Distributed Driver Detection Model

CMPE 255-02: Data Mining

Project Report

Submitted By:

Project Group 2

Team Member	Student ID
Manisha Shivshette	**012560353**
Kashika Jain	**012505649**
Vishwanath Patil	**012526410**

1. ABSTRACT

The objective of the project is to solve a real life problem of distracted driving. Road accidents are the major cause of deaths and have been continuously increasing in last few years around the world. As per the overview of National Highway Activity Security Chairman, about one in five motor vehicle crashes are caused by distracted drivers. In this project we have attempted to create a strong framework for detecting distracted driver and to provide caution against it. Inspired by the execution of Convolutional Neural Systems in computer vision, we show a CNN based framework that not only detects the diverted driver but also distinguishes the cause of distraction. We have deployed two types of models in our project. Using the Sequential model from keras, we were able to achieve an accuracy of 97% while the other model used was SVM and gridsearchCV in python. The second model gave us an accuracy of 98% on the test dataset.

2. INTRODUCTION

Distracted Driving occurs when the designated driver diverts from his task and performs some other activity that can lead to very disastrous results. Finding a valid measures to detect distracted drivers has become a research focus. As per the Center for Disease Control and Prevention (CDC), there are mainly three types of distractions while driving: visual (When driver takes his eyes off the road), cognitive (when driver is mentally distracted) and manual (when a driver takes his hands off the steering wheel). In our project, we only focus on “manual distractions”. They are usually accompanied by changes in hand position and could been effectively distinguished by posture estimation. The dataset is a 2D dashboard camera images which included ten classes of postures like texting, drinking and other behaviors(which we have used as algorithm input), and require people to classify each image into one of the ten classes.

Our project presents a solution to this problem by using Convolutional Neural Network (CNN) algorithm and the combination of SVM and gridsearchCV.

3. MOTIVATION

The objective behind choosing this project to work with is because it aims to help amend the real life issues that are connected with distracted driving. The distracted driver doesn't only puts himself in harm's risk but is also responsible for the wellbeing of the people who share the ride with him. There are road accidents happening every day and the most important cause of such accidents are distracted driving. Our purpose here is to help ameliorate this situation by using machine learning tools. Machine Learning technology has been a great boon in helping with car accidents and a lot of research all over the world is happening to make driving safer. Predictive modelling has been one such beneficial tool and that's what made it an exciting problem for us to solve.

4. ALGORITHMIC APPROACH

We have taken into account 2 approaches to solve the problem of distracted driving - CNN and SVM. In both of these processes, we have applied the data preprocessing steps and and classified the images into a corresponding class. We have discussed both these approaches in the following sections.

Steps Performed -

1. Data analysis
2. Data Preprocessing
3. Model Building
4. Model Evaluation

5. DATASET AND PREPROCESSING

We have used the State Farm dataset from the Kaggle website. The images in the dataset are having a dimension of 640x480 pixels RGB with different drivers' behaviors. The purpose of this project is to classify the input images into different classes showing different divers' behavior, following are the classes into which the dataset is to be classified c0: safe driving, c1: texting on phone- right, c2: talking on the phone - right, c3: texting on phone - left, c4: talking on the phone - left, c5: operating the radio in the car, c6: drinking, c7: reaching behind, c8: hair and makeup, c9: talking to passenger.

C0	safe driving	C5	operating the radio in the car drinking
C1	texting on phone- right	C6	drinking
C2	talking on the phone - right	C7	reaching behind
C3	texting on phone - left	C8	hair and makeup
C4	talking on the phone - left	C9	talking to passenger

The description of the dataset is as follows :

1. training group that contains 22,400 images labeled with one of the 10 classes
2. testing group that contains 79,727 unlabeled images.
3. A CSV file containing the different divers' list. In which each driver is marked with subject number and also the corresponding class to which they belong.

Fig: Images of distracted drivers

We have 26 different drivers among the training examples. We havel used K-folds cross-validation method to split our training data into training set and validation set according to the driver in order to make sure that we do not overfit our training model by over-emphasizing on the features of drivers instead of their driving behaviors.

The training dataset may consists of many repeated images which may cause the model to overfit. Thus in order to avoid this we use inbuilt library of keras called keras.preprocessing.image.ImageDataGenerator. This class of the library allows to configure random transformation and normalization operations that can be done on the image dataset during the training instantiate generators of augmented image batches (and their labels) via .flow(data, labels) or .flow_from_directory(directory). These generators can then be used with the Keras model methods that accept data generators as inputs,fit_generator, evaluate_generator and predict_generator.

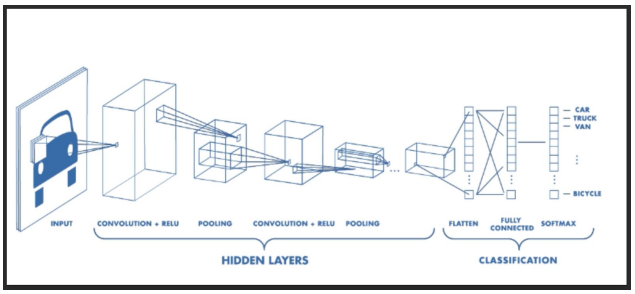
We have used following parameters of ImageDataGenerator class to fine tune model:

1. **rescale** is a value by which we will multiply the data before any other processing. Our original images consist in RGB coefficients in the 0-255, but such values would be too high for our models to process (given a typical learning rate), so we target values between 0 and 1 instead by scaling with a 1/255. factor.
2. **zoom_range** is for randomly zooming inside pictures.
3. **horizontal_flip** is for randomly flipping half of the images horizontally relevant when there are no assumptions of horizontal asymmetry (e.g. real world pictures).
4. **shear_range**: Float. Shear Intensity (Shear angle in counter-clockwise direction in degrees)

6. CLASSIFICATION USING KERAS NEURAL NETWORK

Neural networks are a group of individual units called neurons. Neurons are actually layered pattern. Neurons in each layer are interconnected to neurons in the next layer.The output of each layer is the input to the next layer. Each layer does some small mathematical computations and transfers the data to the next layers.
Convolutional neural networks are a special kind of artificial neural networks. One of the most important uses of CNN is image classification. Facebook uses it Resnet50 model for automatic tagging feature using face recognition.The main task of image classification is to accept the input and classify the images based on the defined classes.This is a skill which humans develop from their birth and are are easily able to distinguish between what they see hence the name neural network.

For this project we are using the keras library for the purpose of prediction. Keras has Easy and fast prototyping through total modularity, minimalism, and extensibility. Keras supports both convolutional networks and recurrent networks and combinations of the two. It can easily run on both CPU and as well as GPU making it machine independent. For the following project we are using the sequential model existing in keras. In a sequential model, we stack layers sequentially. Each layer has unique input and output, and those inputs and outputs then also come with a unique input shape and output shape. If we want to retrieve the weights a layer is having we can simply call get weights on layer and retrieve all those weights as as a list of numpy arrays. If we want to set weights, that's also possible by simply calling set weights with a given list of numpy arrays weights. Also, each layer has its defining configuration which we can get by calling get config on a layer.



Following are the steps are followed to build a sequential model in keras :

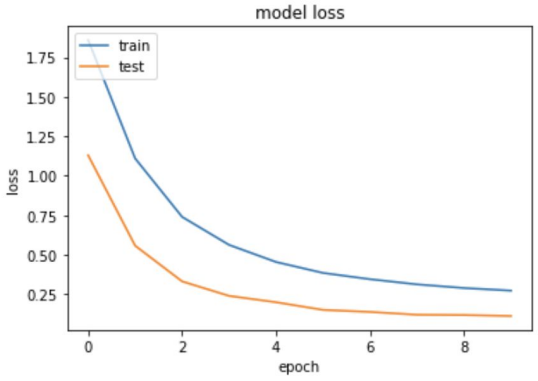
1. Instantiate a sequential model.
2. Add layers to it one by one.
3. Specify the input shape and pass the input shape to the first layer.
4. Compile the model with a mandatory loss function and a mandatory optimizer using the compile method.
5. Keras models are trained on Numpy arrays of input data and labels
6. Fit the model to the training dataset.

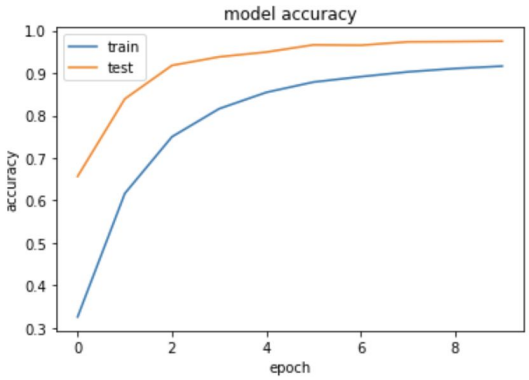
Model Implementation details

1. For intermediate layers we have used relu activation function
2. For final layer to classify images into 10 different classes, we have used softmax function.
3. We have experimented with steps_per_epoch values in set (100,200,300,1000,2000), steps_per_epoch =2000 gave the best results.
4. We limited training epochs to 10 to avoid model overfitting,as there was no significant improvement in model accuracy after epoch 6.
5. We used drop out rate of 25% for our convolutional layers and 50% of our fully connected layers to train small sets of neurons at a time, rather than the whole set. This helped neurons to specialize in specific tasks, rather than all neurons generalizing together.

7. PERFORMANCE EVALUATION USING KERAS NEURAL NETWORK

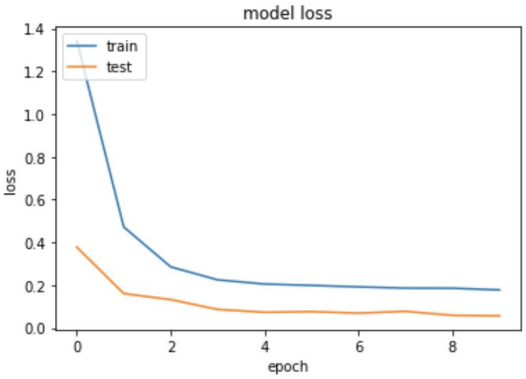
- Input image size: 32X32 Model training time: 1X Model Accuracy : 0.975

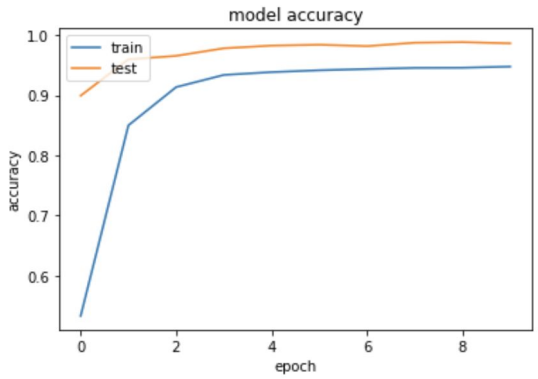




As we can see above there is very slight increase in model accuracy increase after 6th epoch.Hence to avoid the overfitting of model, training is stopped after 10 epochs.

➢ Input image size: 64X64 Model Training time :2X Model Accuracy:0.986



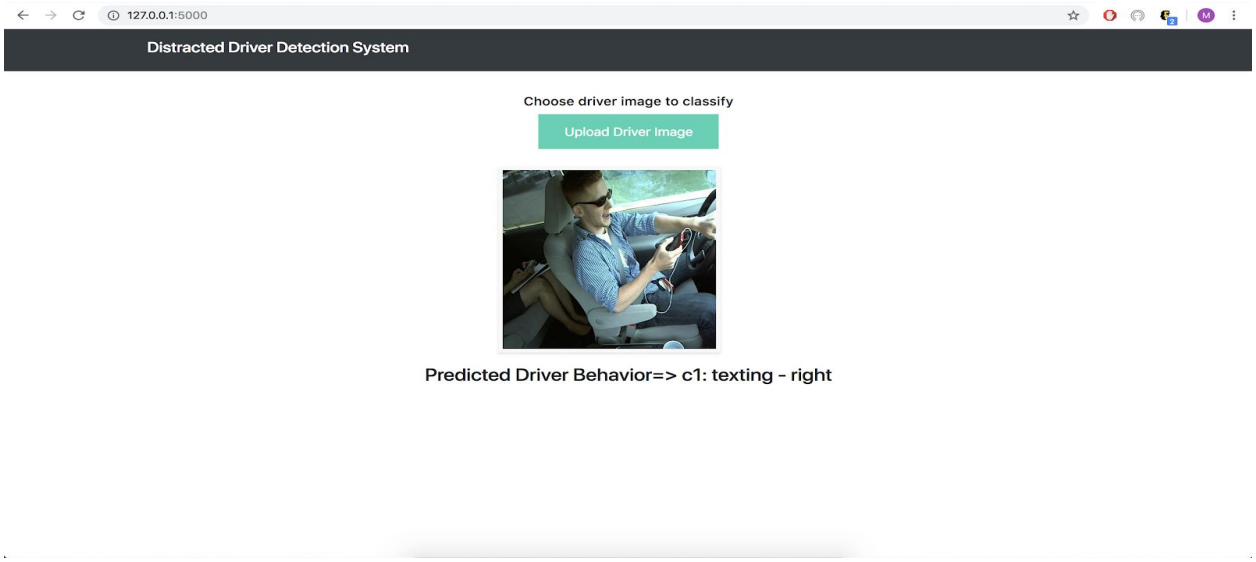


As we can see above there is very slight increase in model accuracy increase after 3rd epoch.Hence to avoid the overfitting of model, training is stopped after 10 epochs.

After comparing both models, we can conclude that to slight better performance can be achieved using 64X64 input images but it would increase the model training time twice. So there is trade off between training time and accuracy.

8.Deployment portal of Keras Model and Sample Test Results

We deployed saved keras model using FLASK library to demo this project wherein feeding any of the images from the test dataset will tell us that what kind of action the driver is actually doing. Thus classifying the images into one of the 10 classes.
The visual demo of the application is as shown below -



8. CLASSIFICATION USING Support Vector Machines:

8.1 Image Processing & Feature Extraction-

In our model using the SVM, we have performed image preprocessing and Feature Extraction by dimensionality reduction of the original image to 64X64 image. In order to make sure that the image does not lose its importance, we have implemented anti-aliasing technique and then the images are flattened to take into consideration the memory issues for processing large image files.

8.2 Model Building-

The SVM model works with the GridSearchCV algorithm. The data is split into 70-30 and grid search fits the parameters by using refit and gets the best value of C and gamma to improve the prediction accuracy. This idea of creating a 'grid' of parameters and just trying out all the possible combinations is called a GridsearchCV. The CV stands for cross-validation which is the GridSearchCV takes a dictionary that describes the parameters that should be tried and a model to train. The grid of parameters is defined as a dictionary, where the keys are the parameters and the values are the settings to be tested. One of the great things about GridSearchCV is that it is a meta-estimator. It takes an estimator like SVC, and creates a new estimator, that behaves exactly the same - in this case, like a classifier.

9. SVM Model Evaluation -

9.1 Classification Report-

The classes predicted for drivers using the SVM gives us almost perfect values for Precision, Recall and F1-score in the training data. The best score of the model is 99.5%

9.2 Confusion Matrix -

It gives the performance of a classification model on a set of test data for which the true values are known. It gives a table for comparison of True Positives with False Positives and True Negatives with False Negatives

	precision	recall	f1-score	support
0	0.99	1.00	1.00	752
1	1.00	1.00	1.00	700
2	1.00	1.00	1.00	748
3	1.00	1.00	1.00	677
4	1.00	1.00	1.00	697
5	1.00	1.00	1.00	696
6	1.00	1.00	1.00	673
7	1.00	1.00	1.00	607
8	1.00	1.00	1.00	574
9	1.00	1.00	1.00	604
micro avg	1.00	1.00	1.00	6728
macro avg	1.00	1.00	1.00	6728
weighted avg	1.00	1.00	1.00	6728

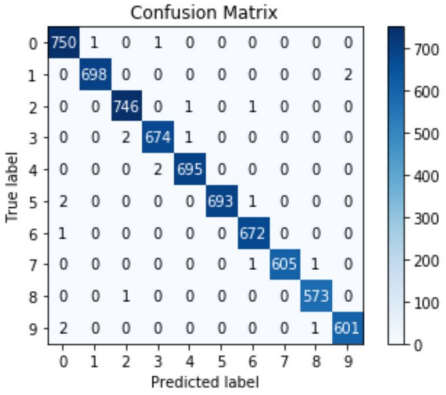


Fig: Evaluation Metrics For SVM

10. COMPARISON- The following section explains the basic comparisons between the two models.

10.1 Model fitting-

Used GridSearchCV to predict the accuracy of the model by passing different values for the respective parameters. Used keras built in sequential model for the purpose of prediction.

10.2 Prediction and speediness-

The svm model gave us a final accuracy of 99%. But took more amount of time to run and predict since it is not specially engineered for image datasets. The keras sequential model gave us an accuracy of 98.6%. This model was also able to run and predict very much faster compared to svm as these libraries are specially designed for image datasets.

11. CONCLUSION:

The project was successfully implemented to solve the distracted driver problem. By using the models CNN and SVM to train the data, extract the features from the images and create a fully-connected layer, the models were able to achieve 99% accuracy. Despite the fact that the classification needs to be done for images, the model does a very good job in classifying. Overall, from the study of these two models, we can see that the problem of distracted drivers can be effectively solved if we use these predictive models effectively and hopefully it will help in reducing the death rates resulting from distracted driving

REFERENCES-

[1]https://manning-content.s3.amazonaws.com/download/3/e0d6f80-038c-49b5-9c3d-1fd3c3bc9e4c/sample_ch10_Shukla_Maching-Learning_January23.pdf

[2]https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

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

[4] Yehya Abouelnaga, Hesham M. Eraqi, Mohamed N. Moustafa, Real-time Distracted Driver Posture Classification

[5] Taamneh, S. et al. A multimodal dataset for various forms of distracted driving. Sci. Data 4:170110 doi: 10.1038/sdata.2017.110 (2017)

[6] https://www.nhtsa.gov/distracted-driving/distracted-driving-kills

[7] https://brilliant.org/wiki/feature-vector/

[8]https://www.researchgate.net/publication/318029453_Realtime_Distracted_Driver_Posture_Classification

[9]https://github.com/mtobeiyf/keras-flask-deploy-webapp