Using Images to Predict Driver Distraction

Deep Learning Project Final Report

June 26, 2018

1 Abstract

According to the CDC motor vehicle safety division, on average 1.3 million car accidents occurs per year and one in five these occurrences is caused by a distracted driver. This leads to 260K deaths due to accident caused by distracted drivers. In an attempt to reduce car accidents caused by distracted drivers and mitigate damages by such horrible occurrences, a data competition hosted by Kaggle provided images data for data scientist to generate solutions powered by machine learning. The goal of this project is to build classification models to predict if a driver is not focused on the road while driving on a given image. Being able to correct determine possible distracted drivers on a image can be used in the future to possibly prevent car accidents caused by distracted drivers.

2 Team

Jose Rodilla and Jade Yun

3 Description of your problem

The problem is a multi-class image classification problem. The goal is to classify each image into the correct category. There are 10 labeled categories, where one specific class indicates normal driving behavior and the nine remaining classes exhibit some form of distraction. For further details, please refer to the data section. The metric used to evaluate our model is multi-class logarithmic loss.

4 Data

The data for this project is from the Kaggle competition "State Farm Distracted Driver Detection". Here is the link to the data source. The data consists of images with 10 distinct classes. Class 0 refers to images in which drivers are not distracted. The remaining classes refer to different types of driver distractions. The table below contains a detailed description for each class.

Classes	Description
0	safe driving
1	texting using right hand
2	talking on the phone and holding the phone with the right hand
3	texting using left hand
4	talking on the phone - and holding the phone with the left hand
5	operating the radio
6	drinking
7	reaching behind
8	touching hair or doing makeup
9	talking to passenger

Table 1: Class Label and Description.There is a total of 10 classes. Only one of the classes refers to non-distracted drivers, with the remaining 9 corresponding to different types of driver distractions.

There are 22424 images in the train set and 79726 images in the test set. Approximately 12.5 percent of train signifies the non-distracted drivers. The distribution of train set is the following:

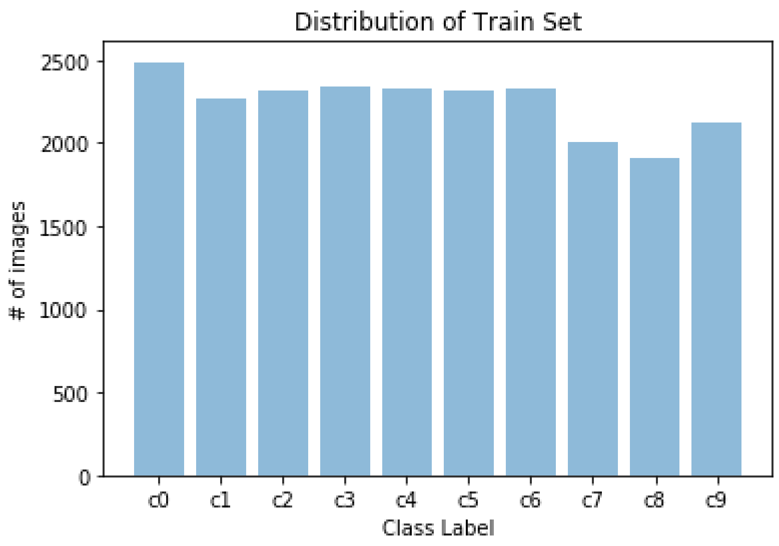


Figure 1: The image counts of each class in train set.

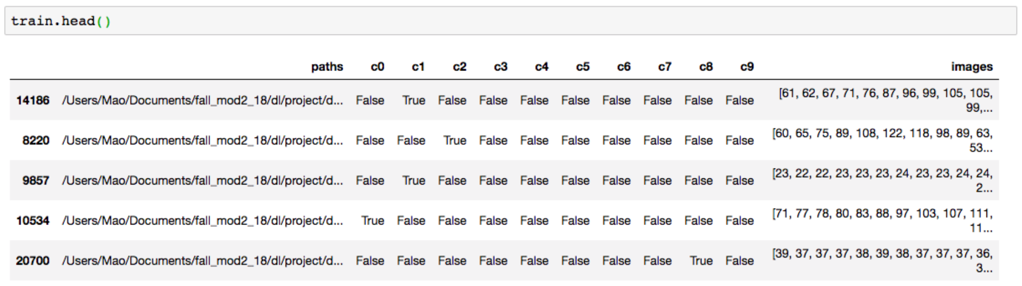
The distribution of train set indicates a fairly balanced data in each classes.

5 Methods

There are two main models we adapted for this project. A random forest and resnet34. The random forest served as a baseline model to be compare with the deep learning model we built.

5.1 Random Forest

For the baseline model, we processed data into a data frame that consists of the path, images converted into list of pixel values and one-hot encoded the class labels. Below shows how the data frame used to train the model looks like:



```
train.head()
```

	paths	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9	images
14186	/Users/Mao/Documents/fall_mod2_18/dl/project/d...	False	True	False	False	False	False	False	False	False	False	[61, 62, 67, 71, 76, 87, 96, 99, 105, 105, 99,...
8220	/Users/Mao/Documents/fall_mod2_18/dl/project/d...	False	False	True	False	False	False	False	False	False	False	[60, 65, 75, 89, 108, 122, 118, 98, 89, 63, 53...
9857	/Users/Mao/Documents/fall_mod2_18/dl/project/d...	False	True	False	False	False	False	False	False	False	False	[23, 22, 22, 23, 23, 23, 24, 23, 23, 24, 24, 2...
10534	/Users/Mao/Documents/fall_mod2_18/dl/project/d...	True	False	False	False	False	False	False	False	False	False	[71, 77, 78, 80, 83, 88, 97, 103, 107, 111, 11...
20700	/Users/Mao/Documents/fall_mod2_18/dl/project/d...	False	False	False	False	False	False	False	False	True	False	[39, 37, 37, 37, 38, 39, 38, 37, 37, 37, 36, 3...

Figure 2: The dataframe used for train the random forest model.

Then, processed the test data in the same way. Three-fold cross validation was used when training the model.

5.2 Resnet34

The first thing we had to decide for our resnet model was which image augmentations to apply. This problem is slightly different to most other problems, since the images are a bit "artificial". All of them were taken in identical conditions, from the exact same angle inside cars, with drivers appearing in more or less the exact same position and size. The pictures in the test also have these characteristics, therefore, augmentations such as horizontal flip or random crop didn't seem pertinent, since the result wouldn't match in any case what we would see in the test set. In other words, we have to account for very little potential variation. We thus decided, as a first option, to do nothing more than apply a small rotation to each picture.

The next issue we had to deal with was splitting the full training set into train and validation sets. One characteristic of the test set is that the drivers shown in the pictures do not appear in the training set (there is no driver overlap). We thus thought that the best option to split training and validation sets would be to ensure that there was no driver overlap amongst them.

Since we have image files across many directories (one for each of the ten classes), together with the data-split we had to carry out some pre-processing to build the final train and validation data frames. These data frames include the following columns:

- Path: string, with path to the image file
- Label: An integer between 0 and 9, depending on the class

Having these data frames simplified the next step–creating pytorch datasets–where we were able to just pass each of the dataframes. Aside from this, the only thing the Dataset does is center crop each image, reduce it to half its size, rotate it slightly (in case of the training set) and normalize it. Based on this datasets we were then able to define dataloaders with a batch size of 64.

Our model was similar to the one used in lesson 6; a resnet34 with added layers. The main difference is that our final layer outputs 10 values instead of one (since we have 10 classes). We also had to apply the sigmoid function to our final layer.

For our learning rates, we performed a learning rate range test, in order to come up with lower and upper bound learning rates that we could then apply to triangular learning steps.

6 Results

For the baseline model–random forest–the logloss on test set is 1.97495.

In the case of resnet, we acknowledge that we must be doing something wrong. We have tried hard to adapt the image classification problem to our needs (multi-class etc). We have stumbled upon many roadblocks which we weren't too sure how to overcome. We overcame a lot of them after struggling for a while, but in the end we have not been able to solve all of our problems.

Initially we encountered a bug where most of our predicted rows consist of all zeros ([0, 0, 0, 0, 0, 0, 0, 0, 0, 0]). Some rows present the correct structure ([0, 0, 0, 0, 0, 0, 0, 1, 0, 0]), but they are not the majority. With this issue, it is not hard to see why our model scores so poorly on Kaggle (score of 4.235). We will try to solve these issues before our presentation. After using long tensors for our final predictions was rounding all numbers, thus all the rows with zeros. Using float tensors solved this problem. We are now getting 10-dimensional vectors containing soft predictions. The score, however, is still pretty bad: 3.32 after submitting to Kaggle.

7 Conclusions and Lesson Learned

The conclusion is that using deep learning techniques such as Resnet34 to build a well performed model is not an easy task. Although our prediction is not the optimal in this case, we will continue to try to optimize it until end of this week.

The main lesson learned is that deep learning requires a deep understanding of both the mathematical aspects and the functionality of the libraries in use. It is easy to follow along what is going on from someone else's notebooks, and solving a similar problem is quite trivial. However, when it comes to solving a problem which isn't quite what you've seen before, the lack of a deeper understanding can make things much more difficult than anticipated.

On the positive side, we think we have learned well some preliminary steps, like pre-processing the data for deep learning and creating datasets. We have also learned a lot about what the library wants or expects in certain cases, and what it doesn't want in others.

8 Work Load Distribution

Jade: EDA and Baseline Model
Jose: Resnet

9 Repository

Here is our code.