Distracted Driver Detection

EE5180 Project
Instructor: Dr. Avhishek Chatterjee

Team 11
Mouni Preetham Malyala EE18B019
B Midhun Varman EE18B113
Dola Akhila Datta EE18B131

Abstract:
Distracted drivers are incredibly dangerous and cause a lot of accidents annually.
To combat this we implemented three different classification techniques namely Naive Bayes, Support Vector Machines and Convolutional Neural Networks to detect if a driver is paying attention or is involved in some other distraction activity, while also classifying the distraction activity.
To improve the classification, we implemented multiple different techniques and fine tuned the parameters to find which best suits distracted driver detection.

Introduction:
Distracted driving is any activity that diverts attention from driving such as talking or texting on your phone, eating or anything that takes attention away from the road.
The U.S Department of Transportation, National Highway Traffic Safety Administration estimates that close to 14% of all car accidents are caused due to distracted drivers making it the leading cause of road accidents.
This has only been worsened due to proliferation of new distraction triggers such as smartphones.
With distracted drivers being incredibly harmful it calls for the need to have some form of monitoring/detection system to identify distracted drivers and to notify the same to prevent accidents from happening.
This can be done by classifying images taken of the driver by training models that can identify distraction activities.
We experiment the detection of distracted drivers using three different classification models and identify in which situations each classifier is optimal.
Along with identifying the optimal classifier we also fine tune the parameters of the classifier to find the best possible accuracy for our distracted driver detection model.

Data Set Description:
The input dataset was collected from StateFarm [4] dataset(an open sourced publicly available dataset) which contains snapshots from a video captured by a camera mounted in a car.
The whole training dataset contains 22,424 labeled images and 79,700 unlabeled test images. The size of each image is 640×480 pixels.
Each image in the dataset can be classified into these 10 categories based on the activity the driver is involved in:

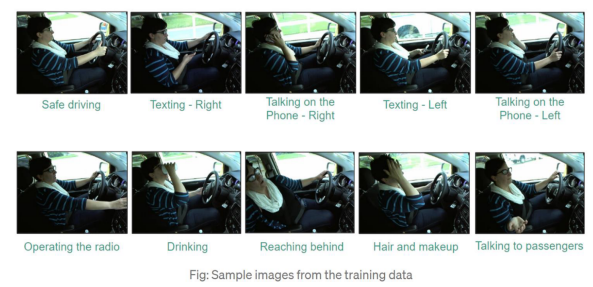


Fig: Sample images from the training data

Classification classes are as follows:
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 passenger

Previous Work:

Initial models were only used to detect distraction due to mobiles, further models classified the distraction activity as well.
In 2011, Zhang et al. used the Hidden Conditional Random Fields model based on face, mouth, and hand features to detect the use of mobile. Le et al. achieved a higher accuracy of 94.2% using the Faster-RCNN deep learning model but it was slow.
University of California San Diego's Laboratory of Intelligent and Safe Automobiles and Southeast University had created high accuracy models with 3-4 classification classes , but their datasets were not made public. Work on this problem statement was highly limited due to non-availability of public data-sets.
But in April-2016 the first publicly available dataset was released by StateFarm's Distracted Driver Detection competition on Kaggle.
In 2017, Abouelnaga et al. created a new dataset similar to the State Farm's dataset for distracted driver detection , they also proposed a solution using a weighted ensemble of five different Convolutional Neural Networks. The system achieved a good classification accuracy of 95.98%, but it's too complex for real-time detection..
Research in this field grew much further due to free and public availability of these two data sets.
Baheti et al. addressed this problem of time complexity and reduced the number of parameters significantly and achieved an accuracy of 95.54% - this was considered the best in terms of accuracy and computational time, except for some research works which used edge and cloud platforms apart from camera based identification to detect driver distractions .

Preprocessing Techniques:

Compressing the image:

The original image size is 640x480 pixels, with 3 channels for r,g,b respectively. When flattened into a 1d vector the vector has 9,21,600 elements.
To reduce the size of images and increase the speed of the models to make it feasible in real time detection we resize the image to 128x128 pixels. This way the number of elements in the 1d array is reduced to 49152 elements which is a significant reduction and increases the speed of the model significantly.
Original Image:

Resized Image:

This significantly improved the speeds in detection particularly for SVMs and CNNs and made it slightly more feasible for real time detection without compromising too much on accuracy

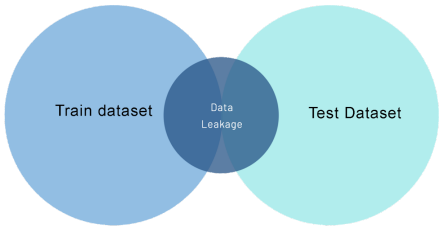
Data Leakage:

Data leakage happens when information from outside the training dataset influences the model creation.

This additional information can allow the model to learn or know something that it otherwise would not know and in turn invalidate the estimated performance of the model being constructed.

In the case of our model here, data leakage can happen when the person in the image is common for both the training and test data set where the model is highly accurate for the particular person but behaves poorly for other people.

This can be avoided by ensuring that both the training and testing data set does not have the same person.



We can do this for the dataset by splitting data based on people in the photo.

```
1 people = data.subject.unique()
2 print(people)
```

```
['p002' 'p012' 'p014' 'p015' 'p016' 'p021' 'p022' 'p024' 'p026' 'p035'
 'p039' 'p041' 'p042' 'p045' 'p047' 'p049' 'p050' 'p051' 'p052' 'p056'
 'p061' 'p064' 'p066' 'p072' 'p075' 'p081']
```

The above photo depicts all the unique people in the training data set. And we ensure that there is no common person while generating the test-train split.

For the test data the accuracy will be lower than if data leakage was not prevented, but is found to be more accurate on the validation data set available on Kaggle when implemented.

Classification Models:

1. Naive Bayes:

Naive Bayes classification model focuses individually on each element in our 1d array, hence the model develops a classification method based on pixel by pixel values.

The classifier assigns probabilities for each pixel value occurring in a particular class and assigns the image to a particular class based on the product of these probabilities.

Results and insights:

The results for a test-train split is as follows:

	precision	recall	f1-score	support
0	0.41	0.59	0.48	467
1	0.50	0.54	0.52	479
2	0.50	0.60	0.54	466
3	0.68	0.48	0.56	485
4	0.59	0.39	0.47	492
5	0.54	0.79	0.64	456
6	0.61	0.48	0.54	423
7	0.75	0.67	0.71	396
8	0.72	0.52	0.61	386
9	0.49	0.49	0.49	435
accuracy			0.55	4485
macro avg	0.58	0.55	0.56	4485
weighted avg	0.57	0.55	0.55	4485

Pixel by pixel classification is not the most accurate and Naive Bayes produces the least accuracy among the three models we built for driver detection.

But was significantly faster in learning compared to the other two models, (<10 mins compared to hours of learning for SVMs)

However, it is not feasible as a solution to distracted driver detections because of the poor accuracy.

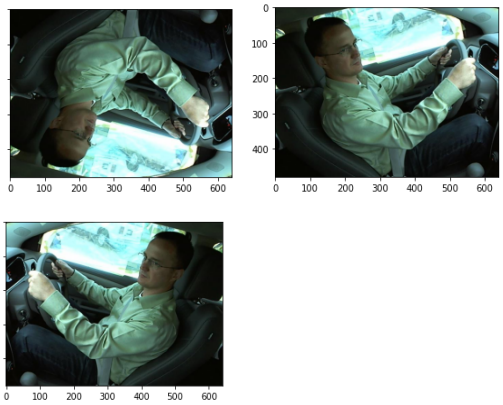
2. Support Vector Machines:

Support Vector Machines are discriminative classifiers that are based on statistical learning theory and can be used to identify patterns, classify them and infer non-linear relationships between variables.

Given labeled data, SVM generates an optimal hyperplane which classifies the new data points.
In an n-Dimensional space, this hyperplane is an n-1 plane dividing the space into parts. SVMs can generate both non-linear as well as linear classifiers efficiently with different kernels .We have resized the image to 64 x 64 for faster training.

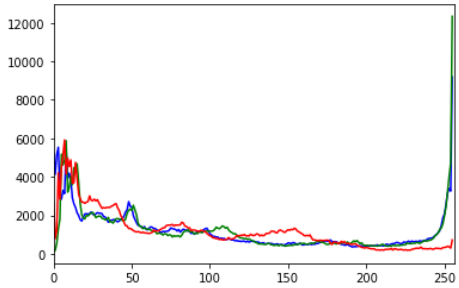
Feature Extraction:
Feature descriptors are a simplified representation of the image that contains only the most important information about the image. We would be using [2] HU moments, Haralick texture features and Histograms.

Hu Moments [1](or rather Hu moment invariants) are a set of 7 numbers calculated using central moments that are invariant to image transformations. The first 6 moments are invariant to translation, scale, and rotation, and reflection. While the 7th moment's sign changes for image rotation.

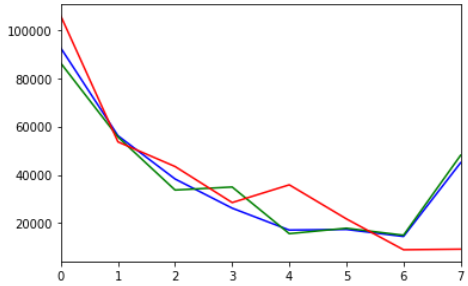


```
[ 1.54947119e-03  4.95366235e-08  3.11140345e-10  8.93288091e-11
 -1.47837571e-20  1.90539360e-14 -1.79580345e-21]
[ 1.54947119e-03  4.95366235e-08  3.11140345e-10  8.93288091e-11
 -1.47837571e-20  1.90539360e-14  1.79580345e-21]
[ 1.54947119e-03  4.95366235e-08  3.11140345e-10  8.93288091e-11
 -1.47837571e-20  1.90539360e-14  1.79580345e-21]
```

We can see that only last number is different ,rest 6 are invariant of rotation
Haralick texture features are based on the adjacency matrix (the adjacency matrix stores in position (i,j) the number of times that a pixel takes the value i next to a pixel with the value j. Standard practice is to average them out across the directions to get some rotational invariance.
A histogram displays numerical data by grouping data into "bins" of equal width. Each bin is plotted as a bar whose height corresponds to how many data points are in that bin. We utilised 8 bins for each RGB colour thus effectively reducing the features from 64 x 64 x 3 to 512. Each bins are calculated by summing adjacent pixels.



Plot of histograms one without bins



Plot of histogram with 8 bins by summing adjacent pixel values

Conclusion:
Using the RandomizedSearchCV function from Scikit learn while altering C and gamma parameters using the Radial basis function kernel as in most applications RBF produces the best accuracy but has higher complexity.Applying Data leakage by normalising after splitting train test split helped avoid higher pseudo accuracy and we reached an accuracy of 62 percent with

recall being lower than 40 for only two of the classes.

	precision	recall	f1-score	support
0	0.99	0.68	0.80	254
1	0.97	0.68	0.80	234
2	1.00	0.63	0.77	244
3	0.20	0.99	0.33	212
4	1.00	0.65	0.79	228
5	0.99	0.63	0.77	237
6	0.98	0.71	0.82	255
7	1.00	0.34	0.50	200
8	1.00	0.33	0.49	175
9	1.00	0.51	0.68	204
accuracy			0.62	2243
macro avg	0.91	0.61	0.68	2243
weighted avg	0.92	0.62	0.69	2243

Accuracy Score : 0.623272403031654

3.Convolutional Neural Networks:

A Convolutional Neural Network (CNN) is a [3] Deep Learning algorithm which can take in an input image, assign weights to various aspects/objects in the image and be able to differentiate one from the other.At present CNN is the state of the art technique for Image classification. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets has the ability to learn these filters/characteristics.The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex

Transfer learning:

Transfer learning is an approach in deep learning where knowledge is transferred from one model to another. We can make significant improvements in accuracy without much more training time by Transfer Learning.

We used VGG 16, resnet 50 and mobilenet models and formed an ensemble of them. We also used KNN to decrease the variance further.

VGG-16:VGG-16 mainly contains three different parts: convolution, pooling, and fully connected layers — it starts with two convolution layers followed by pooling, then another two convolutions followed by pooling, after that repetition of three convolutions followed by pooling, and then finally three fully connected layers.

Mobilenet: Mobile net is lightweight in its architecture and is specially designed to be very efficient during test time. Hence we can use this model for real -time distracted driver detection.

Resnet 50: Short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. The ResNet 152 model was the winner of the ImageNet challenge in 2015.ResNet 50 is the smaller version of ResNet 152.

Implementation of the CNN Model:

Data Augmentation:

Overfitting was the main problem we had faced. Image augmentation is utilised to create a larger dataset to avoid overfitting. We were successful in choosing very good parameters for Image Data Augmentation. We also verified that augmentation did not make much loss in the important features of the image and also that it creates realistic images of the driver. So this made our train dataset to expand artificially - this prevented overfitting to a large extent. We used real-time data augmentation while training the CNN so that we can save memory and also to ensure more randomness. We initially tried making augmented images and passing them to the CNN, we noticed that this caused overfitting whatever the size of the Augmented dataset(upper bound by memory availability) we take. But with

real-time data augmentation we get more or less - different Augmented images in every epoch of training the CNN. So this was our main step towards preventing overfitting. Data leakage is mitigated by splitting the people ID for train and validation set. Both these methods are done using the methods mentioned in the preprocessing column.

Original Image:

Augmented Images: (we tuned brightness_range, shear_range, width_shift_range,height_shift_range, zoom_range, rotation_range parameters to get a better model)

Train-validation split: We split the training data of 22424 images into 18732 training images dataset and 3692 validation images dataset ensuring no data leakage.

Training our models:

We used pretrained models VGG-16, RESNET-50, MobileNet with the weights of image net using real time data augmentation. Our single best performing model is MobileNet named model 10 with log loss 0.3407. We tweaked all the hyperparameters associated with the models, and we finally chose ten best models to perform ensembling.
Out of the ten - four are of VGG-16(With log losses of 0.36,0.41,0.43,0.45) , two ResNet50(With log losses of 0.4677,0.5) and four MobileNet models(With log losses of 0.39,0.37,0.3475,0.3407).
It is found that removing two of them improved the validation log loss and accuracy.
Using mean trimmed ensembling on these 8-models log loss decreased from a minimum of 0.3407 of each of the models to 0.22 .

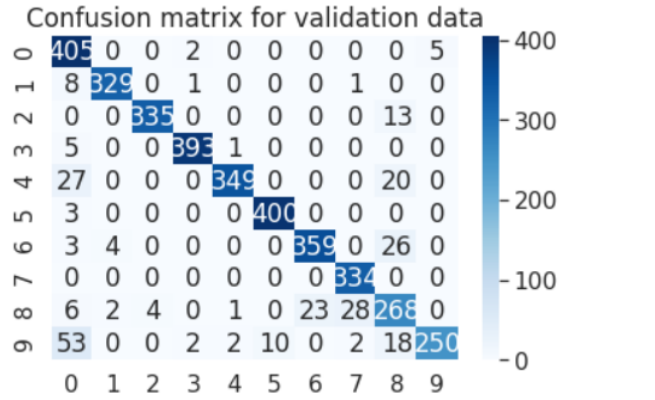
KNN:

As images are taken from a video clip, there exist many similar pictures which should belong to the same categories. So we find the nearest neighbours of an image and average their probabilities and assign it to the image for which neighbours are calculated.
Due to the huge size of test images we had to shrink the original test images to 40 ×30 to decrease the computational time. This increases the stability of the performance of the model and lowers the log loss.
We utilized KNN to further reduce it to log loss of 0.13(on validation data) with n_neighbours=10. The improvement of log

loss is expected as ensembling and KNN method reduces the variance of the model, thus preventing Overfitting.

Our Results on Validation Data for CNN:

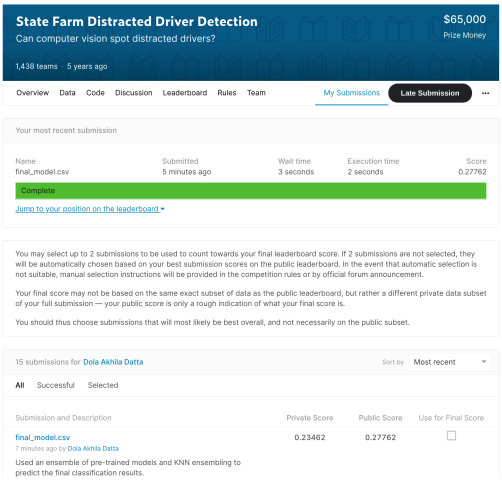
```
The accuracy of this model on validation set is: 0.92686
Log loss of this model is: 0.22665466303130505
```

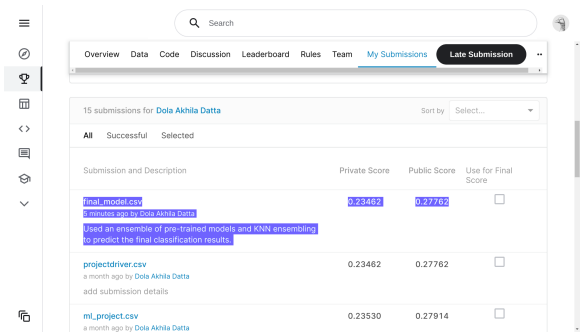



```
Class wise accuracies
Class 0 = 0.9830097087378641
Class 1 = 0.9705014749262537
Class 2 = 0.9626436781609196
Class 3 = 0.9849624060150376
Class 4 = 0.8813131313131313
Class 5 = 0.9925558312655087
Class 6 = 0.9158163265306123
Class 7 = 1.0
Class 8 = 0.8072289156626506
Class 9 = 0.7418397626112759
```

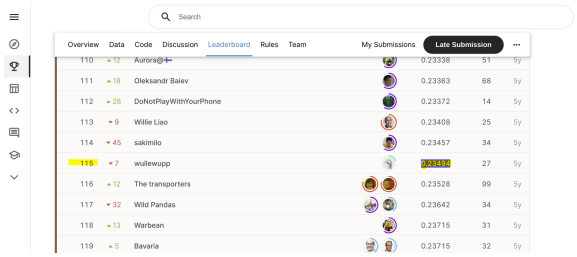
Our Results on Test Data submitted on kaggle: As the competition ended five years ago, we are just getting the score but that is not updated in the Leaderboard.

Log loss(Score) of 0.23462:





If we would have submitted during the competition Our rank would be 115.



But we are allowed a Late Submission on Kaggle. We got a log loss of **0.23462** after submitting our results on kaggle. If we would have submitted during the competition **our rank would be 115 according to our private score**.(Private Scores are considered to be the final rankings according to kaggle as they are evaluated over a larger dataset)

Real time distracted driver detection:

We have a pre-trained model MobileNet which is especially designed to be computationally efficient. We got an accuracy of 90% on the mobilenet model. Hence this model can be used for real-time detection of distracted drivers.

Conclusion

We got an accuracy of 92.68%(on validation data)and log loss of **0.23462**(on test data) by using CNNs, which outperformed both the SVM model(with accuracy of 62%) and naive bayes(with accuracy of 55%) model. SVM gives moderate accuracy and is better than Naive Bayes classifier for computer vision tasks.Highest accuracy is observed with CNN model so CNNs are still the state-of-the-art models for computer vision tasks.

Future Work

We really enjoyed the learning experience and knowledge that we gained. It is the first time for us to participate in a challenge on Kaggle and are satisfied by what we have achieved so far. Given more computing resources(more RAM) and time, we would have tried the following to improve our results:

1. With respect to CNNs - Due to limited availability of RAM we were restricted to resize the image to 128*128*3 which looks a bit blurred. If we could resize it to 224*224*3 .We would surely improve the log loss as more features will be captured.
2. With respect to CNNs - We would use Semi-supervised learning as there are 80k test images. Based on our prediction we can put a threshold on the predicted probability(say 0.99 or some value)and add those images with high predicted probability into the train data set and further train our model on an increased training dataset, which will likely improve our results.
3. With respect to support vector machines , as SVM cant do feature extraction unlike CNNs. Other feature descriptors like histogram of oriented gradients (HOG), VGG and pca for feature extraction could be used.

References:

[1] Wikipedia contributors. "Image moment." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 18 Jan. 2021. Web. 7 Jun. 2021.

[2] DataTurks: Data Annotations Made Super Easy. “Understanding SVMs’: For Image Classification.” *medium*, https://medium.com/@dataturks/understanding-svms-for-image-classification-cf4f01232700.

[3] Satya Naren Pachigolla. “Distracted Driver Detection using Deep Learning.” *towardsdatascience*, https://towardsdatascience.com/distracted-driver-detection-using-deep-learning-e893715e02a4

[4] Statefarm. “State Farm Distracted Driver Detection.” *Kaggle*, https://www.kaggle.com/c/state-farm-distracted-driver-detection.

[5] *keras*, http://www.keras.io/api/.