A fusion between a deep-learning and a filter-based approach for road marker detection

Bachelor's Project Thesis

J.G.S. Overschie, j.g.s.overschie@student.rug.nl
Supervisors: Dr. G. Azzopardi

Abstract: In this study, a fusion between two existing algorithms for detecting road markings is proposed. Considered are two existing algorithms, one supervised, deep-learning algorithm, and another unsupervised filter-based algorithm. The proposed fusion algorithm takes the output images of both approaches in order to train a new model using supervised learning. Under different conditions of downscale factor, classifier and feature-vector composition, the fusion algorithm managed to obtain an accuracy level of 95%.

1 Introduction

This research uses the output of two existing ways of processing and classifying images with road markings, in order to train a new supervised model based on top of the existing two. One of the algorithms has used deep-learning, using a supervised approach to detect the road markings, whilst the other algorithm used a filter-based, unsupervised approach to detect road markings.

Both approaches output images, where the road markings are accentuated by using different color intensities in the image. This study only considers the output of both algorithms, the output images. For this reason, the exact workings of both algorithms are considered as a 'black box'; no information is taken into consideration regarding the workings of the algorithms, only the output is considered.

The data-set used consists of images separated in four different classes: the ground truth, the supervised approach output, the unsupervised approach output and the input images. The data-set contains 1000 ground truth images, for which every image has a supervised- and unsupervised image counterpart, sketching the exact same road situation. This means that supervised training can be used to train the combined output of both the supervised- and unsupervised approach using 1000 ground truth images. The collection of supervised and unsupervised imaging contains a total of 3117 images each, meaning that some imaging cannot be used to train or for which results cannot be automatically validated. This imaging can be validated, however, using manual visual inspection.

Applications of road marker detection exist in various areas of interest. One of these areas of interest is road damage inspections. Road damage inspections are a tedious process, where many hours are spent manually evaluating imaging taken from the road to assess the level of road damage. For this automated ways of road marker detection can be used, with an extension to road marker damage assessment. Another area of interest is autonomous driving, in which it is crucial that the artificial intelligence understands and correctly interprets the markings on the road. A first step to both determining road damage and developing autonomous vehicles is to outline road markings on the road, which process this study, in turn, tries to improve.

This research will focus on proposing a fusion approach between the two existing approaches, where the question will be answered: "What is the performance of a fusion algorithm in comparison to two existing approaches to road marker detection?" To best assess the performance of the fusion algorithm and to understand the most influential parameters, different metrics are compared, like accuracy vs. pixel configuration, or accuracy vs. classifier, as well as the AUC score and ROC curve.

The study is confined to a certain scope. The problem is approached using the designated data-set containing 1000 ground-truth images, which will bring with it its specific distribution of; types of road marking, weather conditions and road type. No data outside this data-set is explored, trained or tested. Road imaging is also down-sampled from its original pixel size resolution using several down-scale factors, in order to reduce the amount of data-points that are to be processed. The study is confined to processing only down-scaled imaging; processing original image resolutions is beyond this study scope since it requires professional-grade hardware not available to this study. The study further restricts the amount of data-points of every down-scaled image by sampling, albeit this is done randomly. Furthermore, the study confines itself to testing two classifiers on the problem; a *Support Vector Machine* using LibSVM (Chang and Lin, 2011) and *Gradient Boosting* using XGBoost (Chen and Guestrin, 2016).

In the next chapter, first a review on relevant existing literature is given. Then, the method of research and the results will be discussed. The method will start with a general outline, which aims to give a broad perspective on the method. Next, in the method a more thorough and technical explanation of the used pipeline will be given, in which the different configurations and parameters will be explained. In the results section, first an insight is given of the overall test results. Then, we compare the performance between approaches. Finally, a set of conclusions are drawn upon the entire set of results, which will be accompanied by a discussion on the different shortcomings in the study or influential biases that might have been at play.

2 Literature Review

The road marking detection problem has been around for a longer time, with relevance ranging between several areas of application. The output that is ought to be obtained also differs between these areas of application. In some applications, it is most important to precisely where a lane is located, in others it is most relevant to be able to precisely compute lane surface area. Several literature pieces are compared and an exploration is made into the contributory value of this study in comparison to existing literature.

In the paper “Real-Time Detection of Road Markings for Driving Assistance Applications” (Chira et al., 2010) a solution is sought for detecting lanes in application of driving assistance. The proposed method provides detection of the ‘five most common road markings’; several samples of road markings are taken from European roads, which the study shows to be able to successfully classify in specific cases. This means that no general-purpose method of classifying any piece of road as a road-marking one is proposed. This is important to take into account since the method proposed in this study aims to provide a general-purpose road marker detection paradigm. Imaging is taken from a driver-seat perspective and is then transformed such that a 2D top-level perspective view is constructed. Classification accuracy is said to be between 85% and 95%, depending on the distance between road marker and camera, over an unknown number of total object classifications. The fact that this study aims to provide a solution for detecting any road marker piece of road and not just a subset of all road markings given by geometrically predefined shapes, the two studies serve different research goals.

Another study that focused on detecting lanes and road markings is “Road marking features extraction using the VIAPIXs system” (Kaddah et al., 2016). This study had as goal in mind not only to detect road marks, but also to perform geo-referencing and determine road surface condition. The study used inverse perspective mapping and color segmentation to detect all white objects existing on a road. This is similar to the filter-based approach this study uses to in its fusion algorithm. Several video feeds were used as image inputs, containing different weather conditions and road situations. Road marking classification accuracy ranged from 87.5% to 100%. Even though the study used color segmentation to process imaging at first, the study confined itself to detecting only a specific type of road marking. Because the study does not attempt to fuse two methods of road marking detection and confined itself to detecting only specific types of road markings, this literature does not overlap with the proposed study.

A study that focused on using machine-learning, more similar to the supervised deep-learning approach this study uses as one fusion-component, is “Robust Road Marking Detection and Recognition Using Density-Based Grouping and Machine Learning Techniques” (Bailo et al., 2017). Several classification methods were tested in the study; PCANet+SVM, PCANet+Logistic Regression, and a Shallow CNN. Classifier accuracy lies in the high 90-percentage ranges for well-performing classifier-stacks. Although different techniques are stacked in a similar way as this study proposes, a different type of combining is performed. In the proposed method in this study, no classifiers but approaches are stacked and the study at hand combines classifiers. Albeit (Bailo et al., 2017) has shown that combining different techniques has a possibility of yielding interesting results, it is not similar to the proposed method.

In the this study, a fusion method is proposed. Several ways of fusion different data sources exist. (Castanedo, 2013) The technique this paper has focused on is decision fusion. For the proposed methods, two already-made decisions are combined into a third. In previously discussed literature no experiment was conducted where decisions were fused - in current literature both approaches exist separately, albeit no decision fusion between a filter-based and a deep-learning approach. Among this reason, the proposed method has not been tried before, therefore making it a unique piece of research, further exploring the best way to road marker classification.

3 Method

Machine-learning is used in to merge the two existing algorithms and predict new results. In order to generate reproducible results, a machine-learning pipeline is used. In this pipeline, a number of steps are executed, each taking the output of the previous step as an input, and then generating new output. This makes for a modular and predictable sequence of steps generating a result.

The pipeline built in the study has a sequence of steps. First, re-sized and re-scaled versions of the data-set imaging are cached, such that this computationally expensive operation has to be ran only once. Next, the cached imaging is taken, and is transformed to the correct data-structure format such that the data can be trained by a machine-learning model. After the transformations the data-set is trained using a chosen classifier, storing the fitted classifier afterward. Taking the trained classifier, finally new results are predicted in the test step, storing the results afterward. Optionally, the results can be visualized by running an additional step in the pipeline.

Python is a common platform for data science projects. Since a broad range of tools and packages are available in this ecosystem, the pipeline is written in Python.

3.1 Support libraries

The pipeline makes use of preexisting tools to help with common tasks. In order to read, write, and process images the scikit-image library (Van der Walt et al., 2014) is used. This library has built-in functions to re-size and re-scale images, as well as convert images between different data type formats, e.g. converting $float64$ image data-types to $uint8$. Scikit-image also facilitates RGB- to gray-scale color conversion.

Next, in order to run existing classification models and other common machine-learning related tasks, such as transforming the data or analyzing test results, the scikit-learn library (Pedregosa et al., 2011) is used.

Lastly, Numpy (Van Der Walt et al., 2011) is used for creating and altering arrays as well as working with n-dimensional arrays.

3.2 Pipeline

The pipeline consists of four essential steps, with one optional visualization step such that results can be plotted. All steps are made to be executed in order, but independently of each other; results are stored on disk once the step finished. This means that for a different training configuration, the previously stored results of the caching step and the transformation step can be used, and do not have to be computed again.

3.2.1 Caching

In the caching step, the raw images of the data-set are re-sized, gray-scaled and stored in a consistent data type format.

See Figure 3.1 for the steps that are taken during the caching process.

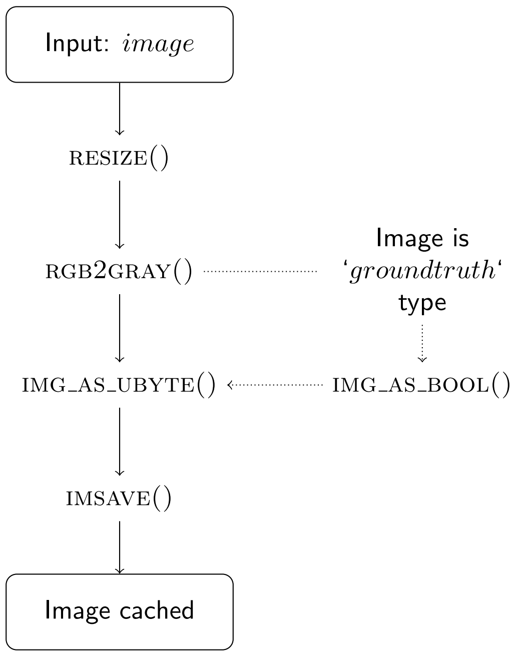


Figure 3.1: Individual image caching steps.

Different caches are built with various pixel configurations of different scaling factors. Images are down-scaled with various factors, ranging from 3.0/4.0 to a factor of 1.0/20.0. Furthermore, a step is required to standardize the image data type, IMG_AS_UBYTE(), because not all images are ensured to be of the same data type. Not standardizing the image data type would lead to inconsistent results.

Noticeable in Figure 3.1 is that *groundtruth* images get an extra step before saving. This is because the data-set *groundtruth* images are not binary, see Figure 3.2.

Although most values lie at the 0.0 or 1.0 value-point, there also exist some values in between. Because there are relatively little values in between, a logarithmically scaled histogram is needed to visualize them. In order to make the groundtruth images binary, a mapping method is used. An easy way is to cut the spectrum in half, allocating all values in the range $[0.0, 0.5]$ as 0 and all values in the range $(0.5, 1.0]$ as 1. The skimage function IMG_AS_BOOL() is used for this functionality. Using a different technique for binary conversion, such as

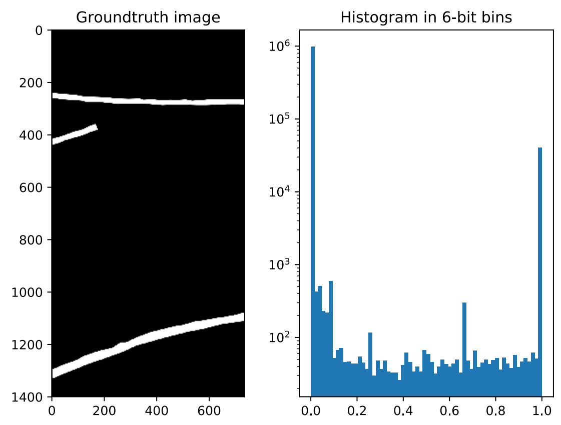


Figure 3.2: Histogram of a groundtruth image in *log* scale.

the THRESHOLD_YEN() threshold function from the skimage library, led to significantly worse results.

Once caching is done, several directories have been created with correctly sized imaging. The imaging has a consistent data type and is suitable for transformations once cached.

3.2.2 Transforming

In the transformation step, all caches are prepared such that the data can be trained. For every cache, which has its own image pixel configuration, a separate new prepared data file is written to disk using the joblib library (Varoquaux and Grisel, 2009).

First, the cache imaging data is divided into several folds using a k-fold split, in order to facilitate k-fold cross validation (Stone, 1974). A shuffled k-fold split is made to ensure randomness between folds. Cross-validation is used to prevent over-fitting or selection biases. The KFOLD.SPLIT() function from the sklearn library is used to split the data-set into a *n_splits* amount of folds, where every fold has a selection of training- and test images.

Secondly, for every fold, individual images of both the training- and test set are converted into vectors, by using a 2-d to 1-d translation. Then, the training set is sampled, in stratified fashion; for every image, a balanced amount of samples are taken between the two classes. This is to ensure that non- road-marking pixels or road-marking pixels are not over- or underrepresented. A maximum of *max_sample_size* amount of samples are taken per training-image. A minimum cannot be set, since the samples are desired to be balanced, and there exists no assurance that there exist any road-marking pixels in any image at all.

Thirdly, the data of the supervised- and the unsupervised approach are combined. For every pixel location, the value of both the supervised- and unsupervised approach are taken and combined into a 2-feature vector, resulting in the following:

$$fv = \begin{pmatrix} sv & usv \end{pmatrix}$$

Where sv and usv are pixels corresponding to the supervised and unsupervised approach, respectively, and fv is the constructed feature vector. Combining vector values from two sets is done using Numpy STACK and HSTACK functions.

A different feature vector is formed conditionally, based upon a configuration preset constant called "FEATURE_VECTOR", which can have values of either "2-ELEMENT" or "5-ELEMENT". As values suggest, a 5-element feature vector can also be constructed. The 5-element feature vector consists out of the supervised- and unsupervised approach pixels, as well as the input image rgb values. A 5-element feature vector will look like the following:

$$fv = \begin{pmatrix} sv & usv & src\_r & src\_g & src\_b \end{pmatrix}$$

Where src is a RGB pixel from the input image, of which each color value is inserted separately, and fv is the constructed feature vector.

Lastly, all individual vectors representing the imaging are concatenated into large 1-dimensional arrays representing training- and test data. The data is then in a format acceptable by sklearn classifiers for training.

3.2.3 Training

The data-set is trained using several classifiers. In order to speed up training times and make use of parallelization speed-up opportunities, a *BaggingClassifier* is wrapped around the designated classifier using parameter $n\_jobs = -1$ such that all system cores can be utilized. Regarding to training times, it is reasonable to notice that in the previous step data was sampled, such that regardless of cache all imaging have the same number of data-points that are to be trained.

Because each cache data-set is divided using a k-fold split in *n_splits* amount of folds, an *n_splits* amount of classifiers are trained for each cache. Once the classifier has been successfully trained, the fitted model is saved to disk, such that it can be used for testing.

3.2.4 Testing

In the test step, the test-split data is used to predict new images and then validate results. The testing step can, based on two configuration variables, both predict new pixel classes or predict new pixel probabilities. Previously trained classifiers are loaded from disk, along with the transformed test data.

If the class prediction configuration variable is enabled, new binary values are predicted based on the 2- or 5-element feature vectors. An accuracy score is determined and a new 2-d image is reconstructed based on the 1-d predicted values vector. Predicted images and prediction accuracy statistics are saved to disk.

If the probability prediction configuration variable is enabled, the designated classifier is setup such that a new probability is predicted for each pixel. Of both the probability for a non- road marking pixel and a road-marking pixel, the probability for a pixel being a road-marker pixel is stored in a 2-d float formatted image.

Because a k-fold cross-validation setup is used, a total of *n_splits* analyses have to be executed. Following the setup of a k-fold cross-validation, all testing images available are tested using all folds. When class prediction is enabled, total cache accuracy score is determined as a mean of all folds compared.

3.2.5 Visualization

Last to the pipeline is the visualization step, which exists to interpret testing performance. Because the test pipeline step is built to output mainly raw data in joblib format, this pipeline step exists as an unmissable component to the pipeline performance assertions.

It reads previously computed results from disk, which have been previously cached by the testing step. Then, data is transformed into several data structures to obtain a suitable data format for plotting. To then visualize the data matplotlib (Hunter, 2007) is used. For some non-standard charts, requiring more advanced options, supplementary libraries *Seaborn* and *Pandas* (McKinney, 2010) are used.

4 Results

In the results, first the performance of the fusion algorithm is explored using a 2-element feature vector. Next, the 5-element feature vector fusion algorithm performance is analyzed and compared to the 2-element feature vector. Then, an analysis is made on the difference in performance between the supervised, unsupervised and fusion approach. Lastly, pipeline performance measurements are taken into consideration.

4.1 Fusion algorithm using a 2-element feature vector

The data-set was tested under several configuration conditions. Different- pixels size configurations and classifiers were used, all using a 10-fold cross validation setup. According to the decided upon scope of the study, two classifiers are used; *Support Vector Machine* (SVM) and *Gradient Boosting* (XGBoost). General results of the prediction are first explored, using the *XGBoost* classifier. Afterward, a comparison between the two classifiers is made.

4.1.1 Overall performance using the XGBoost Classifier

In order to first get a general idea of the overall performance of *XGBoost*, a Violin chart containing accuracy scores of all images categorized by pixel size configuration is first taken. A violin chart extends on a box-plot such that data distribution is better visible. See Figure 4.1.

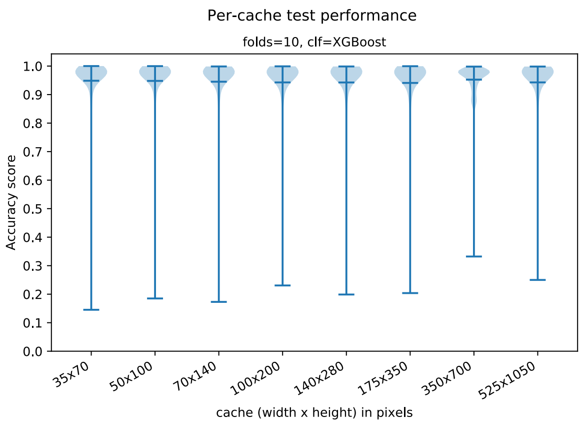


Figure 4.1: XGBoost accuracy performance for total test-set of 1000 images for fv=2-element.

The plot shows, that although test accuracy is mostly distributed in the $[0.9, 1.0]$ range, accuracy scores are minimized at around the 0.2 accuracy level. In pursuance of getting a better idea of how the accuracy scores are distributed, a histogram plot can provide better insight. A histogram is shown in Figure 4.2, showing accuracy score distributions, along with a trend-line, in logarithmic scale.

Plotting distribution in logarithmic scale allows for visualizing less frequent accuracy scores in the lower score ranges. A small amount of scores lie in the $[0.2, 0.8]$ range, with most scores above 0.9. Because the lower score range form 'outliers' in this result data-set, a certain parameter might be a cause for the low scoring. Although most imaging has road-markings on it, some images completely void of any road marking pixels. In order to explore better a potential over-fit or under-fit, a plot of the accuracy score against the amount of road marking pixels in an image is explored. The amount of road marking pixels is represented as the fraction of road marking pixels compared to the non- road marking pixels in an image. See Figure 4.3.

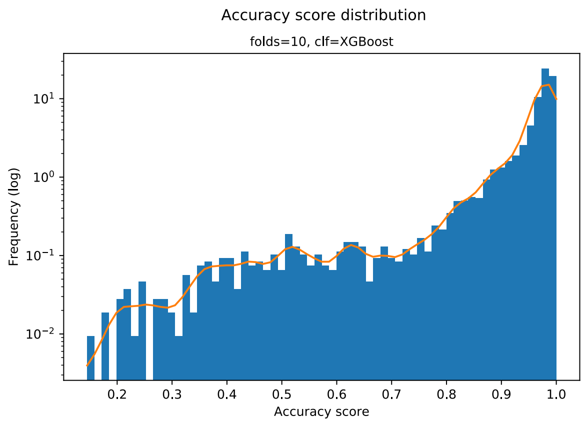


Figure 4.2: Histogram showing accuracy score distribution for all pixel size configurations at a test-set of 1000 images in *log* scale. 2-element feature vector.

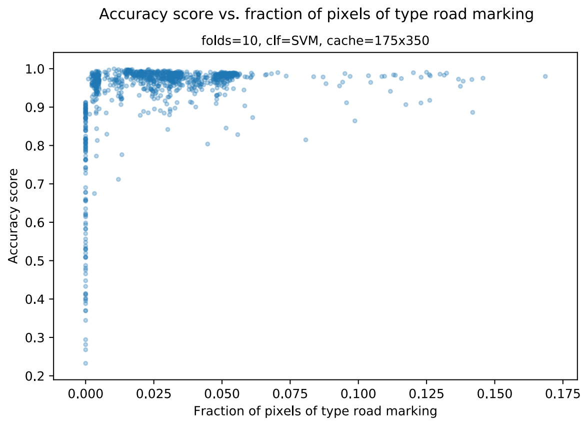


Figure 4.3: Plotting accuracy score vs fraction of road marker pixels, for 2-element feature vector.

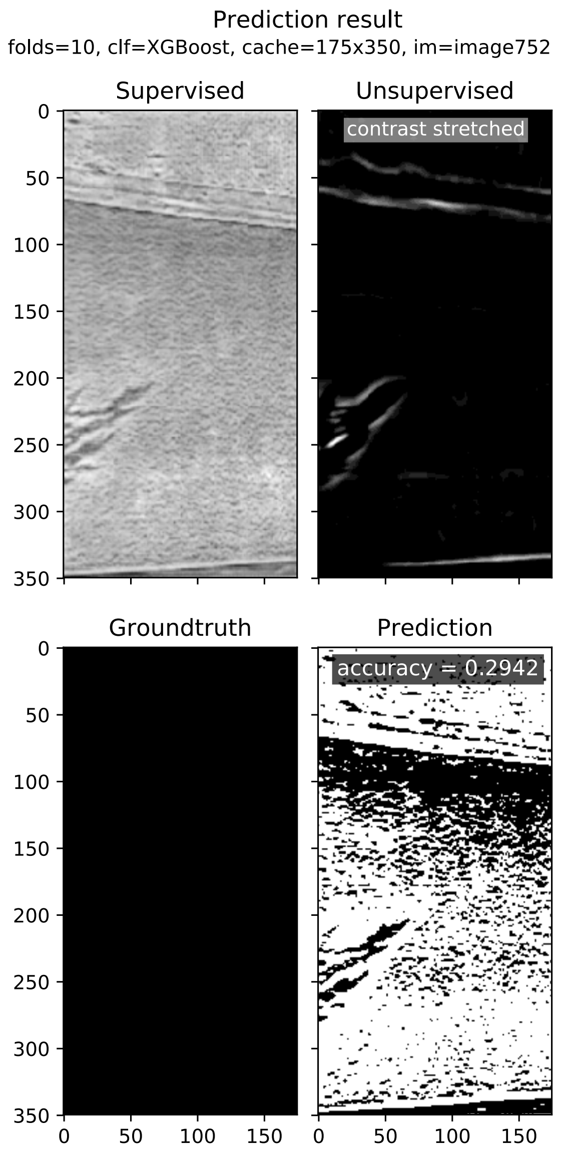


Figure 4.4: Accuracy score of 29.42% of an image, using a 2-element feature vector, with no road markings marked in its groundtruth.

In the chart is visible that a possible correlation exists between low accuracy scores and images with few road-marker pixels in them. There exist a clear vertical line with various levels of low-accuracy scoring, all at the 0.000 fraction level. One such image, taken from the 175x350 pixel size configuration, is shown in Figure 4.4.

It can be observed that the ground truth depicted no road markings at all, but the fusion algorithm predicted almost the entire image as road-marking pixels nonetheless.

There, however, also exist some similarly challenging images with a significantly higher accuracy score, as to be seen in Figure 4.5.

4.1.2 Classifier- vs pixel size configuration comparison

Predictions of different accuracy levels were explored, all for the XGBoost classifier. Classifier performance is similar for most situations, therefore all individual metrics for the SVM classifier are not explored. Instead, a comparison is drawn off the performance per-pixel size configuration for each of the two classifiers. A box-plot categorical comparison is drawn, where every box-plot was supplied with the means of their respective folds, which means in the case of $n\_splits = 10$, there will be 10 data-points per box-plot. See Figure 4.6.

Important to note is the fact that for the 525x1050 cache, results are only obtained for XGBoost. This is due to the very high testing times of the SVM classifier when large pixel size configurations come into play. Although training times for both classifiers remain somewhat constant despite increasing pixel sizes, testing times per image get significantly higher with higher pixel size con-

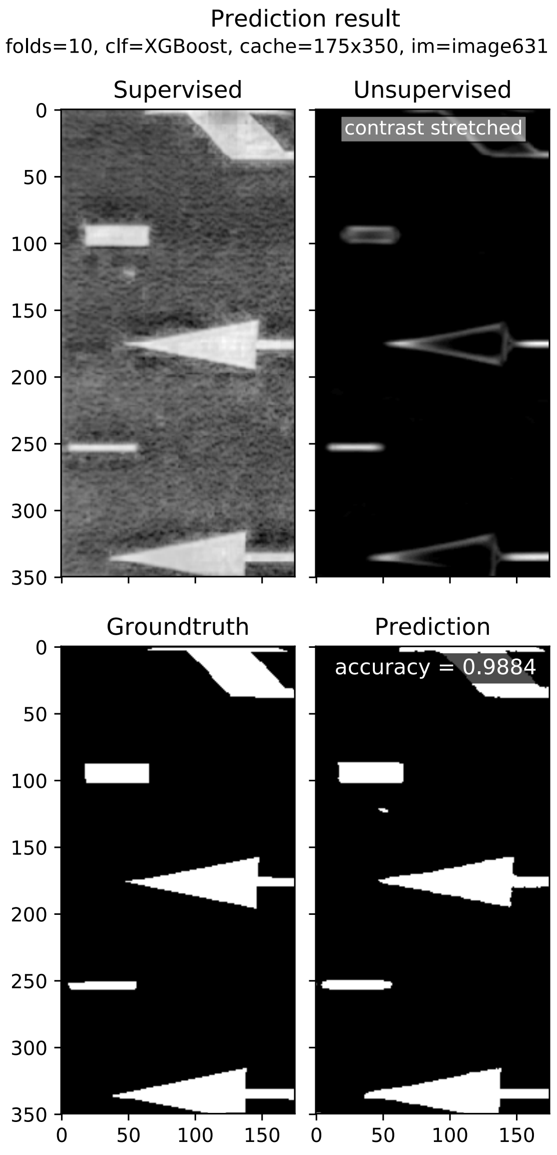


Figure 4.5: A prediction of an image with multiple road markings using a 2-element feature vector, yielding an accuracy score of 98.84%.

figurations. The reason for this is that the training set is sampled, reducing the number of pixels that are used for training, which applies to all pixel configurations. A max amount of samples of *max_samples* are taken per image. In testing, however, an increase in pixel size means a significant increase in total data-points that have to be predicted. Classifier- and pipeline CPU performance is further explored in Section 4.4.

To get a better understanding of what exactly the performance of the individual caches are, a plot with aggregated classifiers is required. In the following plot, data-points are not categorized by any classifier, aggregating all results such as to obtain a per-cache performance chart. See Figure 4.7.

To get a better understanding on the other hand

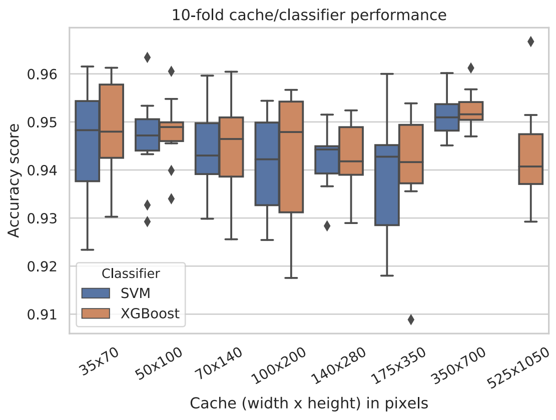


Figure 4.6: Classifiers performance comparison, taken using the means for every split, for $n\_splits = 10$**; resulting in 10 data-points per bar.**

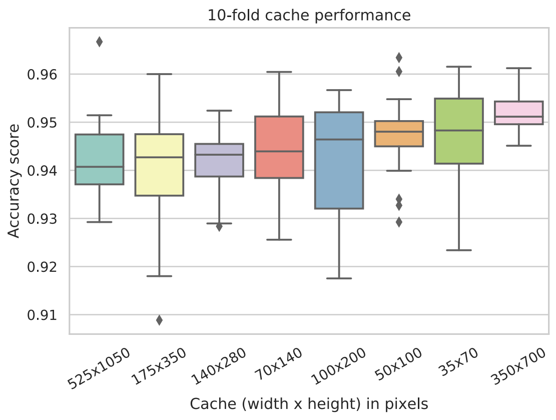


Figure 4.7: Per-pixel size configuration score chart using a 2-element feature vector, obtained by taking the mean of every fold, for both classifiers aggregated.

of the individual classifier performance, a chart depicting classifier performance with aggregated cache results is necessary. See Figure 4.8.

In order to further explore the statistical difference in also the error made by both classifiers, a confusion-matrix is used, drawing a comparison of true labeled classes vs. expected labels. All output images of the designated cache and all the according groundtruth images are taken into consideration. Every predicted data-point is then compared to its true value - in the case of the 350x700 cache 245.000, 000 million data-points are compared. XGBoost- and SVM confusion matrices are presented in Figure- 4.9.

Shown by the confusion matrices is that XGBoost classifies more data-points as non- road markings than the SVM classifier, even though it is a road-marker data-point. This in comparison to the prediction of the road-marker label, which the SVM classifier predicted wrongly more often, 0.0441 scor-

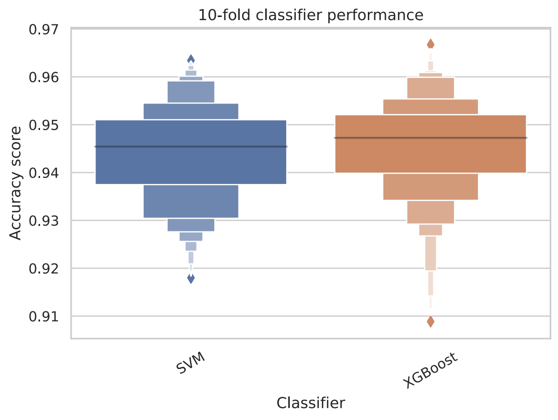


Figure 4.8: Cache data aggregated into classifier categories plotted against accuracy score using a 2-element feature vector. *SVM* showing a median of 0.9454 and mean of 0.9442. *XGBoost* showing a median of 0.9473 and mean of 0.9454.

ing vs. 0.0432 scoring, for SVM and XGBoost respectively. In overall both classifiers achieve an accuracy score of approximately 0.95 for the 350x700 cache; 0.9524 for XGBoost and 0.9515 for SVM to be specific.

4.2 Fusion algorithm using a 5-element feature vector

An experiment using a 5-element feature vector was also run. As was explained in Section 3.2.2, a 5-element feature vector is constructed by combining the supervised- and unsupervised approach images and the input image RGB values. Visible in Figure 4.10 is the overall prediction performance of the fusion algorithm using a 5-element feature vector. Noticeable is the fact that SVM performs significantly worse than XGBoost. SVM performance gets lower with increasingly larger pixel size configurations, whilst XGBoost performance improves with larger pixel size configurations.

Visible in Figure 4.11 is that accuracies are similarly distributed like the experiment using a 2-element feature vector. Accuracies are mostly distributed around the [0.93, 0.99] range, but go down up to 0.2.

4.3 Approaches compared

In comparing the supervised-, unsupervised- and the newly built fusion approach, AUC scores and a ROC curve is used. For a given approach, the metrics are generated by taking into consideration all its images, and comparing them to the groundtruth. For the supervised- and unsupervised approach, the image pixel values are converted into float format and are then regarded as probabilities. For the fusion approach probabilities are computed using the

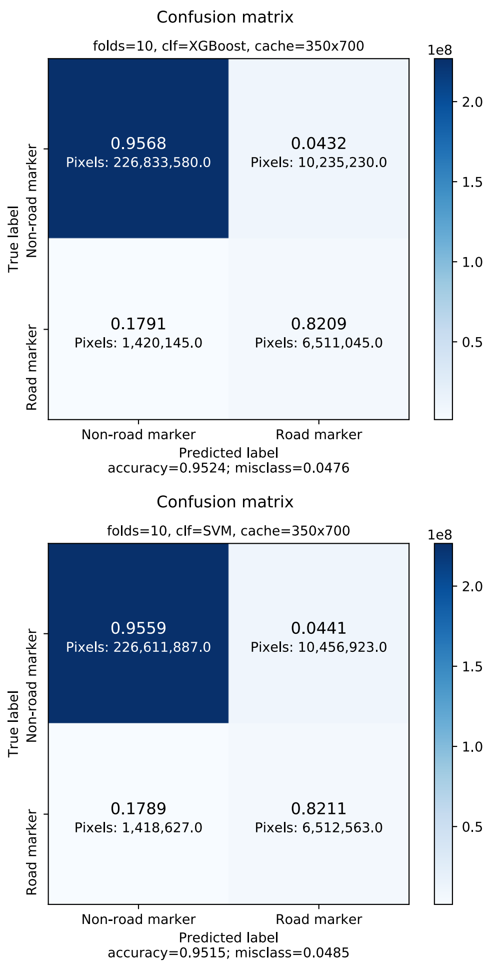


Figure 4.9: XGBoost- and SVM confusion matrices for the 175x350 cache using a 2-element feature vector.

trained classifier. The fusion approach metrics are taken from the experiment ran using a 2-element feature vector, using the combined effort of classifiers trained among a 10-fold cross-validation split.

In the charts of Figure 4.12, Figure 4.13, Figure 4.14 and Figure 4.15 are plotted the AUC score along with the ROC curve for each approach. It is visible that the supervised approach obtains an AUC score of 0.980, the unsupervised approach one of 0.943 and the fusion approach has two scores for its two different presets; 0.985 using a 2-element feature vector and 0.990 using a 5-element feature vector. This means that the fusion approach using XGBoost and a 5-element feature vector has the highest AUC score.

4.4 Pipeline performance

Pipeline performance is an important aspect in the perspective of scalability. The pipeline was optimized by enabling parallel-computing computations as much as possible. This includes caching multiple images simultaneously, training using a

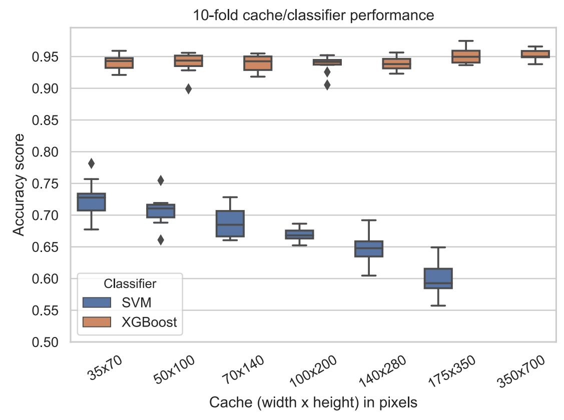


Figure 4.10: 5-element feature vector 10-fold prediction performance per pixel size configuration.

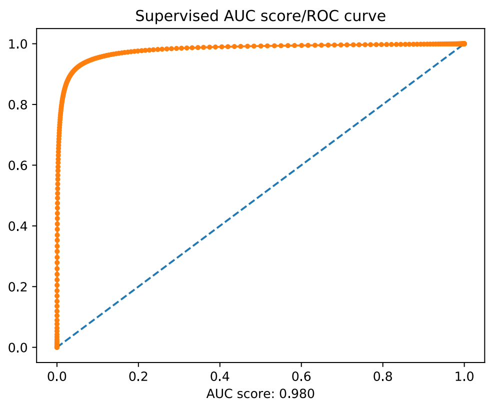


Figure 4.12: Supervised approach AUC score and ROC curve for the pixel configuration 175x350.

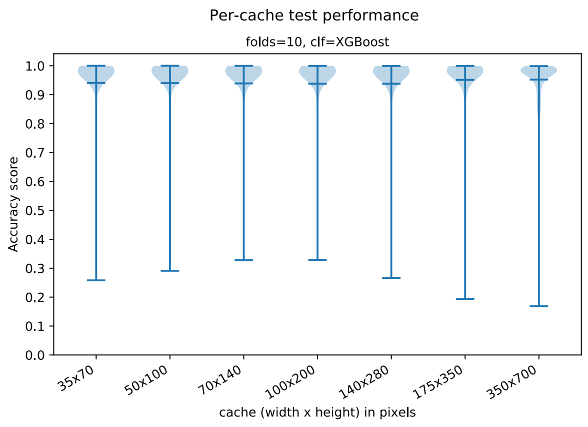


Figure 4.11: XGBoost accuracy performance for total test-set of 1000 images for fv=5-element.

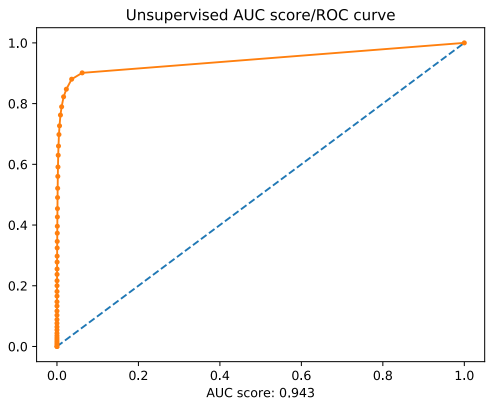


Figure 4.13: Unsupervised approach AUC score and ROC curve for the pixel configuration 175x350.

BaggingClassifier and thus enabling using multiple cores using $n\_jobs = -1$ and enabling parallel estimation again by setting $n\_jobs = -1$ on classifier instances. The pipeline performance was measured for the case where a 2-element feature vector was constructed.

The cache pipeline step executes at a reasonable speed, and at foremost- is a one-time process. Once images have been cached, different training- and testing configurations can be used without re-generating the cache. For this reason, only the transformation, training and testing pipeline steps were bench-marked, since they form the most integral part to the pipeline's performance. See Figure 4.16.

Visible in the figure (Figure 4.16), is that transformation performance decreases in somewhat linear fashion. This is because of the fact that image width and height in increasingly large pixel size configurations are not doubled, but increased by smaller increments. Transformation performance is very fast up to the 175x350 pixel size configuration level. After this, performance slows down fast.

Training times, however, remain rather constant across the entire pixel size range. This is because classifiers train upon a sampled data-set, for which the sample size remains constant across all pixel size configurations. It is visible though, that *SVM* takes longer to train than *XGBoost*.

Test times, however, are significantly impacted by pixel size configurations; for every increase in pixel size configuration, a significantly higher amount of pixel-values have to be predicted. Because of the great differences in test times, the pipeline test performance chart was depicted in a *logarithmic* scale. For the 525x1050 pixel size configuration, a *SVM* takes at least 158 seconds to predict 1 image, whilst *XGBoost* takes 3 seconds for the same case.

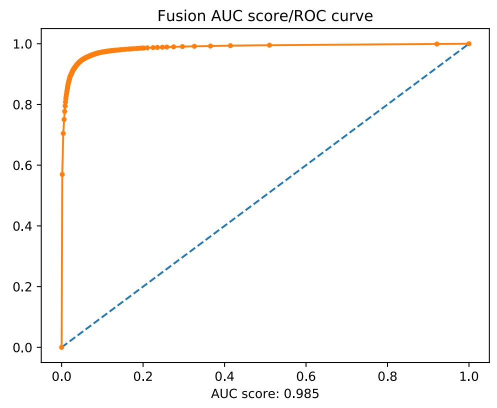


Figure 4.14: Fusion approach AUC score and ROC curve for the pixel configuration 175x350 using a 2-element feature vector.

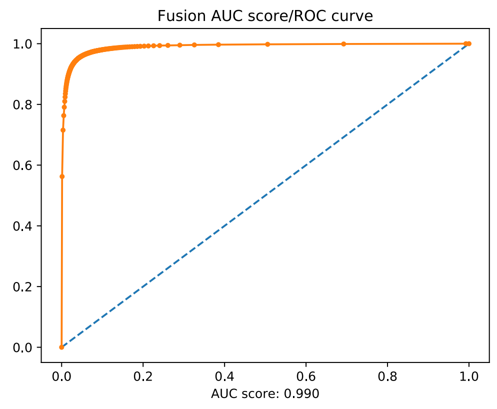


Figure 4.15: Fusion approach AUC score and ROC curve for the pixel configuration 175x350 using a 5-element feature vector.

5 Discussion

In the study, several circumstances that influenced test results might have been at play. Even though biasing has been attempted to be reduced as much as possible, by the reduction of over- or under-fitting through cross-validation for example, results most likely still have a certain bias.

The first point of critique might be of one parameter that remained constant throughout all tests, namely the *max_samples* parameter. The parameter determines the maximum amount of samples to be taken per image during transformation. First of all, sampling was done randomly; for each image, a random set of samples of both the road-marker and non- road-marker class were taken. Due to this randomness, when enough images are sampled, the total data array will be large enough to present a truthful representation of all different pixel values. The process of randomly sampling pixel in an image is repeated multiple times for an image

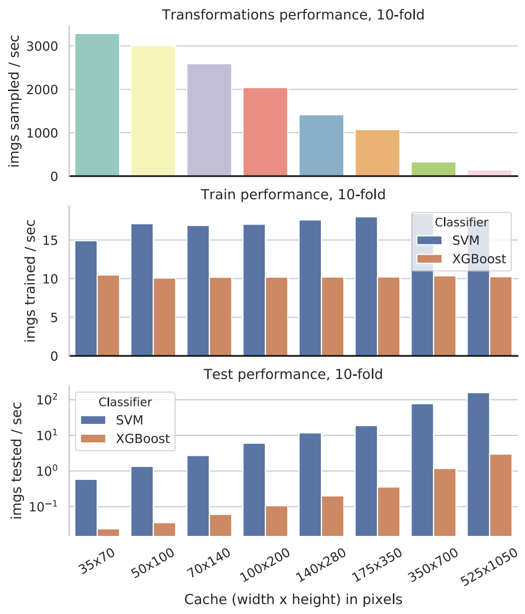


Figure 4.16: 2-element feature vector pipeline performance using an Intel Core i7-3820 Quad CPU @ 3.60GHz, running 64-bit Ubuntu 19.04 system at 8,0 GiB RAM.

too - using 10-fold cross-validation any one image is sampled differently at least 9 times.

Of course, however, increasing the maximum samples taken could lead to a more robustly trained classifier, which would have been trained on a larger distribution of pixel values. Yet, increasing maximum sample size would have a large effect on memory used during training, leading to an increase in training times accordingly. In the test-setup used in this study no high-performance computing systems were available, models were thus trained on a consumer-grade computer; testing the 350x700 pixel size configuration cache using SVM and a 2-element feature vector on the designated system took 21.5 hours at 77 seconds per image despite utilizing multiple cores. Testing the entire 525x1050 cache using SVM and a 2-element feature vector would prospectively take 43.8 hours. Nevertheless, possibly when higher-performance system is used, larger maximum sample sizes can be tested.

Also, in comparing the pixel size configuration for determining the best configuration, it was concluded that the best configuration was 350x700. Even though multiple different charts show the 350x700 configuration as the best-performing one, there can still exist a better pixel size configuration, at least for the 5-element feature vector preset. This is because in the 5-element feature vector pixel size configuration comparison chart, configurations only up to 350x700 were shown. XGBoost, however, showed a steady increase in performance with increasing pixel sizes. This can suggest that if larger pixel size configurations were tested in the 5-element feature vector scenario, its performance could have been higher.

The second point of critique is the fact that the problem has only been asserted by 2 classifiers, SVM and XGBoost. The study took focus on these two main classifiers in order to keep fit within the scope of the study. These two classifiers are generally pretty well performing at the given classification task at hand. Within a larger study scope, more classifiers could be tested on the problem to compare their performance, however, it has to be noted that to structurally improve performance, some other measure should probably be taken instead of only finding the most optimal classifier.

Finally, the given data-set might have presented an unrepresentative distribution of road-imaging, such that the trained classifier is not well fit under different circumstances of imaging. The data-set contained imaging with both complete sun-brightened imaging and partly-shadowed imaging. It does seem, however, that the data-set weather conditions remain somewhat constant throughout the entire data-set. For this reason, the fitted model might be less well suited for non-sunny road images. To improve the model's versatility to classifying different road conditions, a data-set with lots of different weather conditions is best to be used. Also, lastly, the model in this study was trained using 1000 images. Increasing the size of the data-set might prove to make for better classifying results and get an overall more versatile fit on the data.

6 Conclusions

In the study, a fusion algorithm between two existing algorithms was presented. The new fusion approach was then tested under several conditions. The entire pipeline was run using two different feature-vectors; a 2-element feature vector, combining the supervised- and unsupervised approach, and a 5-element feature vector, additionally adding in the input image RGB values. For every feature-vector setup, two different classifiers were tested, ran under different configurations of cached image pixel sizes. 8 different pixel size configurations were tested in total. In the 2-element feature vector pipeline, the 525x1050 configuration was only tested for *XGBoost*, because *SVM* was too slow to predict in this case, see Section 4.4. In the 5-element feature vector pipeline the 350x700 configuration was too slow to predict using SVM. The 525x1050 configuration was too large for both classifiers in a 5-element feature vector setup. All configurations were tested using a 10-fold cross-validation setup.

For both the 2-element feature vector and the 5-element feature vector accuracies are distributed across the [0.2, 0.99] range, with most distribution in the [0.93, 0.99] range, as seen in Figure 4.1 and Figure 4.11, respectively. For both feature vector presets the best performing pixel size configuration is 350x700, as can be seen in Figure 4.7 and Figure 4.10.

As was shown in Section 4.1, very low outlier accuracy scores for the 2-element feature vector approach showed correlation with imaging void of any road-markings. This means that the 2-element feature vector approach performs badly when presented with imaging of only road surface.

For both feature vectors presets, XGBoost outperformed SVM for almost all pixel size configurations. For the 2-element feature vector, SVM showed a median accuracy score of 0.9454 whilst XGBoost showed a median accuracy score of 0.9473 when combining all pixel size configurations, as shown in Figure 4.8. For the 5-element feature vector the difference was much greater and thus clearly visually visible; as shown in Figure 4.10, SVM median accuracy does not exceed 0.75, whilst XGBoost median accuracy score is no less than 0.90. XGBoost is the clear winner for the 5-element feature vector preset.

Comparing the AUC score / ROC curve of the supervised- and unsupervised approach as well as the fusion approach using its 2-element and 5-element feature vector preset, it becomes clear that the Fusion approach using a 5-element feature vector obtains the highest AUC score. AUC scores and ROC curves are visible in Figure 4.12, Figure 4.13, Figure 4.14 and Figure 4.15. Supervised and unsupervised AUC scoring are 0.980 and 0.943, respectively. This means that the supervised approach the highest AUC score of the existing two approaches. The 2-element feature vector fusion approach has an AUC score of 0.985, which is higher than both previous approaches. Finally, the 5-element feature vector fusion approach AUC score, however, is higher than all others, at 0.990. AUC/ROC measurement was made using the pixel size configuration of 175x350.

Thus, the best approach by AUC score is the 5-element feature vector fusion approach.

References

Oleksandr Bailo, Seokju Lee, Franois Rameau, Jae Shin Yoon, and Inso Kweon. Robust road marking detection and recognition using density-based grouping and machine learning techniques. 03 2017. doi:10.1109/WACV.2017.90.

Federico Castanedo. A review of data fusion techniques. *The Scientific World Journal*, 2013, 2013.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

I. M. Chira, A. Chibulcutean, and R. G. Danescu. Real-time detection of road markings for driving assistance applications. In *The 2010 International Conference on Computer Engineering Systems*, pages 158–163, Nov 2010. doi:10.1109/ICCES.2010.5674844.

John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90, 2007.

W. Kaddah, Y. Ouerhani, A. Alfalou, M. Desthieux, C. Brosseau, and C. Gutierrez. Road marking features extraction using the viapix system. *Optics Communications*, 371:117 – 127, 2016. ISSN 0030-4018. doi:https://doi.org/10.1016/j.optcom.2016.03.065. URL `http://www.sciencedirect.com/science/article/pii/S003040181630236X`.

Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.

Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.

Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

Gaël Varoquaux and O Grisel. Joblib: running python function as pipeline jobs. *packages. python. org/joblib*, 2009.

A Appendix

Link to Github repository with algorithm code, access granted to selected users.
`https://github.com/dunnkers/bsc-thesis`

A.1 More prediction results

Predictions between the 2-element and 5-element feature vector presets are generally quite similar, but sometimes differ. Figures A.3 and A.4 show a comparison.

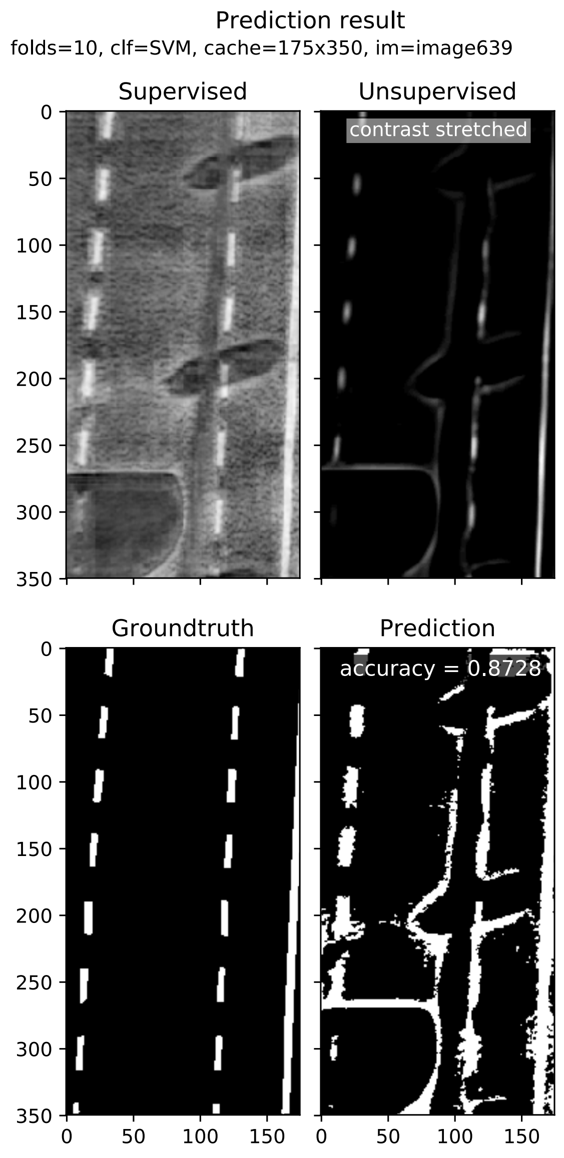


Figure A.1: Shadows on the road proves a hard challenge for the trained SVM classifier. 2-element feature vector.

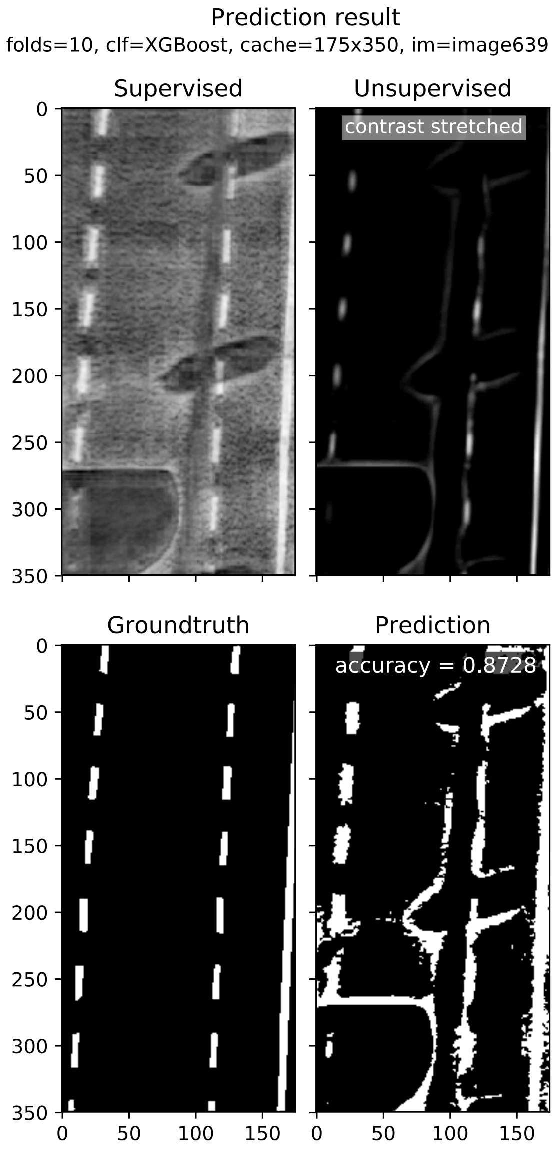


Figure A.2: XGBoost shows similarly bad performance when shadows are existent. 2-element feature vector.

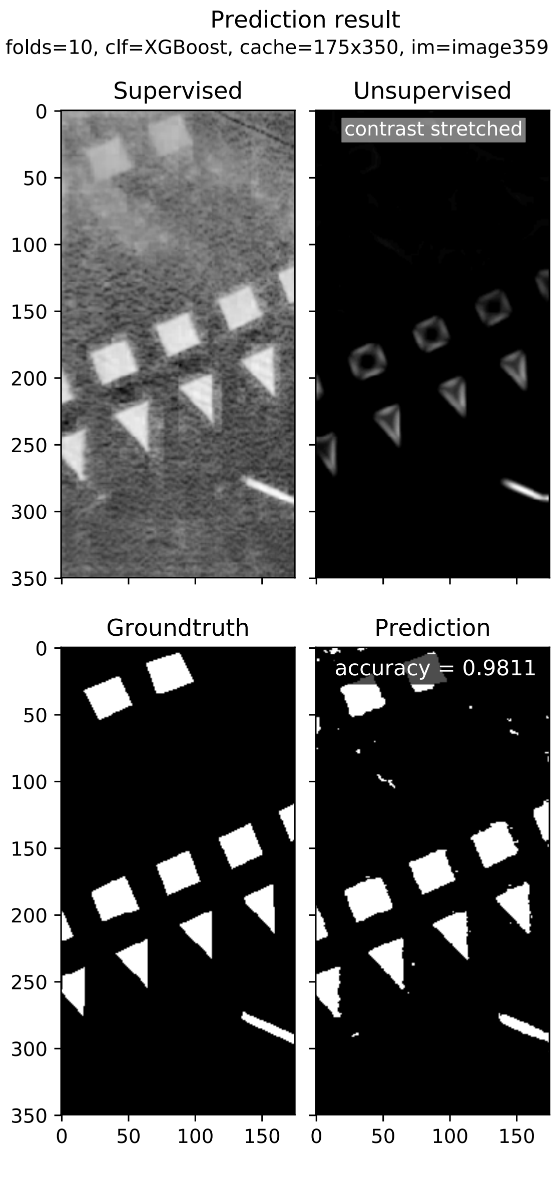


Figure A.3: 5-element feature vector prediction.

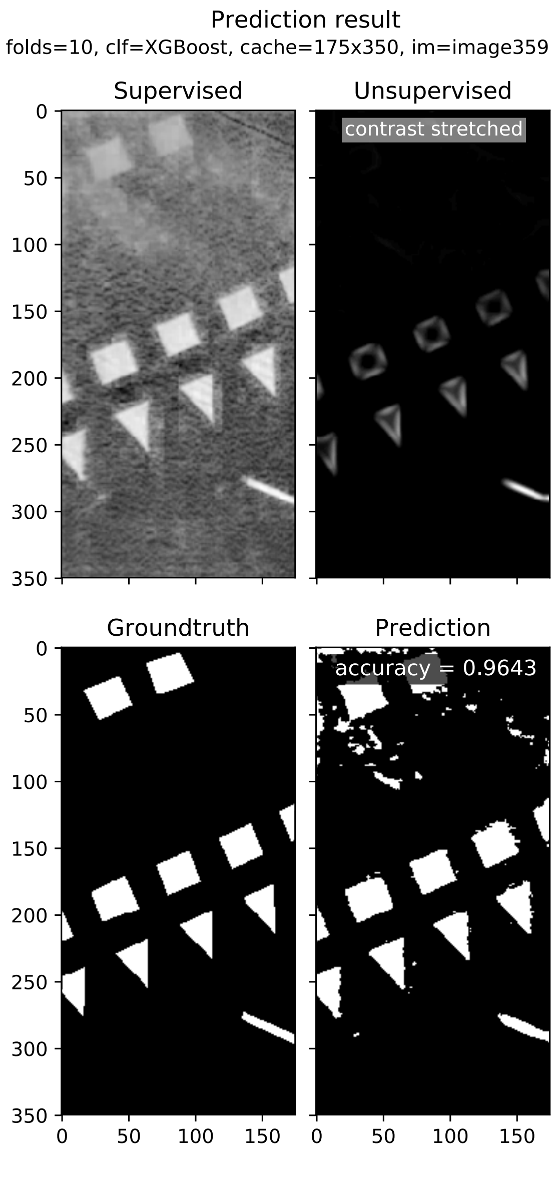
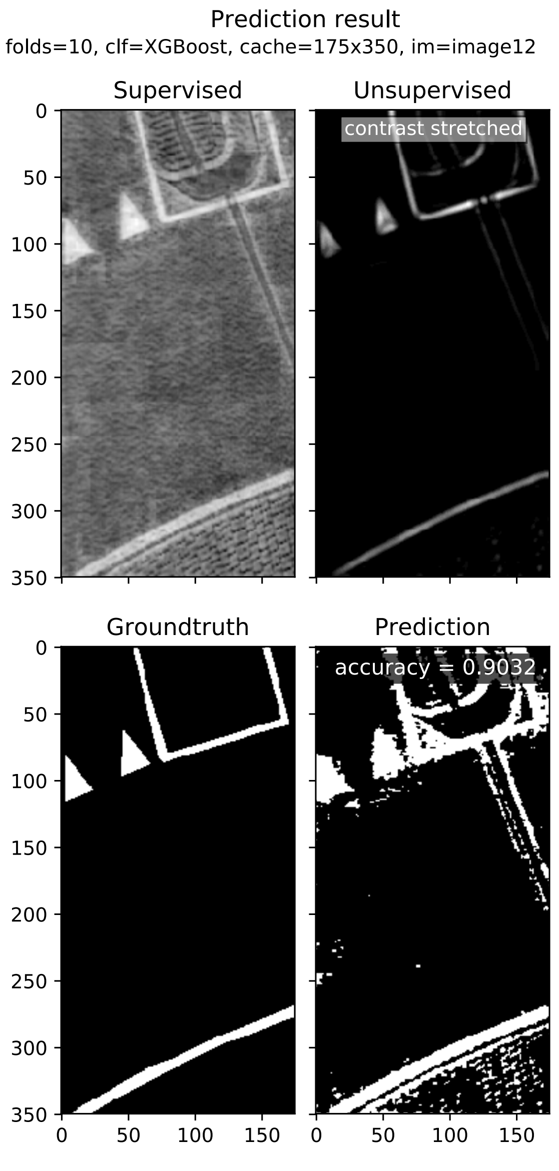
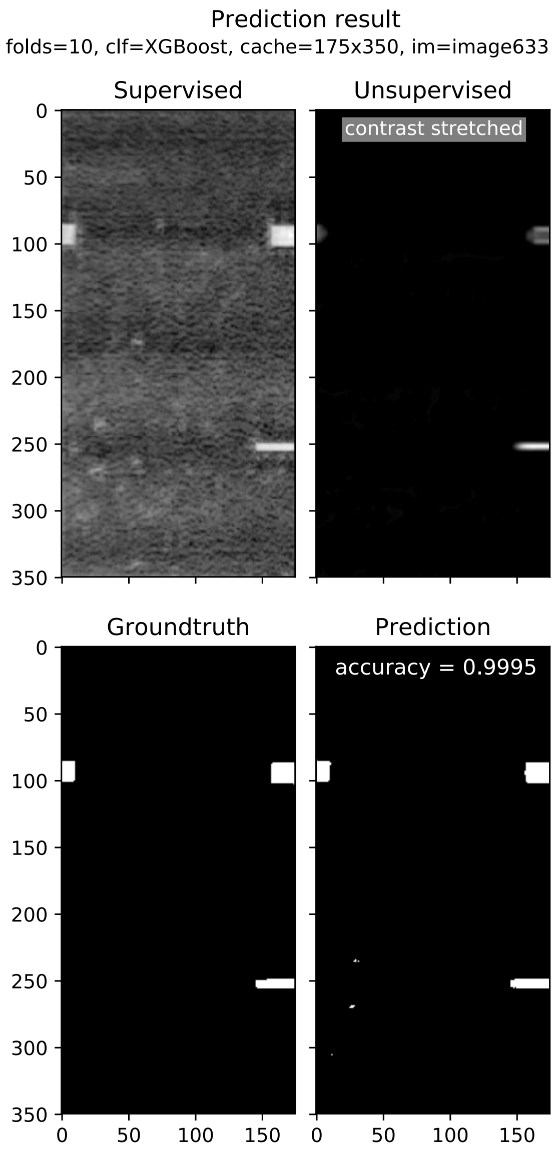
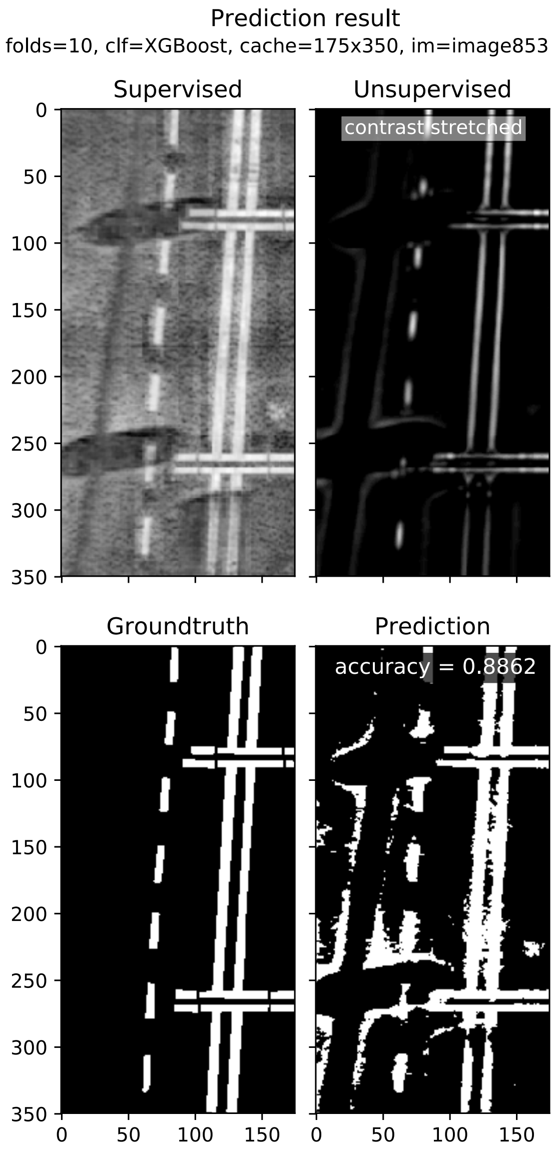


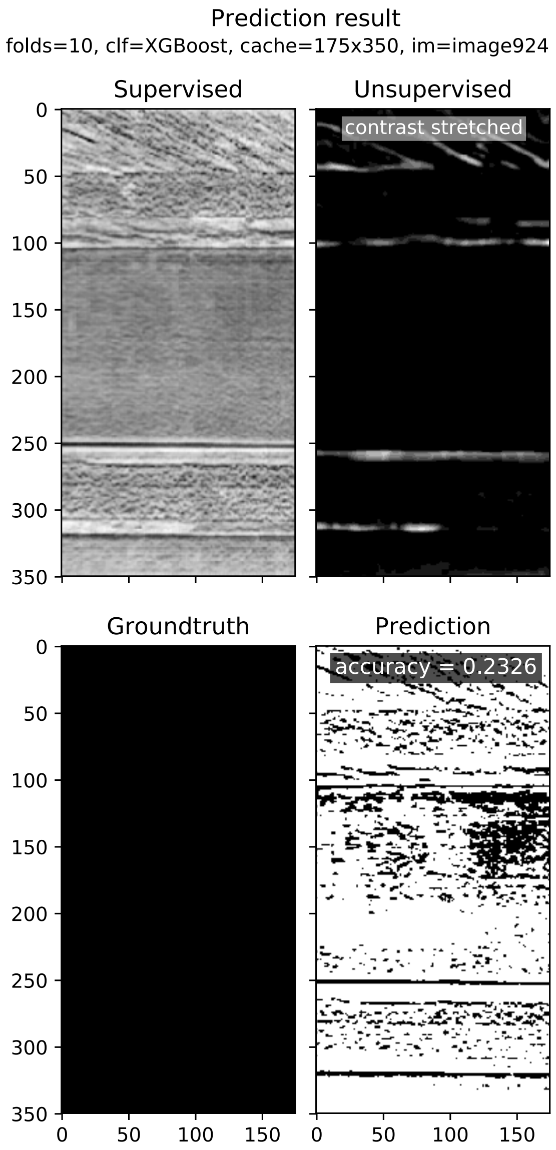
Figure A.4: 2-element feature vector prediction.


Prediction result
folds=10, clf=XGBoost, cache=175x350, im=image12
Supervised
Unsupervised
contrast stretched
Groundtruth
Prediction
accuracy = 0.9032
0
50
100
150
200
250
300
350


Prediction result
folds=10, clf=XGBoost, cache=175x350, im=image633
Supervised
Unsupervised
contrast stretched
Groundtruth
Prediction
accuracy = 0.9995
0
50
100
150
200
250
300
350



Prediction result
folds=10, clf=XGBoost, cache=175x350, im=image853
Supervised
Unsupervised
contrast stretched
Groundtruth
Prediction
accuracy = 0.8862



Prediction result
folds=10, clf=XGBoost, cache=175x350, im=image924
Supervised
Unsupervised
contrast stretched
Groundtruth
Prediction
accuracy = 0.2326