Development of an Intelligent Decision Support System to Classify Unsafe Driving

Polytechnic University of Catalonia

MAI-IDSS

Practical Work 3

June 9, 2016

Luis Fábregues de los Santos

Cole MacLean

James Gray

Mihai Andrei

Contents

1.0 Description of the Domain of Application

This report outlines the development of an Intelligent Decision Support System (IDSS) to identify from images if the driver of a vehicle is performing an action that may be considered unsafe. An overview of the problem is presented, with details about the data structure and architecture, along with the exploratory, modelling and validation analysis required in building the IDSS. The final results are discussed and future work with potential improvements are recommended to further develop the model.

1.1 Kaggle

Kaggle is an online platform that connects companies with difficult Data Science problems with Data Science practitioners, in the form of competitions. Companies provide details, data and a scoring methodology that competitors use to build the best IDSS models to solve the specific data driven problem.

This project attempts to build an IDSS that is competitive with other models in the State Farm Distracted Driver Detection Kaggle competition.

1.2 State Farm Distracted Driver Detection Competition

This competition aims to accurately classify the behavior of vehicle drivers to help determine if their action might be considered unsafe. With the advent of in-car radios, fast-food and cellphones, distracted driving has become a common cause of car accidents and fatalities. The goal of this competition is to build a system that can highlight these distracted driving activities.

1.3 IDSS Application

An application of the proposed IDSS could be used by insurance companies to better assess accurate driving insurance premiums for safe and potentially unsafe drivers. Users could opt-in to a program where a camera is installed in the vehicle for a month that tracks the driving habits of the user and automatically detects and scores their driving behaviour. This technology could create safer roads by encouraging drivers to develop safe driving habits, decrease insurance premiums people with safe driving skills, and increase insurance companies confidence in the fees they should charge their clients.

1.4 Dataset

The State Farm Distracted Driver Detection Kaggle competition consists of 22,424 training images that have been sorted into 10 classes, and 79,726 unclassified images for testing. An example image is presented in Figure 1.

Figure 1: Example Training Image

The 10 classes to predict are:

1.5 Decisions

The State Farm Kaggle competition is a classical image classification Computer Vision challenge, where the IDSS, when given an image, is required to correctly identify the class to which it belongs. The main decision required by the IDSS is to determine the probability that an image belongs to each of the above classes.

1.6 Tools Used

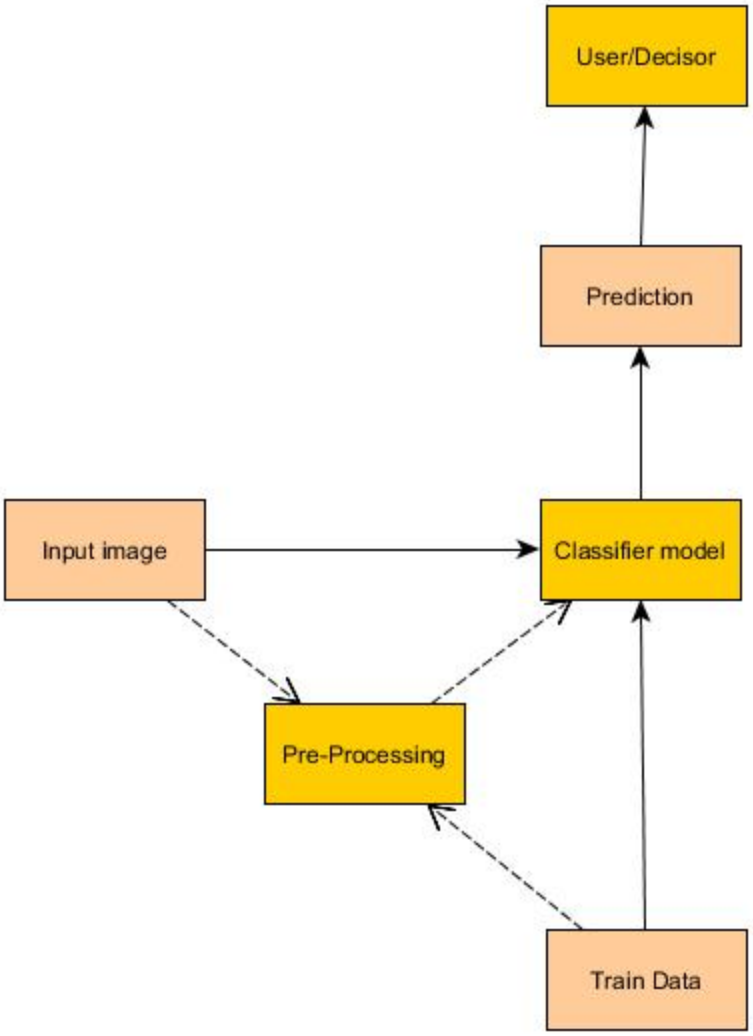
R - R is a programming language heavily utilized in statistical and data analysis applications. This tool was used in the Exploratory Data Analysis section of this report, to better visualize and understand the problem's dataset.

MatLab - Matlab is a numerical programming language that integrates, as part of its toolboxes, some already implemented and efficient algorithms. This is very useful, as most of the applicable supervised classification methods are available, optimized and with great documentation. Another reason to use this tool is that other subjects that include Machine Learning content encourage the use of it in the Master in Artificial Intelligence. The extensive practise with this software made us favour this option over others. A set of toolboxes have been employed for the development of this project:

- **Statistics and Machine Learning Toolbox:** It is an optional toolbox that comes with standard editions of MatLab and contains efficient implementations of the most relevant classification algorithm with very intuitive instructions to use. It also provides feature selection techniques, like PCA.
- **VLFeat[1]:** Is an open source library written in C that interfaces with MatLab for easy use. It implements common Computer Vision algorithms for classification, image understanding, feature extraction and classification.

Keras/Theano - Keras is a Python library built on top of the Deep Learning Theano package, that allows for a high level construction of deep neural networks by abstracting the creation of individual layers (Input, Filter, Convolutional, Output) as pythonic objects with the ability to stack them together.

2.0 Functional Architecture of the IDSS Prototype



3.0 Data Pre-Processing

As MatLab is the principal tool employed for this problem, we faced some problems importing the dataset. The set of images is large (>24,000 images)and most of the already implemented functions in this tool require the data to be in double format. The images have a dimensionality of 640x480 RGB pixels, that makes the whole dataset fill a space of 20 thousand million doubles.

3.1 Grayscaling

Obviously, that much data does not fit in any of the available systems of the members of the team. The first approach was reducing the quantity of the data by converting the images to grayscale, which reduces the spatial needs to a third of the original size. This decision makes sense in the context of the problem as the decisions are performed observing the position of the driver and is not related to any color information.

However, that was not enough, as the dataset still was too large to fit in memory. The team then pondered about various ways to still use the dataset. Dividing it by clusters or classes seem an understandable decision, however most of the methods existing in MatLab do not give support to a divided dataset to do their main functionalities.

We thought that reducing the number of samples was not a good option, as most of the classifiers planned to be tested are very data dependant. Furthermore, without information of the relevant samples, we could be reducing our resulting accuracies by a dramatic margin if we had ignored samples using a random policy.

3.2 Image size reduction

The selected approach for reducing the spatial consumption of the dataset was dimensionality reduction. As the given inputs are images, reducing them does not necessarily delete important features and can even help the classification method to ignore most of the noise. This is a common practise in the Computer Vision branch of AI and, as the results will show, is a very effective approach.

In order to further reduce the dataset dimensionality and prevent noise (which in image data is usually high) some of our methods employed PCA technique[2] to create a set of linearly uncorrelated variables. This usually helps the algorithms computational time and space requirements, as the new set of variables is always smaller, and can improve accuracies in very noisy environments.

4.0 Exploratory Data Analysis

The high dimensionality of image processing and Computer Vision problems make the application of traditional Exploratory Data Analysis (EDA) and Feature Engineer (FE) difficult, but there are still important attributes of the dataset to explore.

4.1 Distribution of Class Training Image Counts

Table 1 outlines the distribution of counts of available training images, to ensure that an even distribution of each class exists within the training set.

Table 1- Distribution of Class Training Image Counts

Class ID	Class	Training Image Count
C0	normal driving	2489
C1	texting - right	2267
C2	talking on the phone - right	2317
C3	texting - left	2346
C4	talking on the phone - left	2326
C5	operating the radio	2312
C6	drinking	2325
C7	reaching behind	2002
C8	hair and makeup	1911
C9	talking to passenger	2129

The classes seem to be fairly consistently represented, with class 0 (Safe Driving) being the most represented and class 8 (Hair and makeup) being the least represented. This makes safe driving 30%

more represented than Hair and Makeup, but most classes have close to 2000 sample images. The effects of these different training sample sizes should be considered when developing the IDSS model.

4.2 Distribution of Class Training Subject Counts

Each image in the training set has been labelled with the "subject" or individual person observed in the image. Figure 2 Visualizes below a stacked barchart for the class representation of each individual.

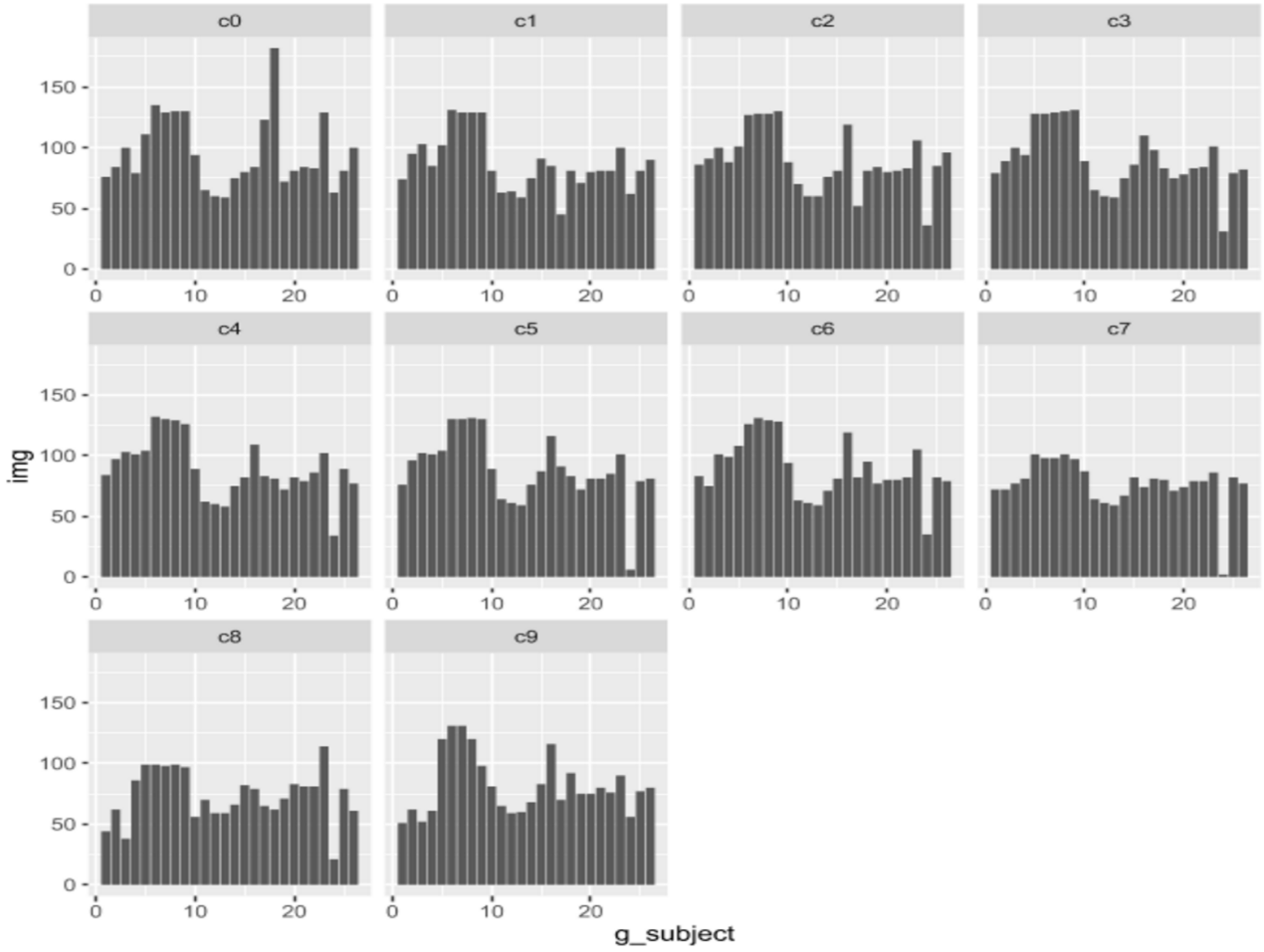


Figure 2 - Distributions of Class Training Subject Counts

The distribution of individuals in each class are fairly similar, but there do exist obvious differences. Some easily identifiable outliers include subject 16 in class 0 (safe driving) being over represented, or subject 26 being largely underrepresented in many categories, such as C5 (operating the radio) and C7 (reaching behind). It is important to review these outliers in analyzing the performance of a model, as it may be possible to develop models that overfit to the above distributions, but do not generalize to real world distribution of these classes. For example, it could be possible to learn attributes about subject 26 and use those attributes as indication that the probability of being in classes 5 or 7 are very low. But because the purpose of our model is to generalize to the larger public, using these attributes specific to subject 26 may incorrectly categorize subjects with similar attributes, but do belong in classes 5 or 7.

4.3 Distribution of Class Training Subject Counts

A major issue with this dataset is that each for each driver, there are many consecutive images sliced out of a video of the particular action. This means that if the dataset is split naively into training and test sets, the test set will, in effect, be a fuzzy subset of the training set. This property of the data led to us seeing implausibly good performance in our initial tests, e.g. k nearest neighbors on raw pixels having an accuracy of >99%.

5.0 Data Driven Models

5.1 Linear and Quadratic Discriminant Analysis classifier

The linear discriminant analysis (LDA) is a method that is used to find in a given dataset a linear combination of features that can characterize or separate between given classes between points in that dataset. This linear combination can be used to reduce the number of featured in the data, but also can be used to classify the data.

LDA assumes that, for each class, the data is generated following some normal distribution. LDA also assumes that the covariance of each of the classes is the same. Thus LDA can be used to find a space that appears to contain all of the class variability.

Quadratic linear analysis (QDA) is similar to LDA with the main difference being that it does not assume that the covariance between classes is identical. Given this set of constraints likelihood ratio can be used to create a separating surface between the classes. It can be shown that this surface is quadratic, hence the name quadratic discriminant analysis.

Since both methods assume that the classes generate the data by a normal distribution in order to create a classifier the fitting function estimates the parameters of a Gaussian distribution for each class.

For our IDSS, using simply these methods with the pca processed pixel data of the images did not give good results.

While splitting the data randomly, for cross validation, the results were quite good, both QDA and LDA performed with accuracy of 0.98 and a logloss of 0.41 LDA and 0.37 QDA. These are extremely good results, but it was obvious that the methods were overfitting. The problem was that we were using the same subject in training images as well as testing images. As soon as we changed the sampling to partition the dataset by subject, the accuracies dropped significantly. LDA got 0.35 accuracy and 22.5 logloss and QDA 0.18 accuracy and 28.3 logloss. Which are terrible results. It is interesting to note that, with the change of sampling QDA performed worse than LDA.

5.2 Binary Decision Trees

Binary decision tree is a well-known classification method that builds a tree from the training data that has at every node the value of some feature and the leafs are possible classes. It splits data based on the discriminative power of that feature (and that value). One of the best point about decision trees is that they are quite fast to train and predict.

The biggest drawback of decision trees is the fact that they are prone to overfitting. This can be seen in the results of our tests. We have used a simple binary decision tree, that is implemented in matlab, using the default parameter values.

Using simple pca pre-processed pixel data, the method did not perform too well. Even with random sampling the binary decision trees performed quite poorly, having an accuracy of 0.79 and a logloss of 7. With sampling by subject the performance dropped to 0.17 accuracy and 28.1 logloss (which is slightly better than QDA).

5.3 Naïve Bayes Classifier

Naïve Bayes classification training means estimating a probability distribution of data into classes. One of the keypoints of this estimation is that the naïve bayes assumes that the features are independent given the class. While this is usually not true in practice, it the method still appears to work well, particularly with datasets with many features. For testing the method computes the posterior probability of the belonging to each class. Then the sample is classified according to the largest posterior probability.

The method can use different probability distributions. We have used the default Matlab distribution which is the kernel distribution. It is good for features with continuous distribution, but is not strict, it can be used even if the distribution of some feature is skewed or has peaks or modes.

Yet, the Naïve Bayes classifier had a very bad result on our IDSS using simple pca pre-processd pixel data. For the random sampling it got an accuracy of 0.53, which is the lowest accuracy we observed for this test. The logloss was, accordingly, 15. On the sampling by subject test it got an accuracy of 0.09 and logloss of 26.9. Which is extremely bad.

5.4 Nearest Neighbors and Probabilistic nearest neighbors

Nearest Neighbors treats each instance as a vector, and calculates the

Nearest Neighbor classification is one of the oldest and simplest classification methods. It is a type of lazy learning, where the computational burden is left to the time of prediction. Nearest neighbor works by computing the distance between the sample we wish to predict the class of and the training points. The points with the smallest distance to the sample (neighbors) are retrieved and the class of the sample is decided by aggregating the classes of the neighbors – usually choosing the majority class. There can be different algorithms to break ties.

We have used the default parameters values for the algorithm implemented in Matlab – k is equal to 1, the distance metric is the Euclidian distance and the tiebreaker is the class of the closest neighbor from the tied classes.

Using pca pre-processed pixel data, which are not invariant to rotation, scale, and angle, it did not give very good results for classification of images. For random sampling it gave very good results, 0.99 accuracy and 0.14 logloss. But it was obviously heavily overfitting, and the results for the sampling by subject are 0.39 accuracy and 20 logloss. These are better results than the previous methods, yet are bad enough not to be significant.

The idea for the probabilistic nearest neighbors came from the fact that the official score for the kaggle competition is computed with the logloss metric. This metric is reliant on the fact that the predictions

are probabilities of what class the tested points are. The original knn is giving only a class not several probabilities, and this is a great disadvantage in the logloss metric. So, the idea was to transform the knn algorithm into a nearest neighbor algorithm that gives probabilities for every class. Intensive research on this area has been done before including methods that are more or less complex. In [8] there is a comparison of several such methods, and the conclusion is that probabilistic methods are not necessarily better than simple knn, nor the other way around. Despite this, we decided to try our version of a simple probabilistic knn.

The algorithm is very simple: 51 neighbors are retrieved and the probability associated with every class is the percent of neighbors that are from that class in the 50 retrieved neighbors. Using this algorithm, the predictions of the knn are probabilities and, as expected, the logloss was improved. But the result was extremely surprising. For sampling by subject, the logloss was 8.23 but the accuracy was 0.07. The accuracy is extremely low, which means that the pknn was not performing almost at all. But the logloss is significantly better than for the previous methods. Still, logloss of 8.23 is insignificant.

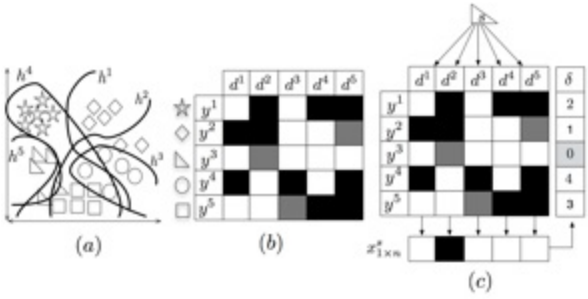
Of course, the high number of neighbors might be the cause of the loss in accuracy, as usually, for knn lower number of neighbors (5, 7, 13) give a better result, but testing with 25 and 7 neighbors did not improve accuracy, on the contrary, the accuracy dropped to 0.06.

5.5 Support Vector Machine

Support Vector Machine is a binary classification method which is based on finding a hyperplane that completely divides the data in two spaces. Once this hyperplane is found, the classification can be done by deciding in which side of the hyperplane the new data point is. This algorithm has the problem that it can not create a dividing hyperplane if the data is not separable. To be able to use this method with a not separable dataset, a margin is added. This margin allows samples to be in the side of the hyperplane that does not belong to its class. Adding this margin, the algorithm will try to find the best tradeoff between finding the hyperplane that better classifies the data and the margin that covers most of the wrongly classified samples.

5.6 Error Correcting Output Codes

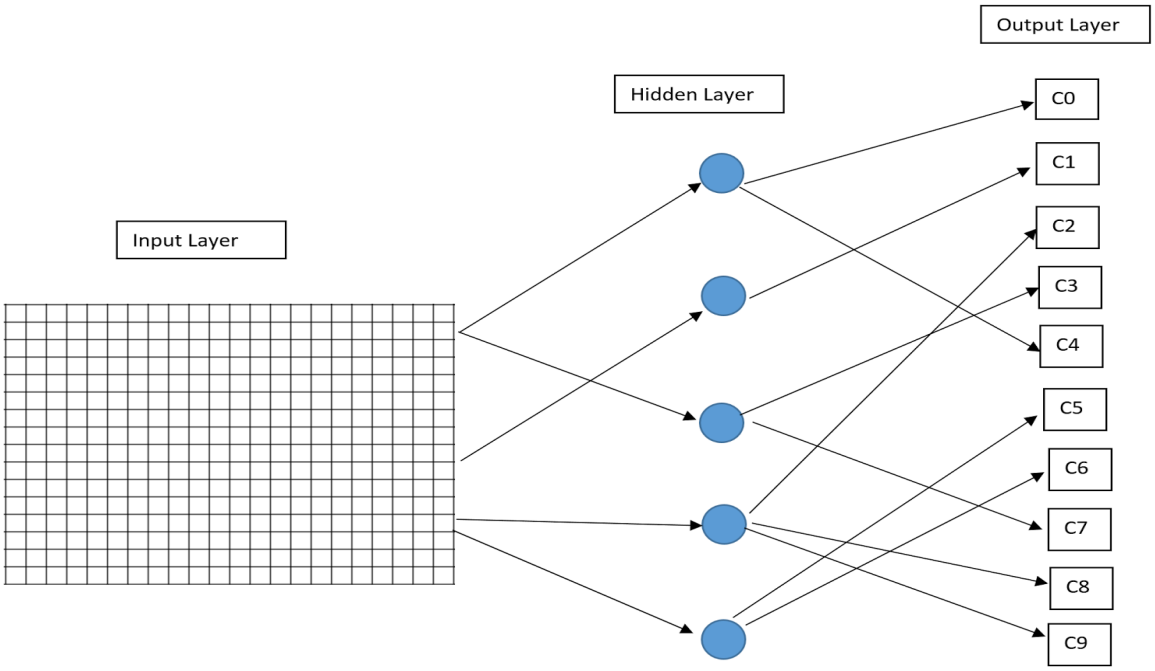
A major limitation of many classification algorithms is that they are binary separators, e.g. SVM. Since this classification task is multi-class, something extra is needed if we want to use such algorithms. One such method is Error Correcting Output Codes, which is used to interpret the output of an ensemble of binary classifiers. [7]



ECOCs work by training n binary classifiers, then each class is assigned a codeword which is a bit array of length n, where the ith element corresponds to the binary class decided by the ith classifier. When using an ECOC to decide an unseen instance, the instance is classified {0,1} by each n classifiers, and then the output code is compared to the codeword for each class. Given a distance metric with the midpoint defined for all points, it is easy to see that ECOCs can correct for at least (d-1)/2 individual classification errors where d is the distance between the two nearest codewords.

5.7 Convolutional Neural Network

The Neural Network developed for this IDSS is a Sequential model, where the results of each previous layer are computed and fed sequentially into the next layer. A baseline Sequential Neural Network for the State Farm competition can be found here. For most image classification Neural Nets, there is a minimum of 3 layers required - An input layer, a hidden Convolutional Network layer and a final output layer with dimensionality equal to the number of classes in the classification problem (in this case 10). The following figure depicts the minimum architecture for a Neural Network IDSS for the State Farm classification competition.



Using this baseline Neural Network with 3 layers and 2-K-Fold cross-validation, the logloss for this model is 1.19. To improve the score, more layers can be added to the network. Although neural networks can be difficult to describe, there are some options we can conceptualize and tune. The following table summarizes the options used for the final Neural Network IDSS, used to tweak the keras script from the linked baseline github codebase.

Layer Description	**Layer Type**	**Layer Options**
Input Layer - Initial data loading and parsing layer	Convolution2D	Kernal Size = 2 Filters = 32 Input shape = 96x128 Activation = ReLu
1st Hidden Layer - Macro level feature learning	Convolution2D	Kernal Size = 2 Filters = 32 Activation = ReLu

2nd Hidden Layer - Reduce dimensionality once macro features are learned	MaxPooling2D	pool_size = 2
3rd Hidden Layer - Dropout layer to prevent overfitting	Dropout	Dropuot_rate = 25%
4th Hidden Layer - Micro level feature learning	Convolution2D	Kernal Size = 2 Filters = 32 Activation = ReLu
5th Hidden Layer - Reduce dimensionality once micro features are learned	MaxPooling2D	pool_size = 2
6th Hidden Layer - Dropout layer to prevent overfitting	Dropout	Dropout_rate = 50%
Output Layer - Reshape dimensions to final classes	Dense	Classes = 10 Activation = softmax
Training Options	NN Training	Batchsize = 64 Epochs = 3 K-Folds = 10

The final logloss score for this model using 10-fold K-fold cross-validation is 0.8, a very impressive score, but likely suffers from some overfitting.

5.8 Classifying by Visual Words and Spatial Histograms

Classifying by means of Visual Words is a complex process. The image is first divided into dense keypoints [3], which describe points of the image by means of analysing patches of the image with a fixed size and frequency over the image. Those patches are analysed by means of SIFT descriptors [4] which describe the patch computing visual invariant features, position, size, etc. The different descriptors for each patch are stored in a database called 'Vocabulary'. After this data is obtained, for each training class, the descriptors from the 'Vocabulary' (called Visual Words) are located in the image. With the amount of Visual Words of the image, an histogram is build. This histogram describes the image. A more advanced technique involves a previous tiling of the image, effectively dividing it into different sections. After that, an histogram of each section is obtained, which better describes the image but is more computationally expensive. This procedure can be seen as a complex way to perform a very informative data transformation of the input images.

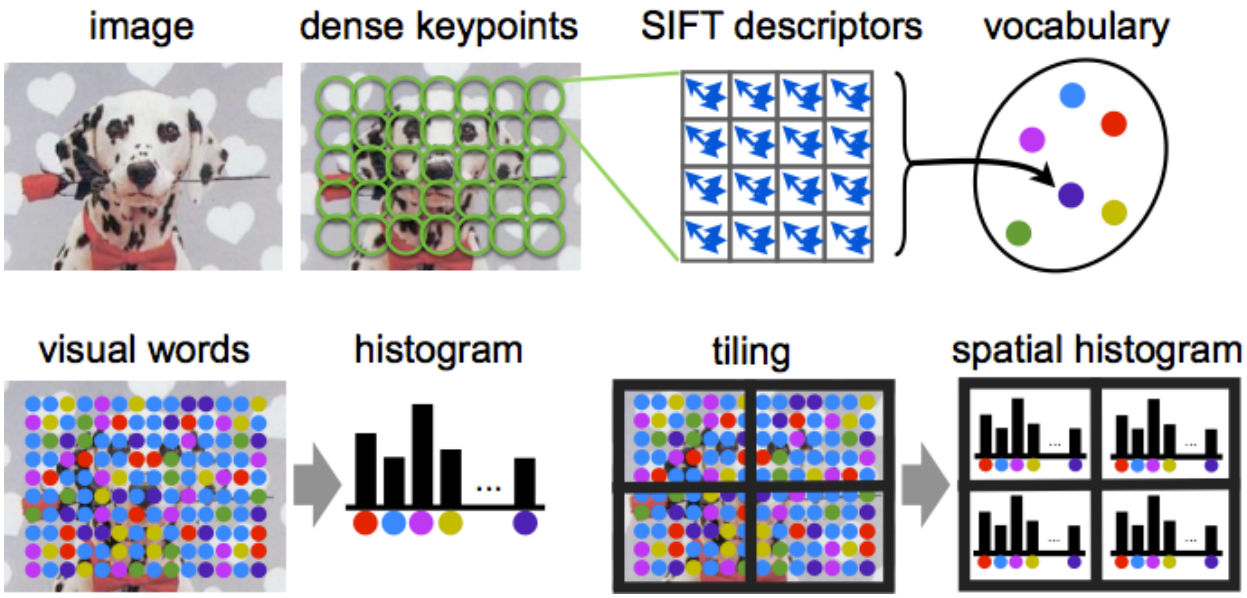


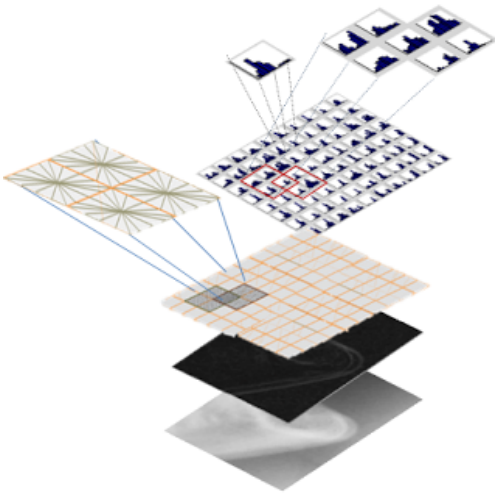
Figure 3 - Classification using Visual Words and Spatial Histograms. [5]

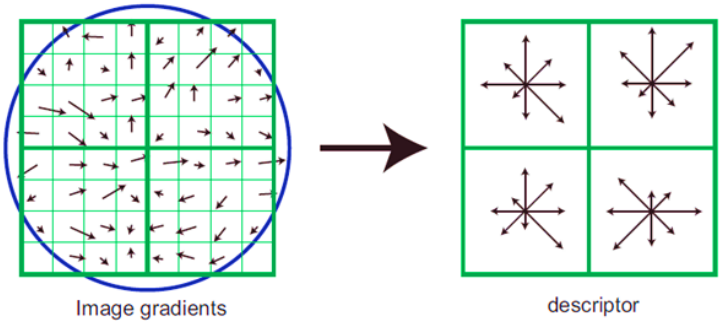
Once that process is over, the system has a description of each of the images based on spatial histograms of visual words. This information can be used to classify samples. In the context of this problem, the samples are classified, which means that the data of the histograms can be used as training samples of any supervised classification algorithm. After the system is build, the test samples should be also described using spatial histograms with the same vocabulary and the same tiling.

The selected algorithm for doing the classification procedure was Support Vector Machines. Most of the examples found in the literature used this technique, for its training a testing speeds and also for its low spatial consumption (very useful in massive datasets like this one). The biggest problem is that the project uses a Multi-Class dataset. In order to use this technique, a One vs All version of the SVM model was proposed and implemented.

5.9 Histogram of Oriented Gradients

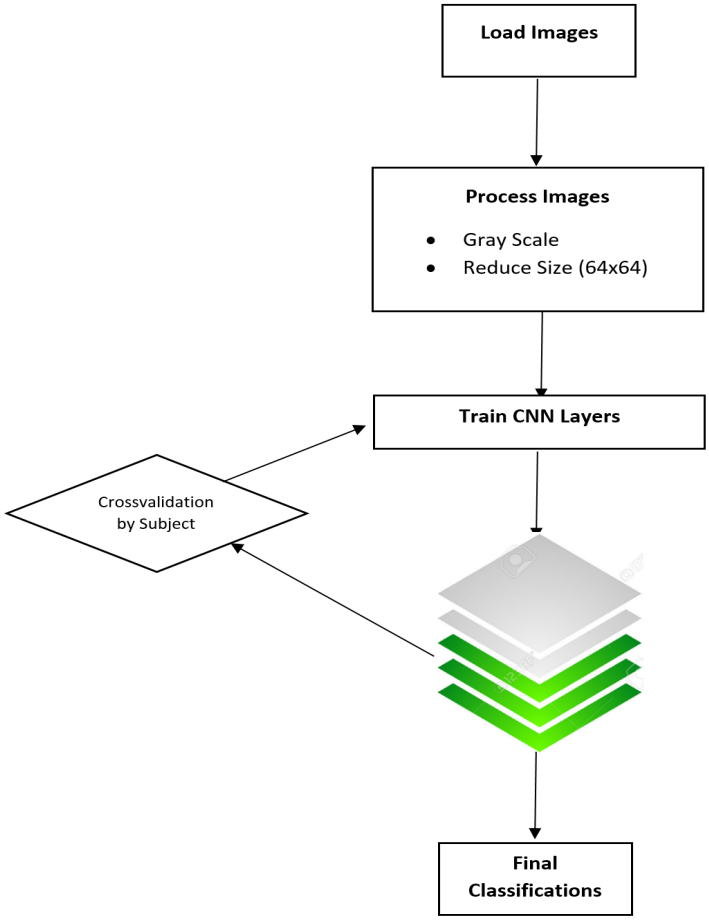
Histogram of oriented gradients is a very popular feature representation for object recognition, as it provides both geometric and photometric invariance [6]. The HOG algorithm computes the strength of all the gradients in each cell of an image, and then places the computed gradients into a histogram based on direction and magnitude. These





features can then be used with any machine learning algorithm, although SVM is the standard.

6.0 Flowchart of the Data Driven Model



7.0 Model Post-Processing and Validation

7.1 SVM Visual Words and Spatial Histograms

The tests over this algorithm were promising using the binary classifier (class 0 versus class 1) as we got 98% test score using data division by drivers. Although the SVM is a very fast method to train and test, obtaining the vocabulary and the transformation of the images to spatial histograms is a remarkably long process. Those results were obtained after some hours of computation.

After this promising result, we implemented a multiclass, one versus all version of the SVM classifier. The training time increased a lot due to the among of images we had to process. Also, the data structures created also consumed a high amount of space. We were planning to do a 5-fold cross validation to obtain statistical significant results, however that would have taken 40+ hours of computation, a time that we were not able to spend. The final results of this method applied only to one fold was a 77% if accuracy over the test set.

We also tried to compute our results over the test set provided by Kaggle with a partial version of our system to observe the generalisation capabilities and compare them with our other methods. Although we had the ten support vector machines trained from previous tests, extracting the histograms of the test data would have taken more than 40 hours, as the test set is also massive. This last part was dropped by lack of computational time.

8.0 Final Results and Conclusions

Multi-class

	LDAC	*Naive Bayes*	*Decision Tree*	*kNN*	*SVM+ECOC*
Raw Pixels	44.6 (35.1)	36.2 (29.2)	33.9 (26.2)	**45.3 (34.1)**	43.0 (34.0)
Raw Pixels + PCA	**48.3 (35.9)**	11.1 (9.4)	23.3 (17.6)	46.2 (39.9)	40.9 (34.0)
Hog features	64.0 (54.0)	56.5 (45.6)	31.4 (26.8)	57.5 (51.0)	**66.0 (54.3)**
Hog features + PCA	63.8 (55.6)	0.12 (0.09)	33.8 (29.7)	63.7 (54.3)	**71.1 (60.2)**

Class 0 vs Rest

	LDAC	*Naive Bayes*	*Decision Tree*	*kNN*	*SVM+ECOC*	*SVM*
Hog features	89.2 (85.9)	89.0 (82.1)	83.1 (79.3)	**91.6 (89.1)**	90.1 (86.9)	90.1 (86.9)
Hog features + PCA	**91.5 (87.9)**	89.8 (88.9)	79.6 (71.7)	**91.1 (89.2)**	90.8 (87.6)	**91.3 (86.4)**

Fig. Max Accuracy (Avg Accuracy) over 10 fold cross-validation

Model	**CV Logloss Score**
LDA	22.5
QDA	28.3
Decision Tree	28.1
Naive Bayes	26.9
k-Nearest Neighbors	8.23
HOG Features - SVM + ECOC - Multi-class	9.9885
HOG Features - kNN - Zero vs Rest	3.0886
8-Layer Convolutional Neural Network	**0.8**

The final validation scores show that the Convolutional Neural Network provides the best results of any Data Driven Model. This result is not unexpected, as many of the cutting edge Computer Vision

applications are utilizing CNN's with record breaking success. Although it was expected that CNNs would outperform the other models, inducing various descriptive, generative and discriminative models provided the team with deep understanding of the advantages and disadvantages of various models, along with developing an intuitive understanding of the problem and dataset.

The CNN model was utilized as the team's submission for the Kaggle competition, and obtained a logloss score of 1.79, more the double the training set crossvalidation logloss. This indicates that the model suffers a substantial amount of overfitting, utilizing details of the driving subjects in the training set that do not exist in the test set. To improve this model, a review of the characteristics that are being overfitted is required, and an iterative design approach to incrementally improve the final results to eventually develop a system capable with higher predictive capabilities.

The final model was number 715 out of 975 entered teams, but was able to beat the base submission benchmark score of 2.3. Although the model surpasses the benchmark, further work is required to build a system of acceptable accuracy to be used in the proposed application of quantifying driver behaviour for use in insurance premium recommendations.

9.0 Future Work

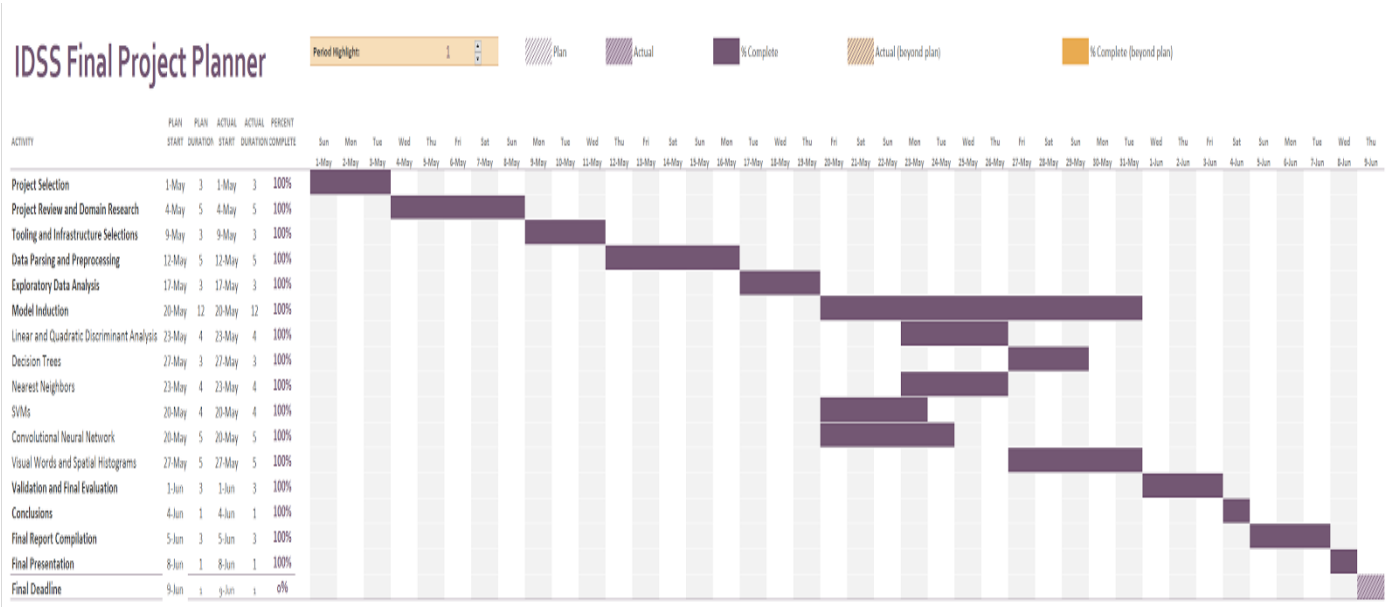
The algorithm using visual words and spatial histograms is a very interesting approach, however we feel that using a SVM classifier is not the proper way to treat the samples. The new feature space makes them a good target for better algorithms for multiclass classification and approaches that requires lots of data, like neural networks. The system will take long to build, as the computations of the histograms and the training of the net are long processes, however we think that it will achieve better results.

10.0 Task Assignment and Responsibilities

Task	**Description**	**Responsible**
Experiment with simple methods	Try some simple methods like KNN and LDAC with simple preprocessing like PCA	Luis and Andrei
Create logloss function	Create a logloss function as described in the kaggle competition	Andrei
Debugging implausible results	Figure out why we were getting 99% accuracy with incredibly simple methods	James and Luis
Probabilistic KNN	Experiment with a modified version of KNN that gives probabilities of being in a class	Andrei

Experiment with HOG features	Use more complex features to better explain data	James
Experiment with multiclass learning for binary classifiers	How can we use SVM in a multiclass setting?	James
Base code for loading data	Loading, grayscaling, resizing and creating numeric datasets	Luis
Visual words and spatial histograms	Represent the images via visual words and classify them using spatial histograms as features	Luis
Describe Domain of Application	Detail introduction section including application domain, kaggle summary, dataset summary, and decisions	Cole
Exploratory Data Analysis	Prepare EDA to review dataset and gain insights to be used in models	Cole
Convolutional Neural Network	Develop and analyse the CNN model	Cole
Final Report and Presentation Compilation	Document, report and present final results and methodologies	All

11.0 Gantt Diagram of Tasks Planning



References

- [1]: VLFeat: http://www.vlfeat.org/
- [2]: Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space" (PDF). *Philosophical Magazine* **2** (11): 559–572. doi:10.1080/14786440109462720.
- [3]: T. Tuytelaars, ***"Dense Interest Points",*** IEEE Conference on Computer Vision and Pattern Recognition 2010, pp. 2281-2288, pdf
- [4]: Lowe, David G. (1999). "Object recognition from local scale-invariant features". *Proceedings of the International Conference on Computer Vision*. pp. 1150–1157. doi:10.1109/ICCV.1999.790410.
- [5] http://www.di.ens.fr/willow/events/cvml2011/materials/practical-classification/
- [6] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection." In CVPR, pages I: 886-893, 2005.
- [7]Thomas Dietterich, Ghulum Bakiri. "Solving Multiclass Learning Problems via Error-Correcting Output Codes" In JAIR, pages 2: 263-286, 1995.
- [8] S. Manocha and M.A. Girolami. "An empirical analysis of the probabilistic K nearest neighbour classifier" in Pattern Recognition Letters, volume: 28 (2007), pages: 1818–1824.