Final Project Report for CS 175, Spring 2018

Project Title: Distracted Driver Detection
Project Number: 1

Student Name(s)
Zhiyuan Liu, 25105882, zhiyual9@uci.edu
Name2, StudentID2, uci_email_address
(Name3, StudentID3, uci_email_address)

1. Introduction and Problem Statement (1 or 2 paragraphs)
[This can be similar to what you wrote in your proposal or progress report]

Each year, the number of road accidents increases continuously, and from the data according to the CDC motor vehicle safety division, one in five car accidents is caused by a distracted driver. Since one major cause of the road accidents is due to distracted drivers, State Farm hopes to improve these alarming statistics, and better insure their customers, by testing whether dashboard cameras can automatically detect drivers engaging in distracted behaviors. The task is to detect what a driver is doing from images taken of this driver in the driver's seat, and there are a total of 10 classes of behaviors that we need to detect including nine kinds of distractions and one which there are no distractions.

We train our model using some of the images, and after that, we input the images of drivers into our model, and then the model would output one of the 10 different behaviors. We can test the accuracy of the model's prediction by using other images that provides us the correct behavior.

2. Related Work: (1 or 2 paragraphs)

There have been many attempts to build algorithms that can achieve the highest prediction accuracy from other people and organizations in the past. some of the most well known models used to solve this problem are VGG16, ResNet-18, Inception v3, and Inception v4, as well as MobileNet. In this project, we would evaluate the performance of some of these models, and after that, using existing knowledge, we can construct our own convolutional neural network to see the difference in performance.

3. Data Sets

[This should have considerable detail – make sure you include a good description of your data set(s) – figures and tables are strongly encouraged. Can be an updated version of what you wrote before.]

In this challenge given by State Farm, there is a grand total of more than 100,000 images in the dataset divided into the training category and testing category. Each image is an image of a specific driver doing one specific behavior from the 10 classes of behaviors. There are 26 drivers in total. In the training category of the dataset, images are further divided into 10 labels which corresponding to the 10 classes of behaviors that each image is in. the 10 classes mentioned above are:

c0: normal 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

For each kind of label in the training part of the dataset, there are from around 1,900 to 2,500 images per class, making the total number of images in the training dataset to be 22,400 images. The rest of images in the dataset are in the testing category of the dataset that have in total 79,726 images. All of the images have a resolution of 640 * 480 pixels and were taken in a very similar manner.

This is one example image from the dataset which will also be used as an input to our model. In this image, the driver is talking on the phone with his right hand, and the goal of our model is to successfully predict that behavior which is in “c2” class.

4. Description of Technical Approach [at least 1 page]

Since using a GPU is much faster in term of performance than using a CPU regarding using neural networks, we decided to use Google Colab to do the work so that we have access to a GPU. The first part of our program is to successfully load and process the data. However, since the size of the data is extremely large, and we cannot simply access local files containing the dataset from Google Colab, we must use efficient ways to load the dataset into our program. Therefore, we installed “kaggle” into our notebook, and use the api from Kaggle to directly download the dataset onto Google Colab. In order to achieve this, we must also import “drive” from “google.colab” since we need to put the json file containing the username and key in order to use Kaggle api here. After downloading the zip file of the dataset using Kaggle, we then were able to unzip the zip file to successfully get access to the dataset containing all those images. After we have access to the data, we constructed multiple graphs and other types of data structures to get a better understanding about the structure of the dataset in order to better work from there. We found that even though the size of the dataset is enormous, there are only 26 drivers in total. We used all the training set given in the dataset to train our model, and when we tested the model, we only used 500 test samples from the dataset because there were no significant differences when using larger test sizes.

When we attempted to solve the actual problem, we used two different approaches to try to get the best results. The first approach is to build our own model by constructing convolutional neural networks. The other one is to use some well-known pre-trained models that were online to see the results. In both approaches, the images in the dataset were converted to Gray scale for faster performance, and we also changed the resolution of each image in the dataset to 64 * 64.

In the first approach, we constructed codes to try to use multiple combinations of convolution layers with epoch number from one to ten. In this model, we used some regularization techniques as well. We used Dropout, which can efficiently reduce the overfitting by dropping out randomly some neurons; and in each convolution layer in this model, we used standard dropouts. Batch Normalization was also used to help improve the performance and stability by forcing the activations through a layer of network to follow a normal distribution. In the end, after some changes to the variables in the model, we were able to build graphs of both model accuracy and model loss for both training and testing dataset with different number of epochs from one to ten. The testing accuracy in the end was roughly equal to 97%.

As for the second approach, we used transfer learning method with is to use a pre-trained convolutional neural network model on a large dataset as an initialization. The model we used is called VGG 16, and it is well-known within the machine learning community with many attempts from others to use it on this project. However, in the end, it actually gave us worse performance than the convolutional network we built in the first approach above. The reason could be that the sample size we chose was not good. In addition, it took a long time to train this model due to its depth and number of fully connected nodes.

5. Software [at least ½ a page]

[Note: this is intended to be a high-level description of your software, not the code itself. Separate this section into subsections of (a) code or scripts you have written, (b) code or scripts written by others that you used in your project (with attribution/references). Tables could also be used to summarize the code]

For writing the codes for this project, the software we used is called Google Colab as well as Google Drive to store necessary recourses during the project. Google Colab is a Jupyter Notebook like software that can be used online, and one main advantage that this software gave us is that we can use GPU during the runtime to train and run our neural network models as there is no Nvidia GPU on the computer I'm working on. This gives huge advantages during the runtime and increases the performance drastically.

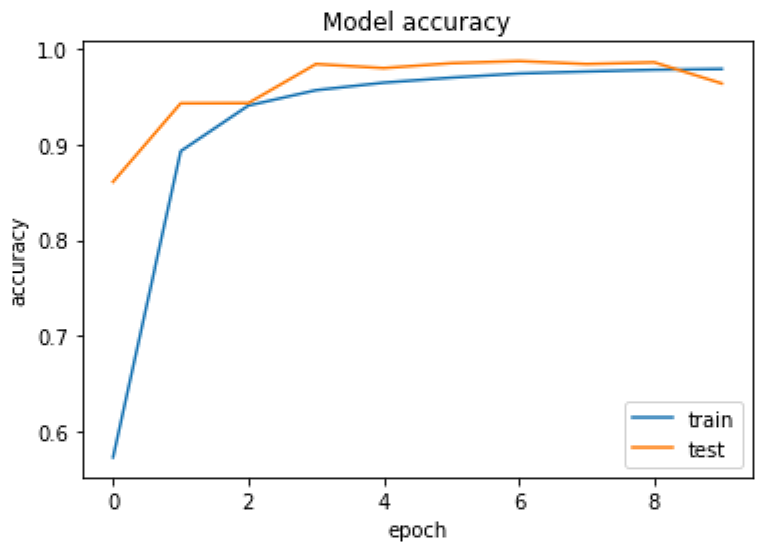
There is also software that I used from other people when trying to complete this project. One of them is Keras/TensorFlow which give the ability to use it as a framework to implement models that are useful for training the neural networks. It is also the main software used in this project. Matplotlib is also used such that I have the ability to construct graphs from data structures in my project. Visualizing the data and results using this software is extremely helpful when trying to compare different parameters and models. Another one that I used is called cv2, and it is used to read images from the dataset and change the sizes and colors of all the images in the dataset.

6. Experiments and Evaluation [at least 1 page]

In order to do the evaluation of different models that we constructed in this project; we have to first decide the size of testing. In the training set, we first split the data into training and testing, and the test size if 20% of all the data. As for the testing on the testing dataset, we chose the test sample size to be 500 samples. The size is large enough to perform testing of our models and is very efficient to test and know the performance when we were doing testing on this sample testing dataset.

For each model we used, we also used the method of constructing a graph to check the performance of different models regarding the accuracy of their predictions on the testing and training dataset with the change of the number of epochs in each model. We epoch number we used is 10, even though from the graph, we are able to find that all the models will have very similar performance at around three to four and all the number of epochs after that.

The graph for the model accuracy of the model that we built by ourselves in the first approach is below:



We can see that the accuracy value stays nearly unchanged after 3 epochs or 4 epochs, and it has an accuracy of above 0.90. Another method we tried to use is early stopping when we find that validation loss did not improve after a specific number of epochs, but we still left this part out just to be safe, and we wanted to see a larger picture of our model from the graph.

7. Discussion and Conclusion [at least ½ a page]

Both models performed well with my own model having around 98% validation accuracy and the other one having more than 70% accuracy. This project can be helpful to the insurance company and people working on these areas.

From completing this project, I learned a lot about how to work on real life problems and data using the knowledge acquired on class. I realized that building a model for a problem like this is much more complicated than what we normally do like in homework. Something I did not expect from the results of this project is that the performance and the accuracy of the VGG 16 model was not as good as expected given its reputation and wide usage. It could be that I was not using it as how it should be used. One thing that I realized but failed to resolve is about the accuracy of the model. Even though the number looks extremely high on paper, but since the dataset only contains 16 different drivers in total while having around 100,000 images, a lot of the images are largely similar with the same driver doing the same thing. It becomes very easy for the model to predict because it was trained much of the same information. In the future, if I want to make the model better at the job it is doing, I must consider this limitation, and build models based on the tests of different drivers instead of just using a simple 80-20 split on the dataset for testing. This is crucial but often easily overlooked when really testing the accuracy and ability of the models. I could also use more models when doing transfer learning and compare more models to get the best results. Some examples of such models include ResNet and Inception.