Machine Learning Capstone Project

Definition

Project Overview

Centers for Disease Control and Prevention (CDC) motor vehicle safety division published a report that one in five car accidents is caused by a distracted driver that translates to 425,000 people injured and 3,000 people killed by distracted driving every year[1]. Any visual or manual activity that diverts the driver attention such as eating or drinking, turning car knobs, adjusting GPS, looking for items in car floor, doing makeup etc. are considered as distracted driving. Distracted driving is an ongoing problem that is getting worse day by day. A potential solution of crash due to the issue of distraction occurring within the vehicle is the use of machine learning computer vision models to determine and classify distracted driver behavior that could potentially avoid crashes caused by distraction. Murtadha et al applied the machine learning approach to detect such behavior as mentioned in the problem[2]. In the Capstone Project work, I aim to determine distracted driver behavior, which leads to prevent potential accident and save life. The result of the model will help to reduce statistics of the specified problem.

Problem Statement

From a set of 2D images captured from dashboard camera[2], the aim of this project is to classify and predict driver behavior if they are driving attentively or engaged in distracted behavior.

The problem that aimed to solve here is a multi-class classification problem. The job here is properly predict and classify driver's behavior given the dashboard images of people doing 10 different actions, 9 of which are considered actions of distracted behavior. The 10 classes are as follows: 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.

To solve this problem, first I will start with importing the training data. Then the training dataset will be splitted for training and validation purposes. Images will be rescaled to keep in a manageable size. Then a Convolutional Neural Network will be created in Keras and trained with training data set. Then, it will be tested using subset of test data. Final step will be predicting the test data and performance evaluation. Here, my plan is to use different models like ResNet50, VGG19 and combination of VGG19 and ResNet50 instead of VGG16 which I proposed before for transfer learning. I will keep repeating the process by adjusting model and parameters until the desired result comes.

Metrics

Solution and benchmark models will be evaluated using the multi-class logarithmic loss. Each image has been labeled with one true class. For each image, a set of predicted probabilities is computed. The formula here is:

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

where N is the number of images in test set, M is the number of image class labels, log is the natural algorithm, y_{ij} is 1 if observation belongs to class j or 0 if otherwise and p_{ij} is the predicted probability that observation I belongs to class j.

Analysis

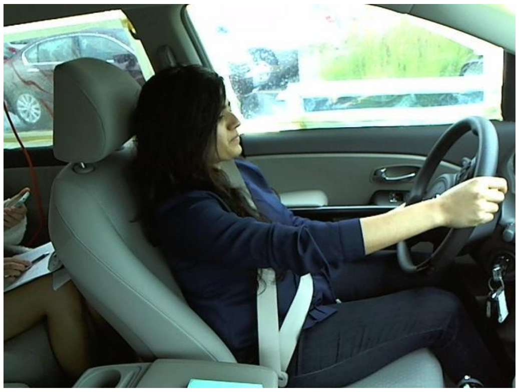
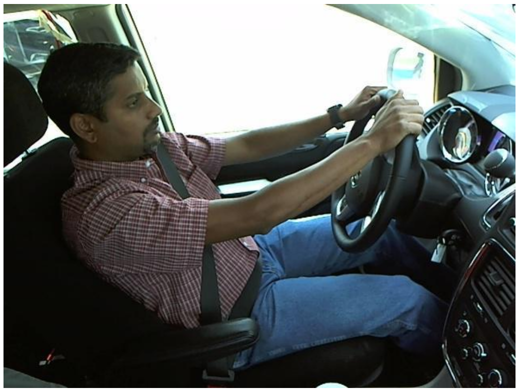
Data Exploration

There is a total of 22424 images in the train dataset and 79726 images in the test dataset. All images are colored and 640 x 480 pixels in size. The images capture the driver from a side-view dashboard camera. To maintain the balance among the classes, number of images in each class is maintained as similar. Each image classify as:

c0: Safe Driving (2489)
c1: Texting – right (2267)
c2: talking on the phone – right (2317)
c3: texting – left (2346)
c4: talking on the phone – left (2326)
c5: operating the radio (2312)
c6: drinking (2325)
c7: reaching behind (2002)
c8: hair and makeup (1911)
c9: talking to passenger (2129)

Exploratory Visualization

The dataset contains image of 10 different classes and images of each class are different from each other. Following is an example of class c0 that represents safe driving behavior:

Here we can see the difference between images from same class due to change in environment, light, camera angle, placement of hand in steering wheel, adjustment of seat etc. It would be helpful to train the model, as generalization is expected to be better.

Algorithms and Techniques

CNNs are known for their extraordinary potential of solving image classification problems. The reason it is so powerful is because CNN's uses network of neurons to learn features and patterns similarly to how the brain works.

This is a more in-depth look at Neural networks structure

- Convolutional Layers – This is the layer responsible for the learning of features. What this layer does is extract features from images by doing matrix calculations. Initial layers extract the more common features such as edges and color. The deeper the network gets(more convolutional layers), the more complex features are learnt (features that are not entirely obvious). The size of the output matrix depends on the set filter size as well as the stride. Stride is what decides how many pixels to skip before it makes the next set of matrix calculations. Every filter has a set of weights that are computed on to the input layer, these weights provide a measure for how a set of pixels closely resembles a feature. Some features are better understood by the model when they are not too compressed while other features would need to be compressed further to learn new features.

- Pooling Layers – These are the layers responsible for reducing the dimensionality of images. Reducing the dimensionality means we remove excess parameters to make images easier to interpret for the next layer. This is normally done using max pooling, which selects the highest value of the matrix based on the filters and stride. And as such, these layers tend to give a more pixelized version of an image, but these pixels are better interpreted by the machine. It is like blurring out the haystack, in order to make it easier to find the needle.

- Normalization Layers – These are layers used to counteract the change of direction and distribution affecting our dataset each time a new layer is added.

The normalization is done to realign the model input to what it is was originally intended to do – learn new features. Not applying a normalization may affect our model to only learn some features while ignoring other features, this is especially true when new data is being fed. An analogy of the problem would be; studying one lesson so much you end up forgetting to study other areas for the quiz. So, the solution is reminding yourself the end goal and reassess 'what' you need to study.

- Fully connected Layers- These are layers that convert complex matrices into more manageable outputs. With this access, we can control the number of outputs we need based on our problem by squishing, transforming and combining features into classifications. Without these layers, the model would not understand 'how many' or 'what' predictions we actually want or need.

The main model architecture that will be used is a Keras Application Model. These architectures have been perfected to classify images. Every layer and parameter in these architectures were professionally selected. Keras also has the option of making use of pretrained weights based of **ImageNet**. Making use of these pretrained weights can drastically reduce the time it will take to train and optimize the model.

The algorithms and techniques that I will try are VGG19, ResNet50 and combination of VGG19 with ResNet50.

Benchmark

The data will be trained and tested using VGG16 model. The model with the public leaderboard score will be used as well for benchmarking model (https://www.kaggle.com/c/state-farm-distracted-driver-detection/leaderboard). The aim is to be in top 50% of the public leaderboard submission (≤0.25). The target result is based on the log loss value of the predicted labels against the actual labels of the 79726 test images.

Methodology

Data Preprocessing

Following are the data preprocessing steps applied on the data:

1. Images are converted into 3 channels, RGB – This preprocessing is done so models may use the 3 channels to learn features with the objective of improving accuracy and log loss.

2. Images are resized to 224 x 224 – Resizing images makes it easier to load on memory at the cost of losing some details.

3. Image labels are converted to categorical integer features/vectors – This is done using the one-hot scheme. This encoding is needed for feeding categorical data to our models because it is the most practical way for models to read categorical data.

4. The list of Images is shuffled– Randomizing images is simply done to change the default order.

5. The list of images are divided into a train set and validation set - This division is important so that we could validate if our model is improving or not when it is training.

6. Pixel values are converted to 32-bit floats – This is done so we could rescale our images.

7. To normalize the data, pixel values are divided by 255 as it is the maximum RGB value.

Implementation

The implementation steps are as follows:

1. Two functions were created to read the images, one function for reading the train dataset and another function for reading the test dataset. Reading data was a time consuming process.

2. Split train dataset into train and validation subsets: train_test_split function from sklearn was used to split the dataset.

4. Built architecture without using any Keras Application Models.

5. Used Keras fit function to train the model.

6. Used the validation set to test the model locally.

Here, I used K fold validation technique and compare the performance using ResNet50, VGG19 and combining these two models together. Importing and training data took a lot of time as first I tried without using GPU. The runtime was unexpectedly long. After using GPU, the performance was a lot faster which I found quite amazing. Due to time constraint, I could not use more that 5 epoches. Details of the implementation steps are described in result section. I would like to mention that, I have gone through other github projects and Udacity's previous projects and taken the necessary help from other's work, which help me to understand and implement better.

Refinement

After using ResNet50 model, the logloss found was 0.4657. The performance improved in VGG16 model where the logloss was 0.3592. Finally, the combination gives us a logloss of 0.3156 whch outperforms both previous models.

Results

Model Evaluation and Validation

The final model architecture and parameters were selected based on the validation loss and test results. Multiple tests and refinements had to be done on the model and parameters before I could achieve optimal results.

The model architecture and parameters are as follows:

1. The model input uses the default shape of our keras application model. The shape is 224, 224, 3.

2. The Keras application model I used and selected is ResNet50 and VGG19.

3. This decision was entirely based on the validation results.

4. Made use of the ‘ImageNet’ (pre-trained) weights for the Keras application model because the random initializations option did not provide results anywhere near that of ‘ImageNet’.

Also using pre-trained weights option is known to yield decent results when working on similar classification problems without needing to spend too much time training the model.

5. The output of the Keras application model is then applied a Global Max Pooling 2D layer because it used to reduce the dimensionality of the data and partly reduce overfitting. Using a Global Average Pooling 2d does only slightly worse and not using either would significantly (negatively) affect the end results.

6. Applied a dense layer with 512 units. This is done to change the dimensionality of the vectors and is basically used to filter out features. Tried with several other units and combination of layers (dropout and other dense layers) but none yield better results

7. The dense layer makes use of the ‘relu’ activation. This activation was recommended in the Udacity course because training is faster and overall gives the best results. I tried all other available activations like ‘selu’ and ‘softplus’. Some did at times provide better test results, but the results differed significantly from one test to the other despite setting a random seed.

8. The next layer added is a Batch Normalization layer. This layer is used to normalize the data. This is a technique helps train the network faster and improve the results.

9. The last layer is a dense layer with 10 units. The units should match the number of classification we need. This is common practice in CNN's.

10. A 'softmax' activation is used in the last dense layer because it yields the best results for a multi classification problem. I tried and tested with other activation functions, but no activation does better.

11. The model architecture is then compiled with the optimizer 'SGD'. This optimizer was selected after looking through all available optimizers.

12. The 'SGD' optimizer has the learning rate of 0.001. At this speed the optimizer learns important features rather quickly. If we were not to apply Batch Normalization, we would need to reduce the learning rate to 0.0009.

13. Applying a Momentum of 0.9 to our optimizer simply aims to speed up the learning process.

Justification

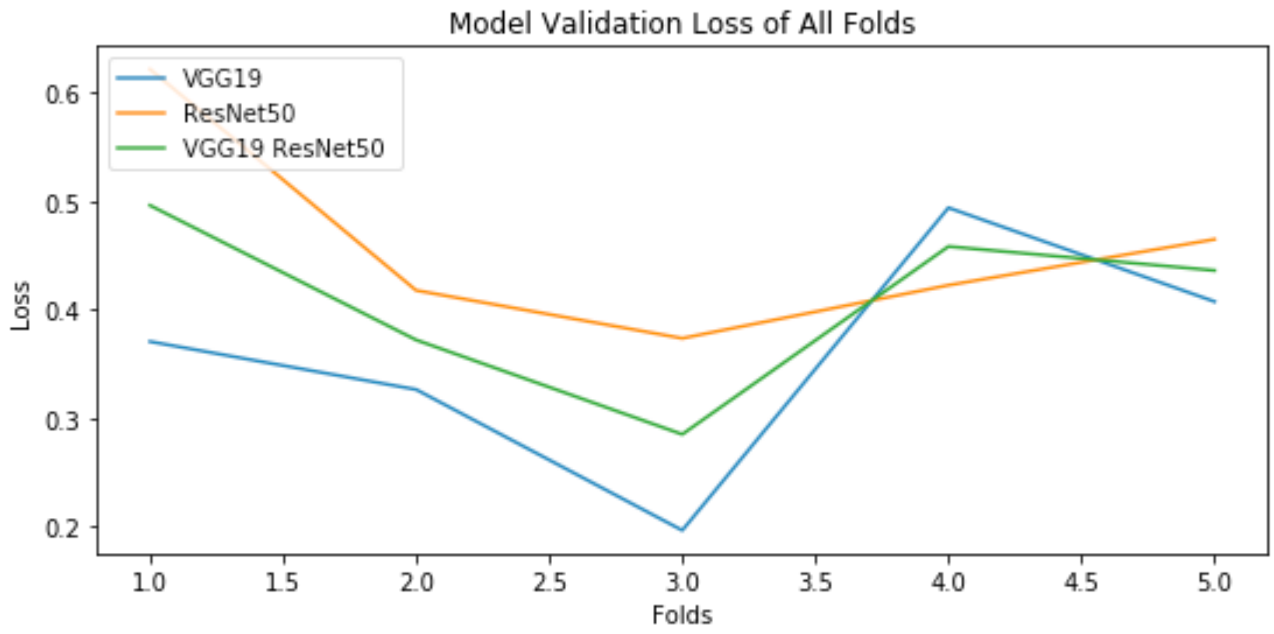
The Kaggle results show that the final solution achieved a public log loss of 0.08739. The models that I tried performed moderate level and the logloss is 0.3156. As you can see, my solution does reach significantly close to the benchmark and I ended up not submitting in Kaggle. At least it was good enough to solve the problem of categorizing drivers into the 10 behaviors with a smaller error margin that I am satisfied with.

Conclusion

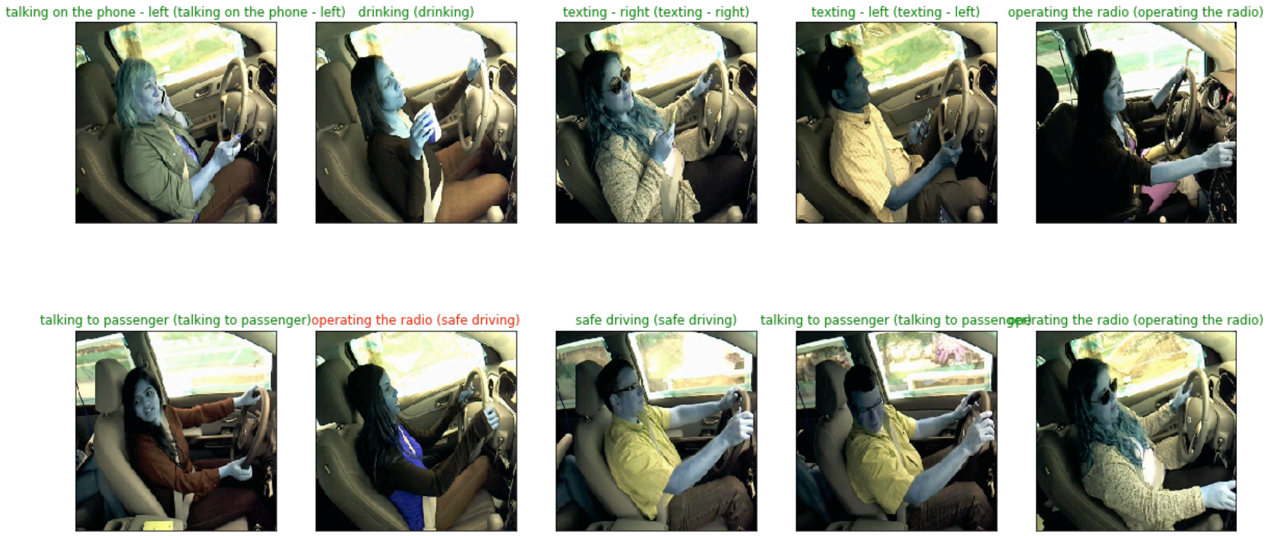
Free Form Visualization

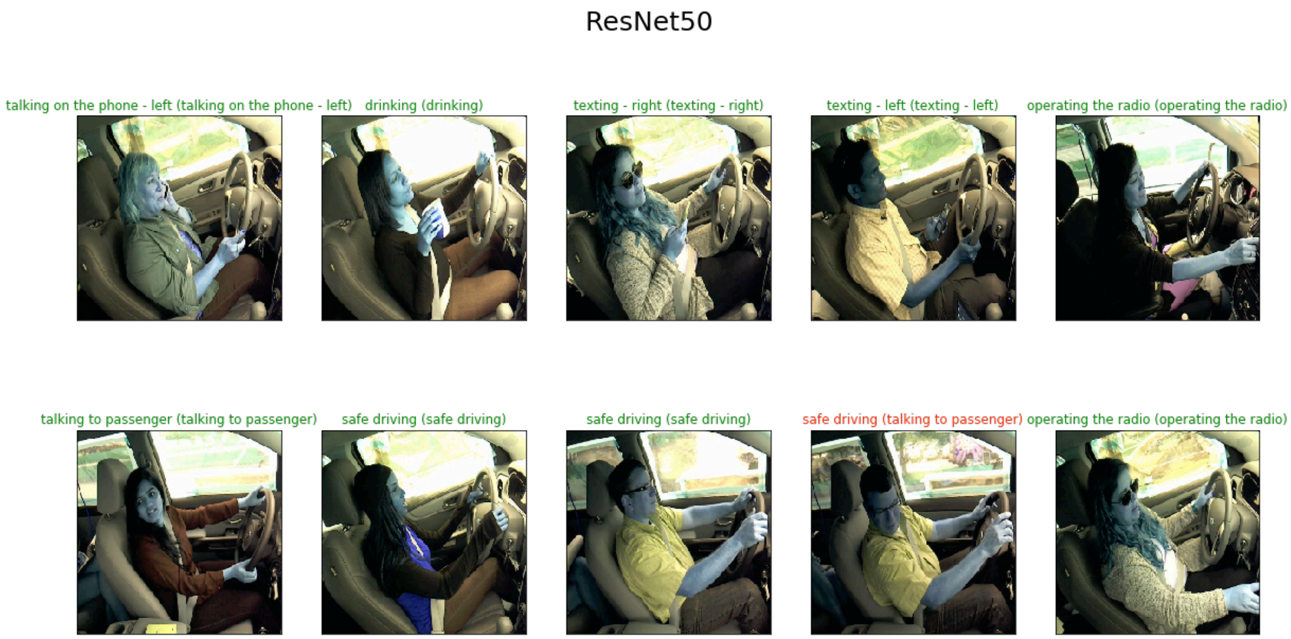
The importance of using K-fold cross validation to solve this problem can be seen, as well as the slight improvements of using 2 models. The validation loss differences of one iteration to the other shows us, that all pictures have important features. If the images were very much alike, the validation loss would have stayed constant and it's because of this that we were not able to get good results by to removing images to maintain an equal sample size across categories. Now from this graph, we can see that there are times when one model captures features better that the other but it's important to note that neither model outperforms the other. When both models have their validation scores averaged per K-fold, we can see a slight improvement in validation loss. A bigger improvement would be seen when we merge all K-fold predictions into one. Averaging would help bump generalization by neither overfitting on the validation data nor the train data.

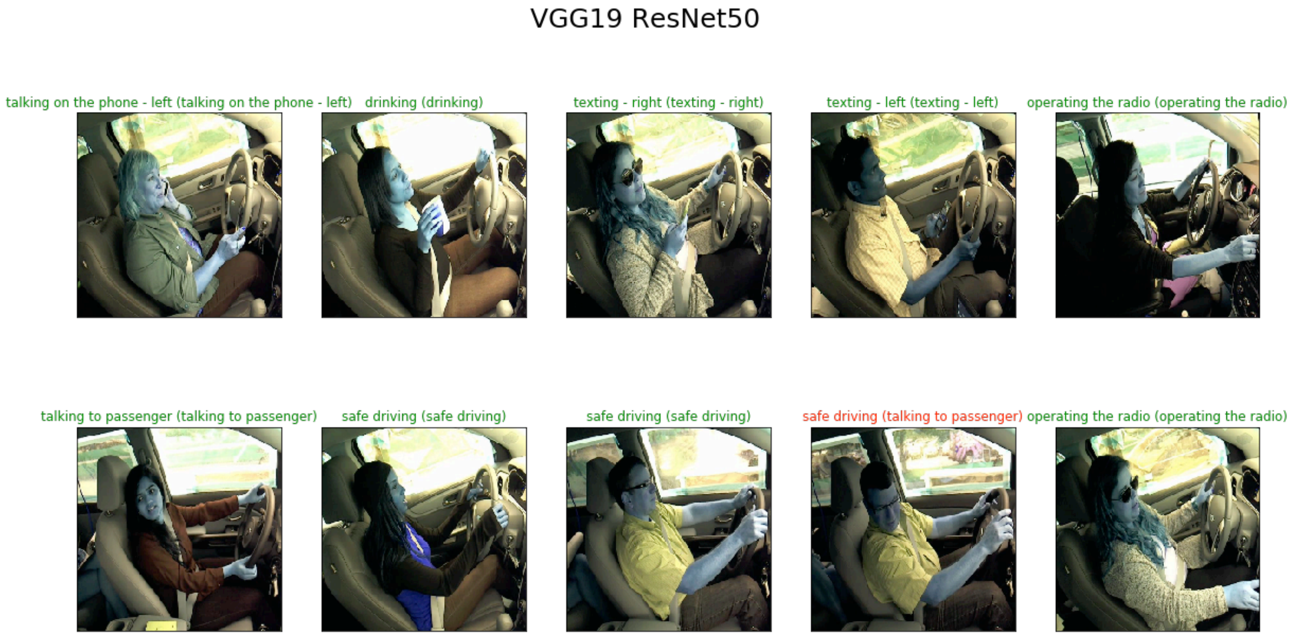
These are 10 random images and their predictions per model:


Model Validation Loss of All Folds
VGG19
ResNet50
VGG19 ResNet50
Loss
0.6
0.5
0.4
0.3
0.2
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Folds

VGG19


talking on the phone - left (talking on the phone - left)
drinking (drinking)
texting - right (texting - right)
texting - left (texting - left)
operating the radio (operating the radio)
talking to passenger (talking to passenger)
operating the radio (safe driving)
safe driving (safe driving)
talking to passenger (talking to passenger)
operating the radio (operating the radio)


ResNet50
talking on the phone - left (talking on the phone - left)
drinking (drinking)
texting - right (texting - right)
texting - left (texting - left)
operating the radio (operating the radio)
talking to passenger (talking to passenger)
safe driving (safe driving)
safe driving (safe driving)
safe driving (talking to passenger)
operating the radio (operating the radio)


VGG19 ResNet50
talking on the phone - left (talking on the phone - left)
drinking (drinking)
texting - right (texting - right)
texting - left (texting - left)
operating the radio (operating the radio)
talking to passenger (talking to passenger)
safe driving (safe driving)
safe driving (safe driving)
safe driving (talking to passenger)
operating the radio (operating the radio)

We can see that all 3 models yield satisfactory results and the error was also differing from model to model. Both ResNet50 and VGG19 were successful in classifying 9 out of 10 images correctly. Though the combined model is better at generalizing data that leads to provide a better log loss but that does not necessarily mean it fixes all the mistakes of the two models.

Reflection

The process used to define and develop the model can be summarized as follows:

1. An initial problem is identified and relevant datasets were obtained
2. The datasets were downloaded and pre-processed
3. Relevant hardware is obtained from cloud to account for huge number of images in the dataset
4. The highest score in the Public Leadership board was taken as a benchmark and a target rank was defined.
5. Implement different models for optimum solution.
6. Test each solution
7. Apply and refine working solutions
8. Iterate steps until satisfied with results

Improvement

Following are the areas for improvement

1. To further improve the score, we need to consider using bigger size when re-sizing the image. Currently the algorithm was trained by using 224x224 re-sized images. An image size of 480x480 might be relevant as original image size is 640x480.

2. Due to time constraint, number of folds was decreased. Increasing the folds and epochs could provide a better result.

3. We could also try using the other pre-trained models and verify the score such as
    - Inception bottleneck features
    - Xception bottleneck features

4. Model Ensembles and use of advanced Image Segmentation algorithms such as R-CNN,Fast R-CNN,Faster R-CNN and Mask R-CNN can also be investigated.

References

1. National Center for Statistics and Analysis. *Distracted Driving: 2015*, in *Traffic Safety Research Notes. DOT HS 812 381*. March 2017, National Highway Traffic Safety Administration: Washington, D.C.

2. Murtadha D Hssayeni, Sagar Saxena, Raymond Ptucha, Andreas Savakis. Distracted Driver Detection: Deep Learning vs Handcrafted Features.

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

4. Luisa Rojas, “Distracted Driver Detection Using Convolutional Neural Networks and Transfer Learning”