IRIS CONTROL REPORT

Objective: Driver Distraction Detection

Abstract:

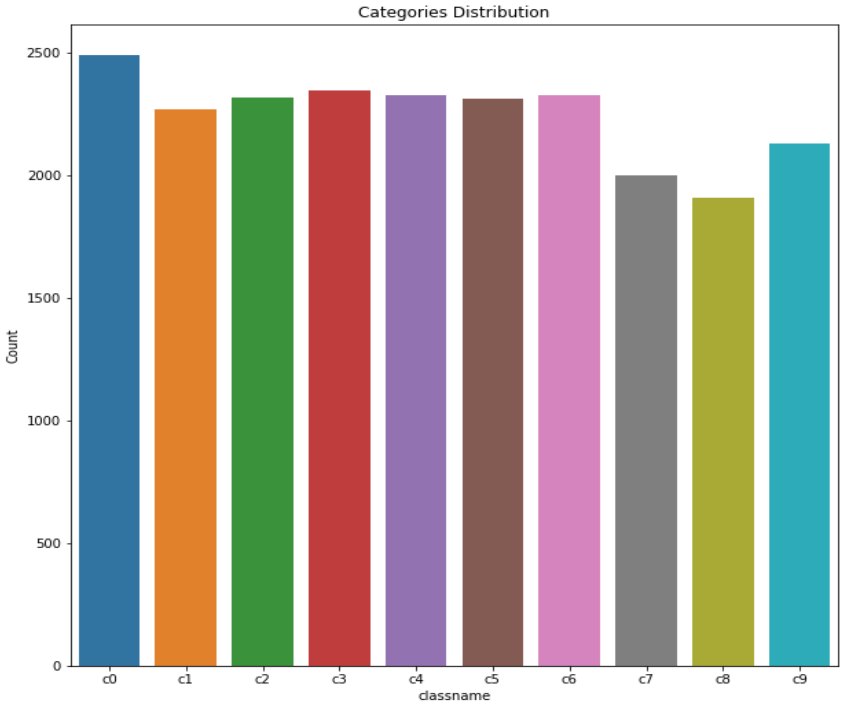
According to the CDC motor vehicle safety division, one in five car accidents is caused by a distracted driver. This roughly translates to 425,000 people injured and 3,000 people killed by distracted driving every year.

This algorithm helps in the detection of drivers who have been distracted due to one of the nine given reasons

Data Engineering:

Data is obtained from Kaggle "**State Farm Distracted Driver Detection**", and contains images of people involved in one of these 9 types of distractions. The volunteers are chosen from different race and ethnicity, to make the model created with this dataset much more robust. The train and test data are split on the drivers, such that one driver can only appear either train or test set

The images are of 640x480 pixels and changed to 64x64 pixels due to hardware bottleneck and the color scheme is changed to greyscale



.

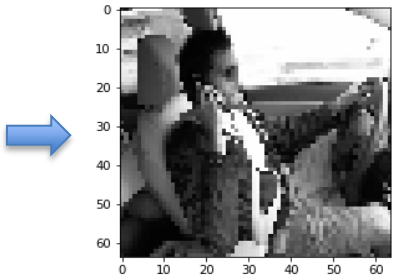
Data:

102150 total images.

` 17939 training images

4485 validation images.

79726 test images.



Identifiable distractions:

Safe Driving

Texting - right, left

Talking on phone - right, left

Operating the radio

Drinking

Reaching behind

Hair and Makeup

Talking to passenger

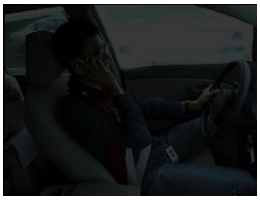
Augmentations:

Horizontal and Vertical Augmentation

Horizontal and Vertical Flip

Random Rotation

Random Brightness



Architecture:

Version 1:

The architecture is a vanilla CNN model with 4 Convolutional Layers and a Max Pooling layer after every convolutional layer and a dropout of 0.5 is applied twice. The model has a total of 5.6 million parameters. The training data is sent in batches of 40 and is set for a maximum of 10 epochs and is set to stop whenever the validation stop isn't improving, which almost always happens at 3rd or 4th epoch.

RMSprop is used as the optimizer and with Categorical Cross entropy for validation.

(Convolution
Max Pooling) * 4
Dropout (0.5)
Flatten
Dense
Dropout (0.5)
Dense

Model Accuracy: 0.983

Version 2:

The architecture is a vanilla CNN model with 6 Convolutional Layers and a Batch Normalization layer (as a set of 2 Convolution and Batch Normalization layer) after every convolutional layer and a dropout of 0.5 is applied five times. The model has a total of 4.5 million parameters. The training data is sent in batches of 40 and is set for a maximum of 10 epochs and is set to stop whenever the validation stop isn't improving (epoch 6 in my model).

RMSprop is used as the optimizer and with Categorical Cross entropy for validation.

(Convolution
Batch normalization
` Convolution
Batch normalization
Max Pooling
Dropout (0.5)) * 3
Flatten
Dense
Batch normalization
(Dropout (0.5)
Dense) * 2

Model Accuracy: 0.977

Improvements:

The above architectures are Vanilla CNN architectures, should try any standard architectures.

Should try transfer learning with pretrained models. (ResNet -18 is proven to be have higher accuracy in these kind of datasets)