CSC2515 Project - Road Prediction

Kevin Murray - 996088175
Leslie Barron - 997577625
April 17th, 2015

1. Intro

Self-driving cars have historically been considered a fantasy, however today significant effort being applied in both academic and commercial research to make them a reality. While there are many challenges to overcome in the quest to create a viable self-driving car, one of the most important is making the car 'aware' of its surroundings. This has historically been tackled using advanced ranging systems such as LIDAR/RADAR. However such systems are bulky and expensive. Using camera systems is much more cost effective, but requires significant processing to extract usable information. Designing such extraction algorithms by hand is very difficult and error prone. Using machine learning techniques to perform this task is likely to be more effective.

For our project we have focused on the problem of detecting the road (or lane) from an image produced by an on vehicle camera. This can be formulated as a binary classification problem that needs to be performed once per-pixel in the image. One class designates pixels that corresponds to the road/lane, and the other class corresponds to non-road/lane pixels. Unlike many traditionally classification problems where a set of input features are used to predict a single output class, this is a multi-output classification problem, since we need to produce multiple output classifications (one for each pixel in the image) for every input image.

To evaluate the efficacy of the techniques we investigate we have evaluated our techniques using the 'Road' component of the KITTI Vision Benchmark Suite [1]. This benchmark suite consists of 384 training images split into three datasets corresponding to common street types: Urban Unmarked (UU), Urban Marked (UM) and Urban Marked Multi-lane (UMM). We refer to the full KITTI data set (UU, UM and UMM) as 'All'. For evaluation these datasets are split into train (60%), validation (10%) and test (30%) sets.

This report is organized as follows. Section 2 describes the pre-processing techniques we investigated to reduce the dimensionality of the dataset to make the problem computationally tractable. Section 3 describes the various types of classifiers we evaluated including the important hyper-parameters and how they were tuned. Section 4 presents the experimental results and discusses the efficacy of the various techniques. Finally, Section 5 presents the conclusion and future work.

2. Pre-processing

A major challenge for any computer-vision type application is handling the large amounts of data found in images. For example, on the KITTI benchmarks the input images have a nominal resolution of 1242x375 pixels with three colour channels. This corresponds to an input feature with nearly 1.4 million dimensions, far too large for most machine learning algorithms. As a result some form of dimensionality reduction must be performed to make this problem computationally tractable.

We investigated two approaches for performing dimensionality reduction: using super-pixels produced by the SILCO algorithm [2], and projecting the data into a lower dimensional space using Principal Component Analysis (PCA).

Before discussing these methods in detail it is important to point out that the KITTI dataset is not entirely consistent, with some images being slightly different resolution than others. This is undesirable since it means some input images have a different set of features. In order to put the data into a uniform features space images were resized to 1242x375 using a high quality resampling filter from the Python Image Library (PIL) [3].

2.1 Dimensionality Reduction using Super-Pixels

Our initial approach to dimension reduction was to perform road prediction on the super-pixels produced by the SILCO algorithm. Under this approach SILCO selects a subset of the raw input pixels in an image as being representative examples of the pixels surrounding them. This serves to eliminate redundant information (many closely adjacent pixels have the same colour providing little new information) and reduce the dimensionality of the input.

While we were able to run SILCO on the KITTI data to generate superpixels we encountered several challenges with this approach. Firstly, although SILCO takes as a parameter the number of super-pixels to produce this constraint is not strictly respected, with some images having fewer super-pixels than requested and some having more. While the number of generated super-pixels was generally close to the number requested, this caused issues with algorithms that expect the input data to have a fixed dimensionality. The second, and more concerning issue was the choice and consistency of the super-pixels generated by SILCO. Since SILCO performs its super pixel selection on a per-image-basis it produces different super pixels optimized for each image. This is problematic since it means the input features fed to a machine learning algorithm differ for every input in the training set - making it unclear what information is actually being presented to the algorithm to learn.

As a result of these issues we did not pursue using SILCO and the super-pixel approach any further, instead choosing to investigate other techniques.

2.2 Dimensionality Reduction using PCA

PCA is a commonly used approach for dimensionality reduction, that projects a high dimensional data set into a new lower dimension space. PCA performs this by projecting the data along the directions that capture the most variance in the data.

To perform dimension reduction on the KITTI dataset we made use of the PCA module provided by Scikit-Learn [4], a Python based machine learning framework. One of the key parameters for PCA is choosing the number of components (i.e. the dimension of the output space). To determine the number of components we evaluated how much of the original input data's variation was captured in the reduced dimension space.

For individual categories in the KITTI dataset only a small number of components were required to capture a large portion of the variation. For example on the UM dataset only 50 components were required to capture 95% of the training set variation. As expected, more components were required to capture the variation of the combined (i.e. All of UU, UM, and

UMM) dataset. However the number of components required remained modest, only 200 components were required to capture 100% of the variation in the training data.

For consistency across the data sets (UU, UM, UMM, All), we held the number of components constant (at 200 components) when evaluating our classifiers. This ensured that all input data variation was captured regardless of what dataset was being evaluated.

The principle components can also be visualized, producing 'Eigenroads' that capture

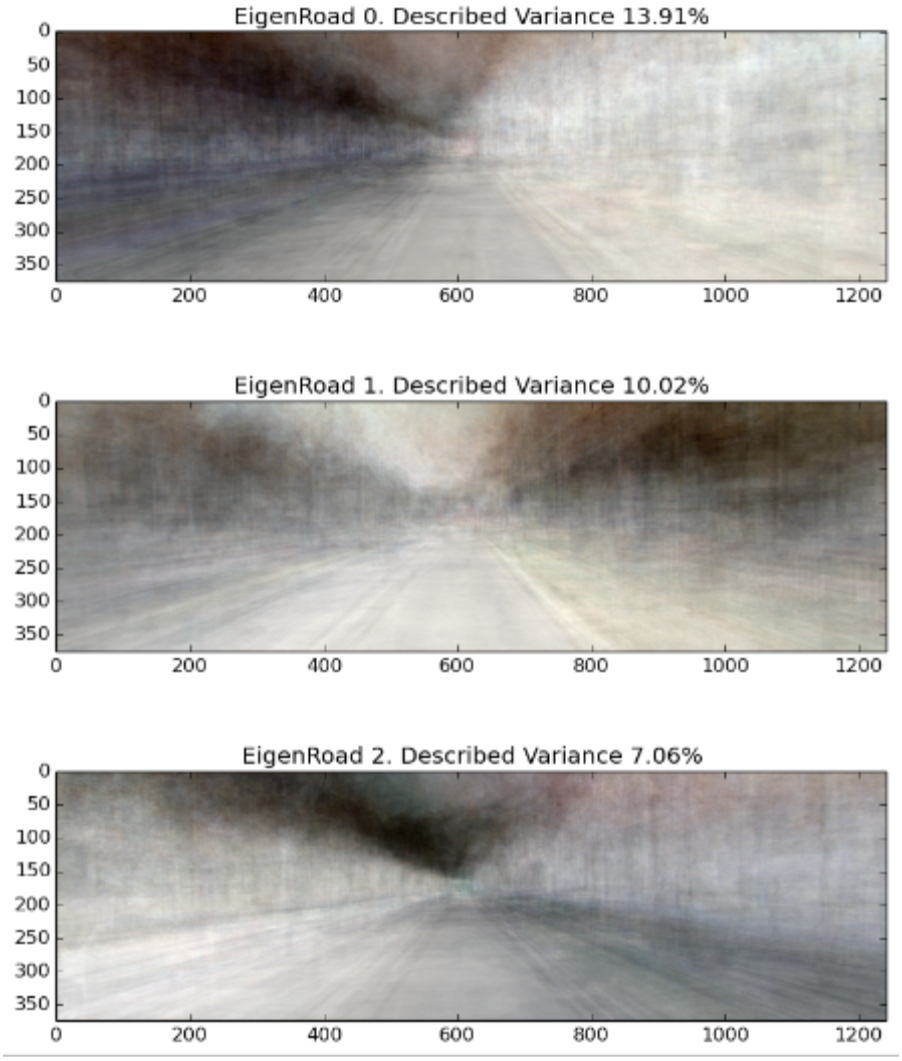

Figure 1: Top 3 eigenroads

the key characteristics of the input data. The top 3 eigenroads for the 'All' dataset are shown in Figure 1. The dark regions in the upper part of the eigenroads likely correspond to the positions of trees located along the sides of the road. Various lane positions and markings can also be seen in the lower parts of the eigenroads.

3. Classifiers

We trained several different types of classifiers on the road identification problem including K-Nearest Neighbours (K-NN), a Decision Tree, an Ensemble Random Forest, and finally a Neural Network (with and without post-processing). Each classifier is discussed in more detail below.

The K-NN, Decision Tree and Ensemble Random Forest are implemented using the Scikit-Learn framework. Their hyper-parameters were tuned using a grid search with objective to minimize the validation set error. The Neural Network is implemented using the Fast Artificial Neural Network (FANN) library [5].

3.1 K-Nearest Neighbours

K-Nearest Neighbours performs predictions by retaining the training input data and identifying the K training set data points which are closest to the current case. The degree of 'closeness' is determined by some distance metric. These K neighbours are then combined to produce the prediction.

In the context of the road-classification problem, the K nearest neighbours to the current image being evaluated are identified and then vote on a per-pixel basis whether the pixel is part of the road or not.

We used euclidean distance (L2-norm) as the distance metric. The results from the K neighbours were combined using either uniform weights (i.e. majority vote) or a weighted vote based on distance. The choice of weights (uniform or distance) and choice of K were treated as hyper-parameters. The optimized hyper-parameters for each dataset are shown in the following table.

	K	**Weights**
UM Lane	1	uniform
UU Road	1	uniform
UM Road	1	uniform
UMM Road	5	uniform
All Road	5	uniform

Somewhat surprisingly for the UM, and UU datasets the best K-Nearest Neighbour classifiers used K=1. This corresponds corresponding to simply returning the ground truth for the closest training point. For the more complicated UMM (and All) datasets the classifiers performed best with K=5, indicating that the averaging between neighbours was beneficial. In all cases using uniform weights (rather than weighting by distance) produced the best prediction.

3.2 Decision Tree

A decision tree consists of a tree of binary decisions that are made based upon the values of input features. The sequence of binary decisions will eventually lead to a leaf node of the tree which contains the output prediction. The decision tree is created (trained) by splitting on the input features that provide the most differentiation between the resultant classes.

The key hyper-parameters tuned for the decision tree classifier were: the maximum number of levels in the tree, and the minimum number of samples allowed to be associated with a leaf node. A larger the number of levels results in a larger decision tree with a more complex decision boundary. While this can help capture more knowledge it may also lead to overfitting. Enforcing a minimum number of samples per leaf can help counteract overfitting by forbidding nodes that account for only a small number of training examples. The optimized hyper-parameters for each dataset are shown in the following table.

	Maximum Depth	**Minimum Samples per Leaf**
UM Lane	10	10
UU Road	10	10
UM Road	10	10
UMM Road	10	20
All Road	10	20

A maximum tree depth of 10 performed best on all datasets. The simpler datasets (UM, UU) performed best with a minimum of 10 samples per leaf while the larger and more complicated datasets (UMM, All) performed best with a minimum of 20 samples per leaf. It is possible that the large number of samples per leaf helps to prevent overfitting on those datasets.

3.3 Ensemble Random Forest

The Random Forest classifier uses an ensemble approach, where a collection of classifiers (in this case decision trees) are trained on different subsets of the training data, are evaluated and their results averaged to produce a (hopefully) more accurate prediction.

The hyper-parameters for the random forest are the same as those for the decision tree, but also includes an additional parameter, the number of decision trees estimators. The optimized hyper-parameters for each dataset are shown in the following table.

	Number of Estimators	**Maximum Depth**	**Minimum Samples per Leaf**
UM Lane	5	10	5

UU Road	10	10	5
UM Road	10	10	5
UMM Road	30	7	5
All Road	80	7	5

The hyper-parameters selected illustrate an interesting trend. Generally, as the data set size and complexity increases the number of estimators also increases (from 5 to 80), while the maximum tree depth drops from 10 to 7 and the minimum samples per leaf stays constant at 5. This indicates that better results are achieved, as the complex of the datasets increases, by using a larger number of less powerful estimators. This makes sense for an ensemble approach.

It is also interesting to note that the minimum samples per leaf is smaller here than in the decision tree classifier. This is likely because overfitting is less of a concern in the ensemble, since each tree in the forest is trained only on a subset of the data, and the final output is produced by averaging - both of which counteract the tendency to overfit.

3.4 Neural Net

The Neural Nets (nnets) were created to map the reduced images from PCA to the binary decision for each pixel. So the three layers in the network have the following sizes: <pca_components>, <hidden_units>, <num_pixels>. At every layer a sigmoid function was used, so the output for each unit ranged between [0, 1]. The correctness of a pixel was determined by checking if the output was within 0.5 of the expected output.

To tune the hyper parameters the validation set was used to ensure the nnet was not overfitting. While tuning the network size it was apparent adding more than 1 hidden layer was unhelpful and just made the nnet harder to train. Given the nnet had over 400000 outputs a large hidden layer would require a large amount of memory and processing. This limited the hidden layer to at most ~500 neurons. While tuning the optimal hidden layer size, it was found that for all road types 20-30 hidden units performed best. The most likely reason for only needing a small hidden layer is the low number of training examples. There is not enough data to warrant a more complicated network. Finally, tuning the learning rate had little effect on the result. A learning rate of 0.1 was used for all tests.

	Hidden Neurons
UM Lane	30
UU Road	30
UM Road	20
UMM Road	30

All Road	20

3.5 Neural Net with Post Processing

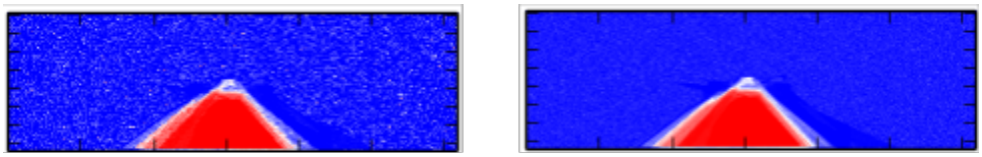

Fig 2: Prediction before (left) and after (right) smoothing

Unlike the other classifiers the output of the nnet was noisy, even in regions of the image which clearly fell into a particular class. To remove the noise we smoothed the nnet output by averaging the surrounding 5x5 pixel region (for the edges we averaged just the pixels in the image). The intuition behind smoothing is that if a pixel is surrounded by pixels that are part of the road then that pixel most definitely also part of the road. Figure 2 shows an example of a prediction before and after smoothing.

4. Experimental Results

The following table presents the mean error generated by each classifier on the test set for each dataset.

	K-NN	**Decision Tree**	**Random Forest**	**Neural Network**	**Neural Network + Smoothing**
UM Lane	**1.851%**	2.106%	2.098%	2.933%	2.646%
UU Road	5.885%	**5.148%**	5.350%	5.975%	5.375%
UM Road	**2.911%**	3.500%	2.928%	3.458%	3.324%
UMM Road	8.106%	8.889%	8.503%	7.979%	**7.861%**
All Road	**6.264%**	7.053%	6.893%	7.447%	6.971%

First we can look at the relative difficulty of the different datasets. The UMM Road dataset is clearly the most challenging, producing highest error rates across all classifiers. This is likely due to the complicated nature of the roadways in this set as illustrated in Figure 3. Since it also includes the UMM data set the All dataset is also difficult. Somewhat surprisingly (since the street layout is relatively simple) UU dataset has higher error rates than the UM datasets. Closer inspection shows that this is likely due to the presence of obstacles

such as parked cars which make the decision boundary between road and not road more complicated, as illustrated in Figure 4.

Looking at the overall performance of the different classifiers they generally perform similarly with error rates within 1% of each other. However some classifiers do tend to perform better than others. In particular K-NN performs best on three of the benchmarks by relatively large margins. However K-NN performs poorly on the more complicated road patterns in the UMM dataset, where the Neural Network performs best. The Decision Tree and Random Forest classifiers tend to do well on the simpler data sets (UU and UM) but also perform poorly on UMM. Typically the Random Forest classifier produces better predictions than a single decision tree. An example prediction by the Random Forest is shown in Figure 5. It is interesting to note that in Figure 5 the classifier does shy away from the car ahead, but fails to do so aggressively enough to avoid over-predicting. This could be an artifact of the ensemble method where some of the ensembled classifiers correctly detect the car, but the majority overrule them, leading to only partial detection of the car.

The neural net classifier tends to hedge its bet on the average road based on the shape in the image. Then it surrounds the road in a margin of uncertainty. In doing so it misses out on obstacles such as cars in neighbouring lanes. It is also likely to pick up strips of grass along the side of the road if there is not a harsh change in colour or lightning. This is all illustrated in figure 6.

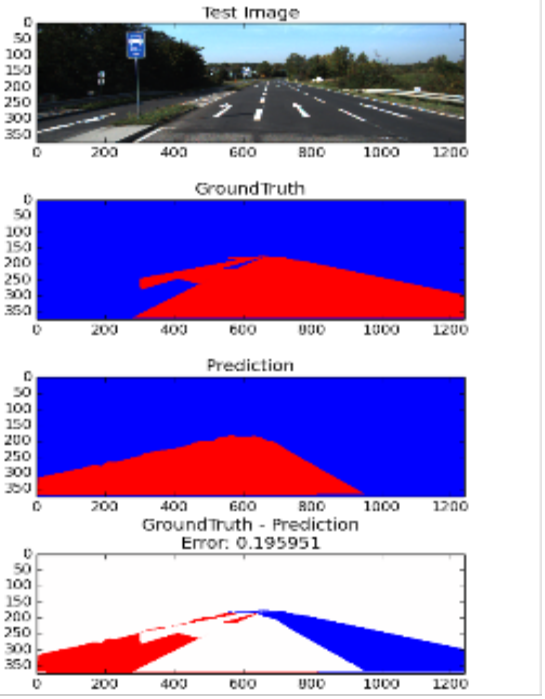

Figure 3: K-NN Errors on a UMM test example. For the centre two images roadway is shown in red. In the bottom image over-prediction is shown in red and under-prediction in blue.

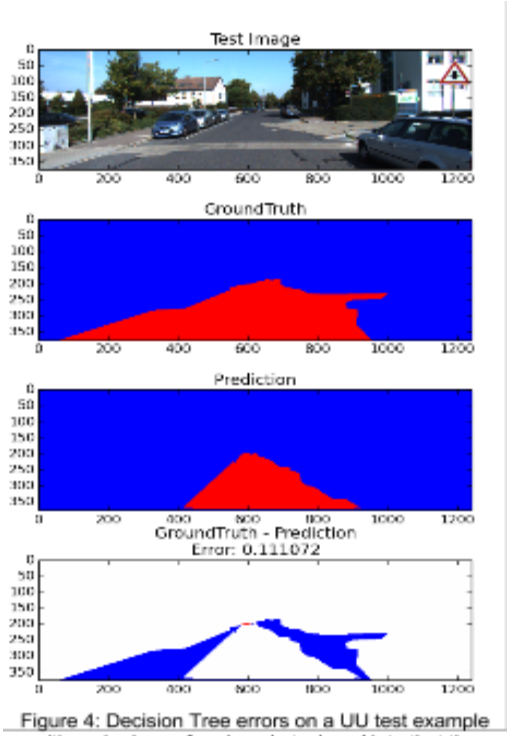

Figure 4: Decision Tree errors on a UU test example with parked cars forming obstacles. Note that the decision tree successfully avoids classifying the cars as part of the road.

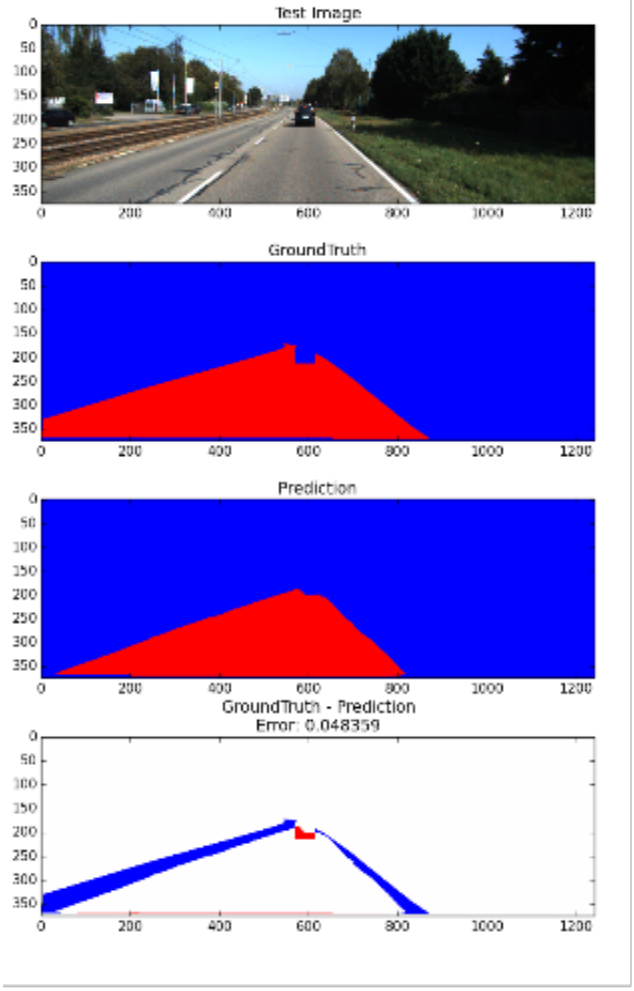

Figure 5: Ensemble Random Forest prediction on a UM test example.

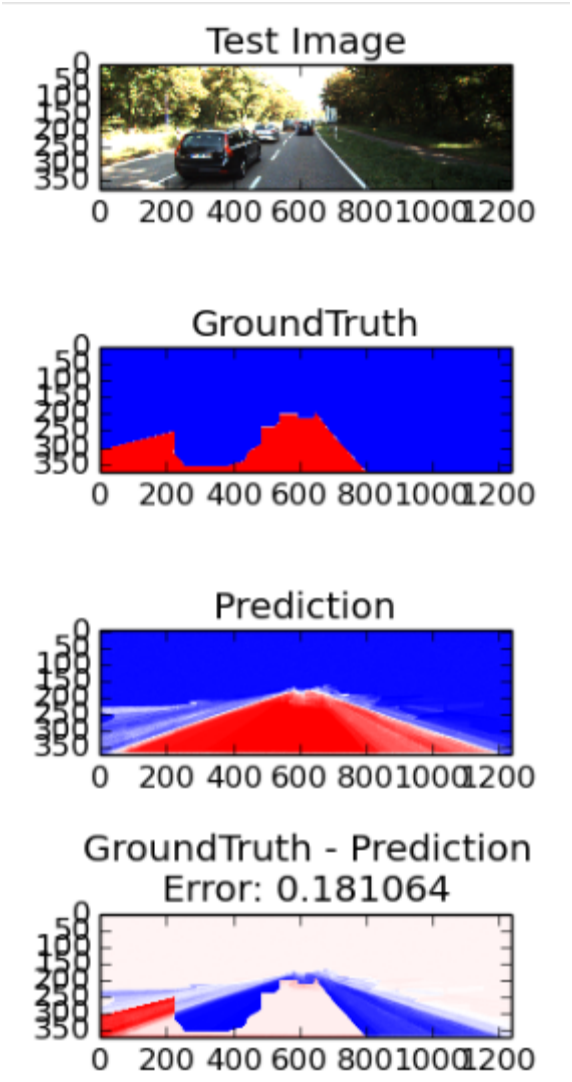

Fig 6: Neural Net prediction on a UMM test example.

5. Conclusion & Future Work

In conclusion, the classifiers on average tended to perform similarly, although they each have different strengths and weaknesses. K-NN performs best on the simpler cases such as UM lane and road, while the smoothed neural net is better on the more complicated UMM dataset. With more data we expect the techniques could be better differentiated.

We found that PCA was also very effective, reducing the dimensionality of the data set by multiple orders of magnitude. This allowed us to process the full data set at the pixel level with only a modest amount of computation. However, PCA does not help reduce the (relatively high) dimensionality of the output prediction. This was particularly problematic for the Neural Networks. One possible approach is to scale the training targets down by a factor of 2 or 4 (i.e. predict in a lower dimensional space) and then scale it up to the original output space.

In the process of investigating these techniques many directions for future work were identified. For starters the most limiting factor we faced was the small amount of training data. Additionally, most of the data is taken from a very similar vantage points, ie in the middle of the road looking straight ahead. Having images where the road is not in the center or the the road is not at a 90 degree angle would help make the classifiers much more powerful. However, collecting such data (e.g. driving off or across the road to get these images) is difficult. Instead one could artificially inflate the amount of training data through simple scaling, translation, and rotations. For this to work the edges of images would need to be filled by some content aware algorithm. The label images would be able to be updated trivially by applying the same transformations.

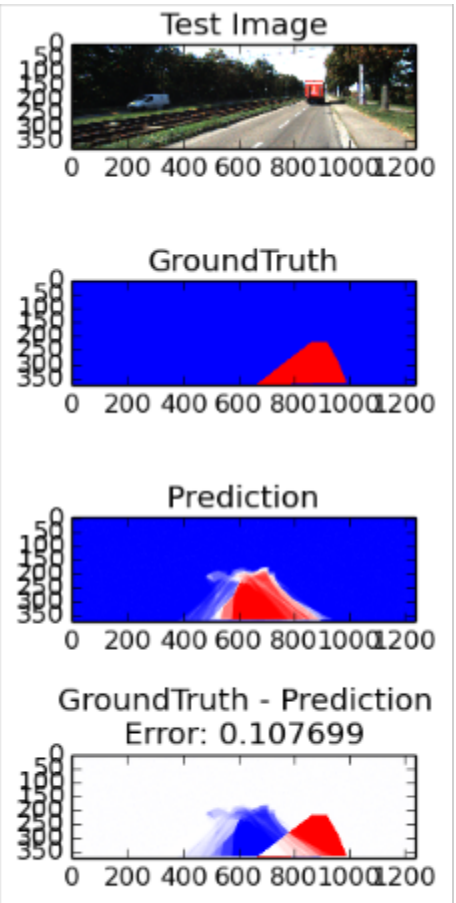

Fig 7: Neural Net prediction on a UM lane test example.

The example above is one of the few data points where the lane extends up and to the right, The lack of such data points leads many of the predictors to perform poorly on these cases. Rotating existing images would create more cases for the classifier to train on improving performance on this case.

It would also be useful to provide training data covering different times of day (resulting in different lighting/shadow positions) and different weather conditions (e.g. rain, snow). Any practical road classification system will need to handle these conditions.

Another direction for future work would be using a mixture of expert strategy, allowing classifiers to specialize in certain road types or conditions. Given that we observed specific types of classifiers performing better on some data sets than others, this would allow their various strengths to be combined. Being able to delegate cases to specialized classifiers would probably be very effective.

Another potential future improvement would be to consider re-weighting the different classes. This could help to remove the natural bias in the data set towards classifying pixels as non-road (since the road forms generally a small portion of the overall image). It could also be used to emphasize the relative importance of the different types of prediction errors. Specifically it could be used to bias the classifier towards safe behaviour (i.e. predicting not road when there is really a road) and against unsafe behaviour (i.e. predicting road when there is no road).

References

[1] J. Fritsch, T. Kuehnl and A. Geiger, “A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms”, Int. Conf. on Intelligent Transport Systems, 2013

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fau and S. Susstrunk, “SLIC Superpixels Compared to State-of-the-art Superpixel Methods”, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 34, num 11, p. 2274-2282, May 2012

[3] “Python Imaging Library (PIL)”, Python Ware, www.pythonware.com/products/pil/, 2009

[4] F. Pedregosa, G. Varoquaux et al., “Scikit-learn: Machine Learning in Python”, Journal of Machine Learning Research, vol. 12, p. 2825-2830, 2011

[5] S. Nissen, “Implementation of a Fast Artificial Neural Network Library (FANN)”, Department of Computer Science University of Copenhagen (DIKU), 2013