding more dropout.

Further refinement of this model to reduce the overfitting problem was accomplished by ensuring a dropout layer between each fully-connected layer and after the output of the pretrained model. The number of nodes in each fully connected dense layer was also reduced to provide less capability for memorization. Both of these are appropriate responses to overfitting.

After refinements driven by the graphs, the best model achieved the following.

- Test accuracy: 66.3137%
- Log loss: 1.010329

This was done by having the following parameters

Test-results

- batch_size=20
- epochs=20
- Resnet50-model-v4
    - Resnet(50)
    - Flatten()
    - Dropout(0.5)
    - Dense(256) + ReLU
    - Dropout(0.5)
    - Dense(256) + ReLU

    - Dropout(0.2)
    - Dense(10) + softmax
- optimizer=SGD
    - lr=0.0001
    - momentum=0.9

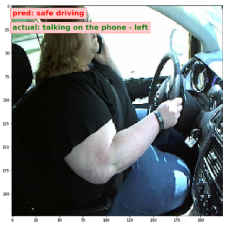
### Justification

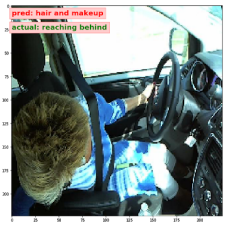
The results of the transfer learning solution at above 66% accuracy and 1.01 log loss on the test set compared to the 12.3% accuracy and 30.28 log loss of the benchmark solution clearly show an ability to classify images of distracted drivers. There is clearly room for improvement with the top Kaggle Competition entries posting a log loss of below 0.1.

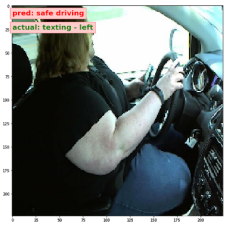
The level of accuracy and log loss required depends on the application. The level I achieved is not enough for a system integrated into a vehicle and used to alert a reminder chime. It is sufficient to show the potential of the technique and justify additional resources into improving the solution.

## Conclusion

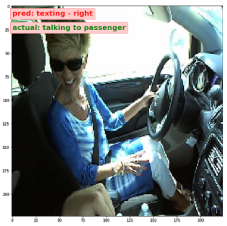
A few misclassified examples are presented in the following image with the true class and the mispredicted class. These examples were useful in understanding why the classification system was performing incorrectly by highlighting potential issues with the data or nuances with the data.

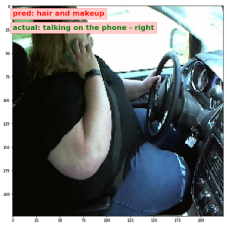

pred: safe driving
actual: talking on the phone - left


pred: hair and makeup
actual: reaching behind

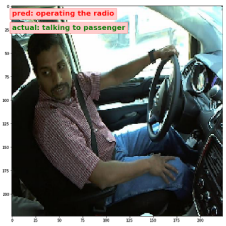

pred: safe driving
actual: texting - left


pred: drinking
actual: talking on the phone - left

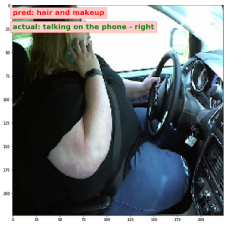

pred: texting - right
actual: talking to passenger

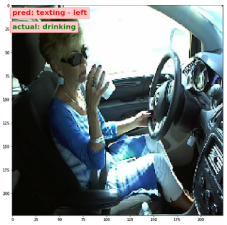

pred: hair and makeup
actual: talking on the phone - right


pred: hair and makeup
actual: talking on the phone - right


pred: operating the radio
actual: talking to passenger

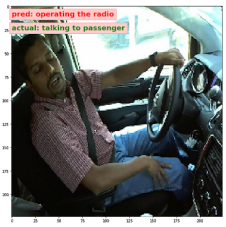

pred: drinking
actual: talking on the phone - right

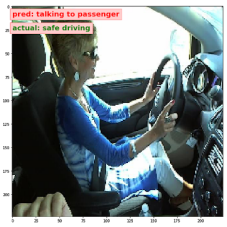

pred: hair and makeup
actual: talking on the phone - right


pred: texting - left
actual: drinking

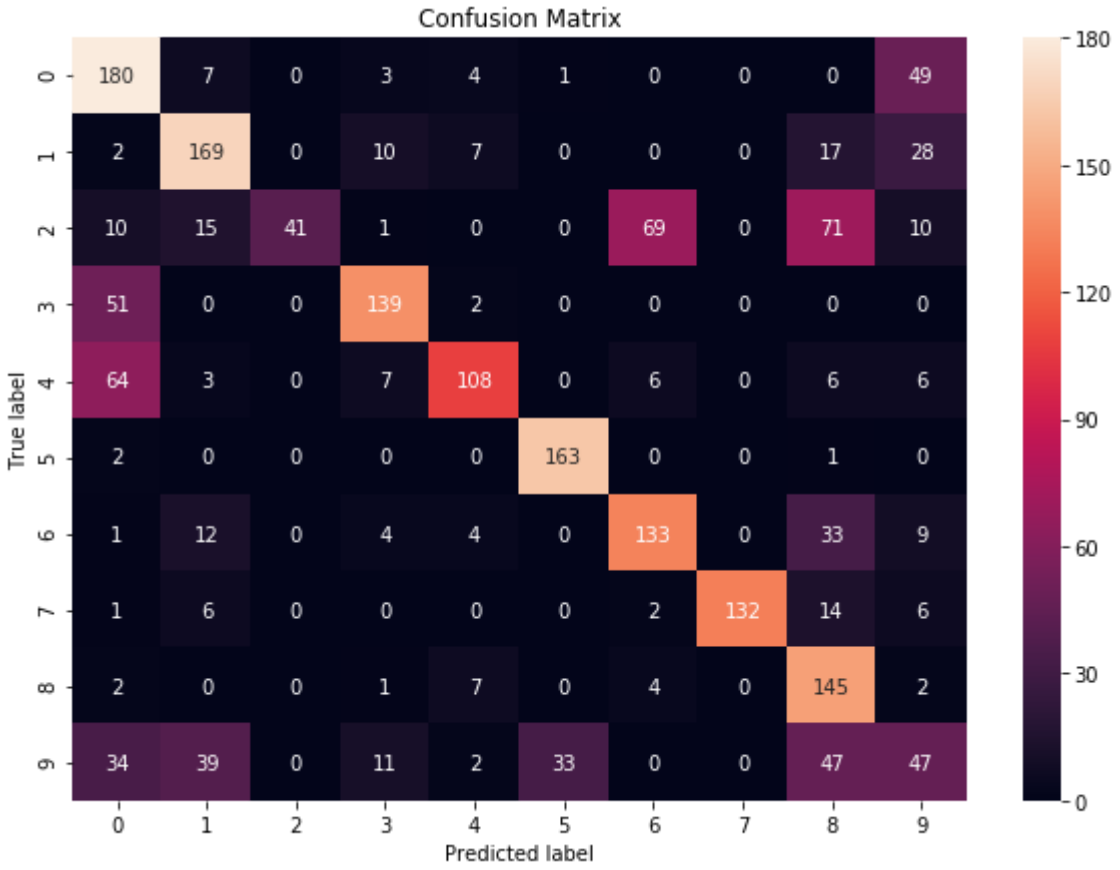

pred: safe driving
actual: talking to passenger


pred: safe driving
actual: talking on the phone - right


pred: operating the radio
actual: talking to passenger


pred: talking to passenger
actual: safe driving


pred: operating the radio
actual: talking to passenger




## Reflection

- Normal driving was misclassified frequently as talking to passenger. The talking to passenger can also easily be confused for checking the mirrors.
- Talking on the phone right was misclassified frequently as drinking and hair/makeup.
- Texting left and talking left were frequently misclassified as normal driving. The texting and talking left is hard because the camera angle doesn't show much of the phone for those two classes.
- Drinking was misclassified as hair/makeup.
- Talking to passenger was misclassified equally often as normal driving, texting right, operating radio, and hair/makeup.

One of the hardest aspects of this project was the extremely slow iteration time, even with using an Amazon EC2 GPU accelerated instance. A more effective way to work might be to generate numerous experiments that would all be executed in a batch job. Instead of paying by the hour for an active EC2 instance, it might be nice to have an extremely powerful system that jobs could be submitted, quickly returned, and charged by the time spent in the job. Paying for machine time while coding and reviewing results incentives selection of a less powerful system to avoid excessive charges while not actively training.

## Improvement

A very obvious way to improve the results would be to have an even wider set of training data captured and labeled. If a driver and capture assistant were paid 15 per hour each, it might be possible to capture more subjects at a cost starting at $30 per subject. It is possible this dataset could be doubled in size for a thousand dollars.

K-fold cross validation could be used to get the most out of the training data by not losing some of it to a validation set. This could be implemented by using the driver identifier as the split for the folds in the same way the train, validate, and test sets were created.

Although rotation and flipping augmentation would not be useful, there are possible augmentations that may provide advantages. Blurring, lighten/darken, translation, and masking out everything but the segmented subject may be useful. The lighten/darken could compensate for an uncontrolled free variable of lighting conditions in the source data. Translation may be useful to handle different seating or camera positions. Masking a segmented subject has the potential to eliminate many unrelated signals from the windows, visor, wheel position, visor, mirrors, and vehicle contents. Implementation of non-affine transformations to model different camera intrinsic parameters would be useful if the model needed to handle different camera intrinsic parameters from multiple camera vendors or lenses.

Ensemble methods are a proven way to create better predictions. Creating an ensemble from the output of several models using different pretrained networks would likely increase the accuracy and lower the log_loss.

It is also possible to unfreeze training of the pretrained layers after the trailing dense layers had been initially trained. Additional epochs of training could then allow back-propagation into the pretrained network to customize it for the distracted driving dataset.

A much more complex change would be altering the problem from identifying distracted driving from a single image into using a sequence of images of the video. The additional information could allow the model to make more accurate predictions using multiple images that represent the same activity.