Sujeeth Shetty 03/10/2020

BUAN 6357 Advanced Business Analytics with R

Project Update

Distarcted Driver Detection

Executive Summary:

One efficient way to reduce car accidents is preventing distracted driving, which is the act of driving while engaging in other activities such as texting, talking on the phone, etc. Activities of that nature distract the driver from paying attention to the road. These distractions in turn compromise the safety of the driver, passengers, bystanders and others in other vehicles. The United States Department of Transportation states that one in five car accidents are caused by distracted drivers, meaning that distracted driving causes injuries to 400,000 people and claims the lives of 2,841 just in 2018. 2.9% of driver using handheld cell phones in 2017, which was down from 3.3% in 2016.

Introduction:

To improve the alarming statistics of car accidents, innovative methods should be tested. One such method would be to develop an algorithm to detect drivers engaging in distracted behaviors by feeding it dashboard camera images. This algorithm can then be used as an API in a device to classify the driver's behavior by checking if they are driving attentively, wearing their seatbelt and remind them if they are not.

The dataset provided by State Farm consists of images, which means that the most efficient way to tackle the problem at hand is to develop a neural network model and then train it on a training subset of the dataset with the objective to optimize a certain evaluation metric. The model will then be tested on a testing subset of the dataset and evaluated.

Data Exploration:

The dataset used in this project was provided by State Farm through a Kaggle competition, which has a set of images of drivers taken inside a car capturing their activities such as texting, talking on the phone, eating, reaching behind, putting on makeup, etc. These activities are classified into 10 classes as:

+ 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

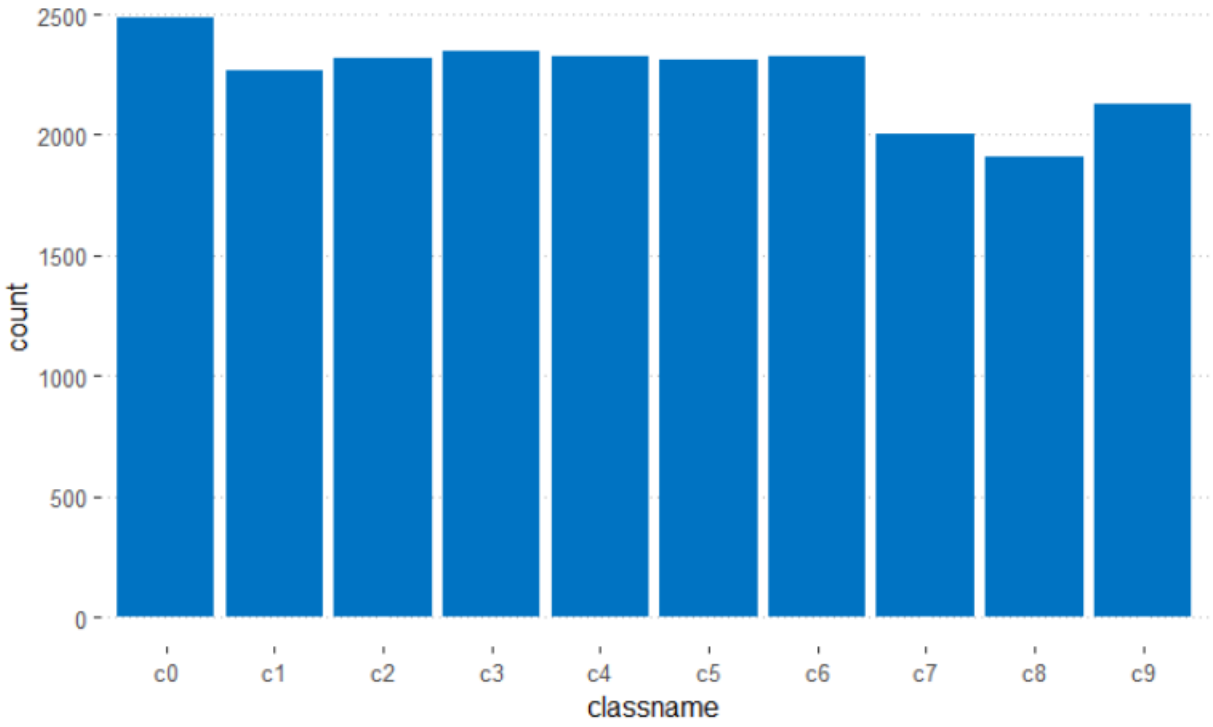
c9: talking to a passenger

The dataset contains a total of 102150 images split into a training set of 22424 images and a testing set of 79726 images. Here is a sample of the dataset images.

The images are 480 X 640 pixels and the distribution of the classes in the training set are relatively uniform. The total size of the dataset is 4 GB, which is publicly available at: https://www.kaggle.com/c/state-farm-distracted-driver-detection/data

Data Visualization:

One important factor in choosing an evaluation metric and a validation method is the uniformity of the target class distribution. The class distribution for this dataset is relatively uniform.



Algorithms and Techniques

The project will be executed using a convolutional neural network (CNN) of Keras Package. CNNs are made of neurons that have learnable weight and biases, each neuron receives an input, performs a dot product and usually followed by a non-linear function. The inputs of CNNs are image pixels as vectors and its outputs are class scores. As the vectors go from the input layer to the output layer they pass through a series of hidden layers, each hidden layer consists of neurons where each neuron is fully connected to every neuron in the previous layer while remaining completely independent from the other neurons in the same hidden layer.

Data Preprocessing:

The training set which holds 22424 images has taken over 20 GB of memory and test set has a memory of 61 GB which holds 79726 images with no labels. So, a subset of test images with 100 for each category and 400 random test images were considered for the initial run.

```
Number of images per class:
  0   1   2   3   4   5   6   7   8   9
100 100 100 100 100 100 100 100 100 100
```

Moreover, the project was set to use Keras with TensorFlow as backend which means that the input has to be a 4D tensors to be compatible with Keras' CNN, so the images were first resized to 256 x 256 pixels from 480X640 pixels, then converted into a 3D tensors and then into a 4D tensors of shape (N, 256, 256, 3) where N is the number of images. After that, the tensors were scaled by dividing them over 255.

Implementation:

The initial model was implemented first by creating a model through the use of keras' function Sequential and adding a Convolutional layer of 16 filters, kernel size 2X2, same padding, relu activation and input shape the same as the 4D tensors that are (None, 256, 256, 3). A second hidden layer of 128 filters, kernel size 2X2, relu activation was added which followed by a 2X2 max-pooling layer. The difference between the first Conv layer and the second is that no input shape was specified as that only required for the first layer of the model. The model was regularized by adding Dropout layer of rate 0.2 followed by a Flatten layer. The output had Dense layer of 10 outputs representing the data classes with a softmax activation.

Model
Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_16 (Conv2D)	(None, 256, 256, 16)	208
batch_normalization_16 (BatchNormalization)	(None, 256, 256, 16)	64
conv2d_17 (Conv2D)	(None, 255, 255, 128)	8320
batch_normalization_17 (BatchNormalization)	(None, 255, 255, 128)	512
max_pooling2d_6 (MaxPooling2D)	(None, 127, 127, 128)	0
dropout_8 (Dropout)	(None, 127, 127, 128)	0
flatten_2 (Flatten)	(None, 2064512)	0
dropout_9 (Dropout)	(None, 2064512)	0
dense_1 (Dense)	(None, 10)	20645130

Total params: 20,654,234
Trainable params: 20,653,946
Non-trainable params: 288

Preliminary Conclusions and Additional Analysis:

While fitting the model, an error occurred, which could be because of the image_load function which is unable to find the Pillow package. I am currently working on fixing this issue as the Miniconda need to be reinstalled as well as TensorFlow. The next step after the initial run would be to run the model on entire dataset and experiment with additional layers/neurons. Also tune the model with various hypermeters(epochs, learning rate) to get better accuracy.

To build a powerful image classifier using very little training data, image augmentation is usually required to boost the performance of deep networks. **Image augmentation** artificially creates training images through different ways of processing or combination of multiple processing, such as random rotation, shifts, shear and flips, etc. In addition to the scaling, Image augmentation will be performed in the final model.