**State Farm Distracted Driver Detection**

Ghatikar, Mohit Ravi

Farheen, Nishaat

Zhang, Lanyin

Abstract — O**ne in five car accidents is caused by a distracted driver. This is an alarming statistic, even more so for car insurers. Profits would take a toll every time such a customer made a claim. State Farm is giving dashboard cams a try to check if it would help in these situations. If the footage can show that the driver was distracted at the time of an accident, State Farm could save money in those one in five cases. In this project, we will be detecting if a driver is distracted or not based on the images obtained from dashboard cams using machine learning algorithms.**

1. **Business Understanding**

Drivers are faced with a growing number of potential distractions behind the wheel – anything from pets, to passengers, phones, text messaging, entertainment systems and much more. There are three main types of distractions:

1. Visual – removes a driver’s eyes off the road
2. Manual – removes a driver’s hands off the steering wheel
3. Cognitive – removes a driver’s mind off the driving task

According to the CDC motor vehicle safety division, [one in five car accidents](http://www.cdc.gov/motorvehiclesafety/distracted_driving/) is caused by a distracted driver. Sadly, this translates to 425,000 people injured and 3,000 people killed by distracted driving every year.

[State Farm](https://www.statefarm.com/) hopes to improve these alarming statistics, and better insure their customers, by testing whether dashboard cameras can automatically detect drivers engaging in distracted behaviors. Given a dataset of 2D dashboard camera images, our challenge is to classify each driver's behaviour. Are they driving attentively, wearing their seatbelt, or taking a selfie with their friends in the backseat?

The idea of using dashboard cams in cars is still in its infancy. Once the data show that a particular safety device will save car insurance companies money if used by its customers, they begin to roll out discounts. Therefore, State Farm has a clear incentive to encourage their customers to use them. While merely owning a dash cam may not lower the [car insurance rates](http://www.insurance.com/auto-insurance.aspx), the footage that the camera provides may turn out invaluable in certain situations.

Some of its benefits are:

1. Having a record of your accident

Drivers often have completely different memories and descriptions of an accident. With video proof the driver can save themselves from being found at fault by a car insurance company. The dash cam footage can also expedite claims, as it may prevent the driver from having drawn-out discussions with insurance companies about who was at fault.

1. Getting out of a ticket

Moving traffic violations will normally incur points on the driver’s state driving record and raise the car insurance rates.  Not receiving a ticket, or beating a ticket in court due to video proof of no wrongdoing – will be helpful in avoiding auto insurance surcharge.

1. Helping fight insurance fraud

Catching a fraudster in a staged accident, can indirectly lower insurance rates. The FBI estimates that the cost of insurance fraud is more than $40 billion per year. When auto insurers unknowingly settle fraudulent claims, all motorists pay hundreds of dollars extra each year because companies divide the cost of claims among consumers.

1. **Data Understanding**

In this competition we are given driver images, each taken in a car with a driver doing something in the car (texting, eating, talking on the phone, makeup, reaching behind, etc). Our goal is to predict the likelihood of what the driver is doing in each picture.



Fig-1: Sample Image in the training set.

The 10 classes to predict are:

* 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

The dataset consisted of 22,000 training images and 79,000 images. Due to computational limitations, we used 8,000 training images to build the model and tested the model on 1,000 images. This competition ends on 1st August, 2016. We plan to use Amazon web services to train the model on all images and submit our predictions on Kaggle to obtain the rank.

1. **Data Preparation**

The most difficult task in any image classification problem is to extract meaningful features to feed into the machine learning algorithm. A feature is simply a number that we compute from an image. An image is represented by a large rectangular array of numbers (pixel values). We could try to use all of these numbers as features but this is not a very good idea because the relationship of each pixel (or even each small group of pixels) to the final result is very indirect. Plus it is computationally heavy. So, instead, we try to extract only meaningful features from every image and use those to represent them.

The first step before extracting features is to convert the images from RGB format to Grayscale format. This helps in speeding up the computations. A relatively recent development in the computer vision world has been the development of local-feature based methods. Local features are computed on a small region of the image. This technique is known as Speeded up robust features (SURF). These features are designed to be robust against rotational or illumination changes (that is, they only change their value slightly when illumination changes).

In order to decide where to compute these features, we used a technique called key-point detection which detects interesting areas of the image. This is relevant for our dataset since the model needs to be built only on certain parts of the image. For example, if a driver is texting with his/her right hand, the area of the image where the driver is texting is the most interesting part of the image.

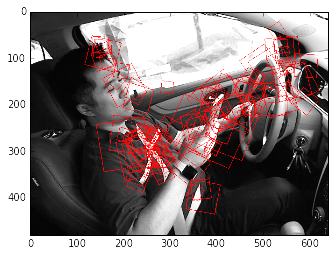


Figure 2 – Keypoint detection of a sample image from the training dataset.

After we detect the interesting areas of an image, we need to find a way to describe those areas in terms of numbers such that their spatial intensity patterns are recorded. This is known as a descriptor. We use SURF to get around 100 descriptors for each image. We cannot directly feed these descriptors to a support vector machine, logistic regression, or a similar classification system. In order to use the descriptors from the images, there are several solutions. We could just average them, but the results of doing so are not very good as they throw away all location specific information.

The solution we used is the visual bag of words model, which is a very recent idea. It was published in this form first in 2004. Similar to a bag of words model for text, we represent the image as a histogram of visual words. In this case, we cluster together similar looking regions and call them visual words. The number of words used does not usually have a big impact on the final performance of the algorithm. Naturally, if the number is extremely small (10 or 20, when we have a few thousand images), then the overall system will not perform well. Similarly, if we have too many words (many more than the number of images, for example), the system will also not perform well. As a rule of thumb, a value of 256 or 512 is a good trade-off for large images. In our case we chose 256.

In our final stage of data preparation, we performed k-means clustering on all the descriptors obtained from the training data set, with the value of k=256. Then every image is represented in terms of these clusters by counting if a descriptor falls under a cluster or not. Thus we have a visual bag of words model where an image is represented by a single array of numbers, of the same size. Therefore, we can start applying classification algorithms from the extracted features.

1. **Model Building**

We have tried out a number of algorithms such as logistic regression, random forest, and support vector machines on the features extracted from training images. These algorithms performed well giving accuracy rates of over 60% which is pretty good considering that classification by random chance would be 10% as there are 10 classes.

Logistic Regression

Logistic regression (LR) is a standard probabilistic classification model that has been extensively used across disciplines such as computer vision, marketing, etc.

We expect logistic regression to perform better as our data as it may have a lot of noise considering that we have extracted features from images.

Support Vector Machines

We used Multiclass SVM (one versus rest) to perform the classification as there are 10 classes. Surprisingly, SVM gave a really high accuracy rate. This could mean that the classes are really well separated and it was easier to define support vectors for them. Generally speaking, SVM usually does well for image classification.

Random Forest & Boosting

As logistic regression and SVM are basically binary classifiers, decision trees seem more suitable for multiclass problems. Simple decision trees gave a low accuracy, so we tried random forest and boosting trees. But as the number of observations is less than 10,000 we wouldn’t expect decision tree variants to outperform the linear models.

1. **Model Evaluation**

The following table shows the accuracy obtained on training and testing data for all the models that we built.

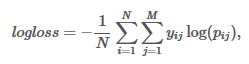
|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy on training data** | **Accuracy on**  **test data** |
| Logistic Regression | 67.4% | 58% |
| Support Vector Machines | 100% | 92% |
| Random Forest | 70.5% | 49% |
| Boosted Trees | 91.5% | 57% |

Boosted Trees seem to have overfit the data but this is as expected. But surprisingly, even random forest has overfit the data compared to logistic regression and SVM. This could be because decision trees are much more flexible than the others in terms of fitting the data.

To get the correct accuracy rate, we performed cross validation on the training features. SVM is still highly accurate.

|  |  |
| --- | --- |
| **Model** | **Cross Validation Accuracy** |
| Logistic Regression | 58.8% |
| Support Vector Machines | 88.9% |
| Random Forest | 49% |
| Boosted Trees | 57% |

Another method of comparing the models is using Log Loss. The formula for that is as shown below.

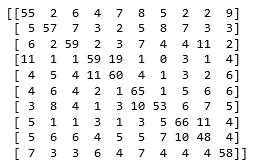


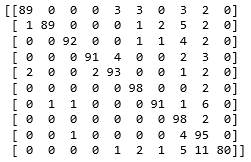
where N is the number of images in the test set, M is the number of image class labels,  *log* is the natural logarithm, *yij* is 1 if observation *i* belongs to class *j* and 0 otherwise, and *pij* is the predicted probability that observation *i* belongs to class *j*.

The lower the log loss the better. So, SVM has again performed the best out of all the algorithms.

|  |  |  |
| --- | --- | --- |
| **Model** | **Log Loss on training data** | **Log Loss on**  **test data** |
| Logistic Regression | 1.023 | 1.26 |
| Support Vector Machines | 0 | 0.22 |
| Random Forest | 1.5 | 1.8 |
| Boosted Trees | 0.9 | 1.49 |

The confusion matrix for logistic regression and SVM (in the same order) is shown below for comparison of their performances





Therefore, we chose SVM as the model that should be deployed for our web interface. This model was converted into pickle format so that it can be used by the web interface without having to run the model over and over again.

1. **Model Deployment**

We used Flask for deploying our model on a web application. Flask is a lightweight Python web framework. The demo page is written in HTML and CSS.

1. **Deep learning – An Alternative**

Deep learning is a branch of [machine learning](https://en.wikipedia.org/wiki/Machine_learning) based on a set of [algorithms](https://en.wikipedia.org/wiki/Algorithm) that attempt to model high-level abstractions in data by using multiple processing layers, with complex structures or otherwise, composed of multiple non-[linear transformations](https://en.wikipedia.org/wiki/Linear_transformation). It is being widely used in the industry for tasks such as speech and Image recognition, Natural language processing etc. Deep learning works better for image classification because they are able to cater to vast amount of variations in images, which are not possible with other algorithms. It works by consecutively modelling small pieces of informationand combining them deeper in a network.

Understanding the layers in the model is key. For example, in a sample image, the first layer will try to detect edges and form templates for edge detection. Then subsequent layers will try to combine them into simpler shapes and eventually into templates of different object positions, illumination, scales, etc. The final layers will match an input image with all the templates and the final prediction is like a weighted sum of all of them. So, they are able to model complex variations and behaviour giving highly accurate predictions.

Some of the disadvantages of deep learning are that it needs a large dataset to train, large computational power to train the model, it is very time intensive which makes it expensive for production and the parameters are hard to interpret.

**References:**

1. <http://www.analyticsvidhya.com/blog/2016/04/deep-learning-computer-vision-introduction-convolution-neural-networks/>
2. <http://www.insurance.com/auto-insurance/claims/7-reasons-to-use-a-dash-cam.html>
3. Building Machine Learning Systems with Python – Second Edition by Luis Pedro Coelho and Willi Richert.