Vehicle Classification from Inductive Loop Signature for Embedded Application.

Stéphane B. Bourgeois and Fabio Cuzzolin
Department of Computing and Communication Technologies, Oxford Brookes University
Oxford, United-Kingdom
12085745@brookes.ac.uk, fabio.cuzzolin@brookes.ac.uk

***Abstract*— This paper presents applications of machine-learning algorithms to classify vehicles from inductive loop signals. Our contribution is to achieve a good performance per individual class for 9 classes of vehicles, whilst earlier studies have targeted 4 or 5 classes only. We compare the performance of Adaptive Boosting with Support Vector Machines (SVM). We show that SVMs with the Gaussian Kernel can deliver an f1-score between 98% and 83% a per class. We show that the Synthetic Minority Oversampling Technique (SMOTE) improves the performance of the polynomial kernel. We estimate the computational complexity of the classifier, and show that our solution is usable on an embedded platform that delivers in the order of 10 MIPS.**

Keywords—Inductive loop (IL); Support vector machine (SVM); class imbalance; embedded system; vehicle classification.

I. INTRODUCTION

Inductive loops (IL) are sensors that are widely deployed on road networks for the purpose of traffic data collection. Our aim is to classify vehicles in a 10 category scheme, such as the SWISS10, from dual loop signals. Furthermore, commercial solutions run on embedded platforms that deliver in the order of 10 MIPS, using fixed-point arithmetic. We want to assess which solution is viable on such platforms.

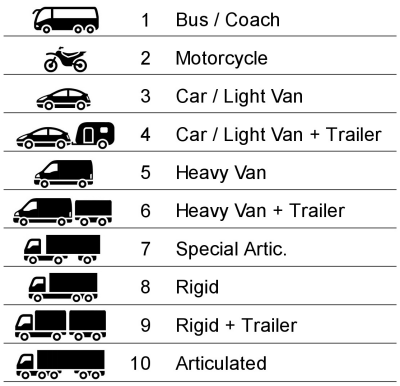

Fig 1: SWISS10 vehicle classes

Earlier studies are aimed at 4 or 5 classes of vehicles, and rely on other sensors such as single IL or Anisotropic Magneto-resistive (AMR) sensors.

We compare two machine-learning algorithms: Support Vector Machines and Adaptive Boosting with decision stumps. We use the two most common algorithms for multi-class classification, One-versus-One and One-versus-Rest. In order to address class-imbalance, we are asses how our methods respond to undersampling, oversampling and Synthetic Minority Oversampling Technique (SMOTE)[1].

In our study, Support Vector Machines with the Gaussian Kernel delivered the best results, with the smallest difference of performance between classes. An average f1-score per class of 90% was achieved with a maximum of 95% and a minimum of 85%. The SMOTE algorithm proved useful to detect outliers and improve the results with the polynomial kernel.

A prototype implementation in fixed-point arithmetic demonstrated that sufficient resolution can be obtained using 32 bits resolution for variables, and 64 bits for multiply-accumulate registers. Finally, an estimate of the ARM assembly and CPU cycle count was produced for a classic ARM platform running at 8MHz. The computation time would be in the order of 120ms per vehicle, which makes it practical for industrial use.

II. BACKGROUND

A. Inductive Loop Detector

Inductive Loop Detectors are based on the change of inductance that occurs when a conductive object is placed in the proximity of a coil. The loops are driven by an LC oscillator whose frequency changes when vehicles pass over the loop.

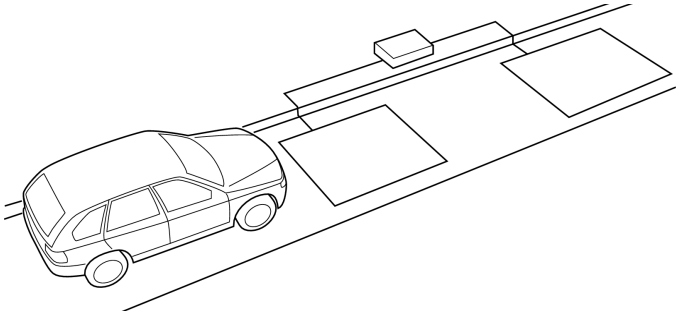

Fig. 2: Inductive loop road installation

A dual loop installation allows to estimate the speed and length of the vehicle. The dimensions of the loops is around 2x2 meters, which has a low-pass effect on the signal. The

waveform produced is characteristic of the vehicle, as shown in the samples below.

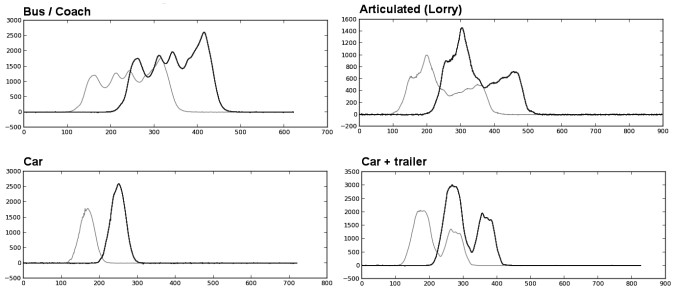

Fig. 3: Inductive Loop waveforms

B. Related work

Vehicle classification is a small player in the field of machine learning. A majority of the literature is based on single loop detectors and Artificial Neural Networks. The classification accuracy is around 90%, for 4 to 5 vehicle classes. Although using a different sensor, articles using Anisotropic Magneto-resistive Sensors (AMR) are of interest for their use of machine-learning techniques.

In 2006, Ki & Baik extract 19 values from the frequency waveform of a single-loop detector, to be fed into the input layer of a Back-propagation Neural Network (BPNN). They achieve an average detection rate of 91.5% for 5 classes of vehicles (Car, Van, Bus, Truck, Motorcycle)[2]. This paper shows that good results can be obtained with a small input-vector size and a minimal amount of pre-processing.

In 2010, Kaewkamnerd et al. target an embedded platform, built around the Texas Instruments MSP430 16-bits microcontroller. Kaewkamnerd's team are using two AMR sensors, and a decision tree classifier, based on low complexity features of the signal such as relative vehicle length, hill-pattern peak and distribution of energy.. The average accuracy is around 82% for 4 vehicle classes [3].

In 2009, Feng and Minghze achieve better classification rates from AMR sensors, using Support Vector Machines arranged in a binary decision tree. Classification rates for large-truck, bus, van and car categories are respectively 86%, 80%, 81% and 89% [4].

In 2010, Meta & Cinsdikici derive two types of parameters from inductive loop signals: (1) Principal component analysis (PCA) from which they retain 16 eigenvectors, (2) Statistical parameters derived from the waveform: median, mean, number of local maxima. A vector composed of the 16 PCA components and count of local maxima is fed into a three layer back-propagation neural network. The average recall rate for 5 vehicle classes is 94.2% [5] .

We note a widespread use of artificial neural networks by other authors [6], [7] .This could be as most of the articles predate the release of the open-source LIBSVM library, in 2011 [8] .

C. Properties of the dataset

We were provided with data collected from around 2000 vehicles. The loop data was sampled at 500Hz and manual classification was performed from video imagery. We ignored the motorcycle category as only 2 samples were available. The dataset is unbalanced with a maximum ratio of 200:1 between the Cars and Bus / Coach categories.

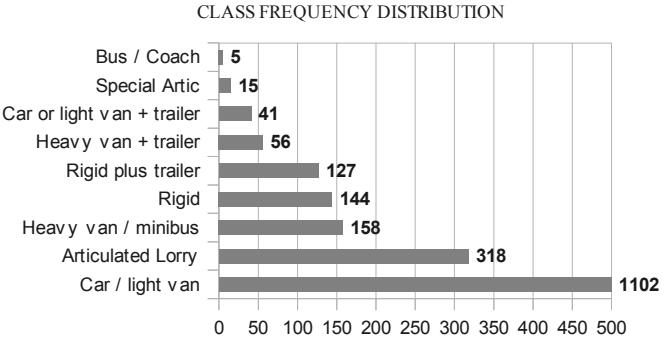

Fig. 4: Class frequency distribution

III. METHODOLOGY

A. Choice of algorithms

We focused on two classification algorithms, Support Vector Machines and Adaptive Boosting.

Neither algorithms are scale-invariant. We therefore detect the start and stop of the signal and conduct a simple scaling of the data. Following the approach of Ki & Baik [2], we down-sample the waveform to a set number of samples and derive an estimate of the vehicle length from the signal time-delay between the two loops. This data can be used directly as the input vector for a Support Vector Machine classifier.

Adaptive Boosting is a meta-algorithm which improves the performance of a large number of simple classifiers. We used decision stumps as our base classifier. However we compared two pre-processing methods:

- First method: derive statistical features from the signal and add those to the waveform. This follows the approach of Kaewkamnerd et al. [3].
- Second method: produce a pool of random discriminants from the normalised waveform.

We used both SVM and AdaBoost as binary classifiers and compared the results of one-versus-all using winner-takes-all strategy and the one-versus-one with max-wins voting [9] .

As we have an unbalanced dataset, we tested a number of class-imbalance techniques: undersampling, oversampling and Synthetic Minority Oversampling Technique (SMOTE) [1] . Our approach was to:

- review the performance of Support Vector Machines and AdaBoost over a range of parameters and pre-processing methods.
- Narrow our choices to the best performing solutions, and apply class-imbalance techniques to those best performers.

B. Cross-validation and performance metric

Due to the low number of Bus / Coach samples we used the Stratified Shuffle Split method for cross-validation. This method ensures that the ratio of samples in the training set versus the testing set is the same for all classes. We used 1000 iterations and a training set of 80%.

As noted by Nitesh Chawla, with unbalanced data sets, precision and recall can vary in conflicting directions [10] . We therefore use the f1-score to assess the performance of a classifier. (The f1-score is the harmonic mean of the precision and the recall rates.)

C. Properties of the signal

An analysis of the frequency response for various vehicles shows that the signal decays in frequency with a cutoff point around 30Hz. This value depends on the speed and type of vehicle. However it indicates that we can down-sample to around 50 samples between the start and stop of the signal whilst remaining within the Nyquist frequency limit.

Once normalised in amplitude, the difference between the signal of the two loops is marginal, and may be caused by vehicle acceleration. We only retain the samples from the loop with the highest signal amplitude.

The signal-to-noise ratio is around 32dB which is equivalent to 11 to 12 bits per sample.

IV. Implementation and Experimental Results

A. Software libraries and programming environment

We are using the Python programming language with the NumPy and Scipy libraries for scientific computation and signal-processing [11]. We are using Scikit-learn as our machine-learning library [12] .

B. Pre-processing

We use a common pre-processing stage for both SVM and Adaboost classification.

1. Remove DC component in the signal
2. Normalise both loop signals by max. amplitude
3. Detect start /stop at 10% amplitude threshold. Resolve errors caused by trace signals from a preceding or following vehicle.
4. Compute an estimate of the vehicle length from the average time-delay between two loops
5. Down-sample the channel with the highest signal amplitude to a set number of samples.

C. Support Vector Machines

A number of parameters will affect the performance of Support Vector Machines. We assessed the impact of the following:

1. Timing of start/stop of the signal. This can be decided on a single loop, globally for the two loops, or scaled to vehicle length with zeros inserted for the non-vehicle part.
2. Number of signal samples in the input-vector after down-sampling. (128, 64, 32 or 16 samples)
3. Kernel choice: Gaussian (RBF), Polynomial (degree 3 or 2), Linear.
4. Multi-class method: One-versus-all or One-versus-one

For each of those choices, we tune the soft margin parameter C, and the gamma value for any non-linear kernel. We plot the unweighted average f1-score per class, and the minimum f1. We show here an example of such performance comparison.

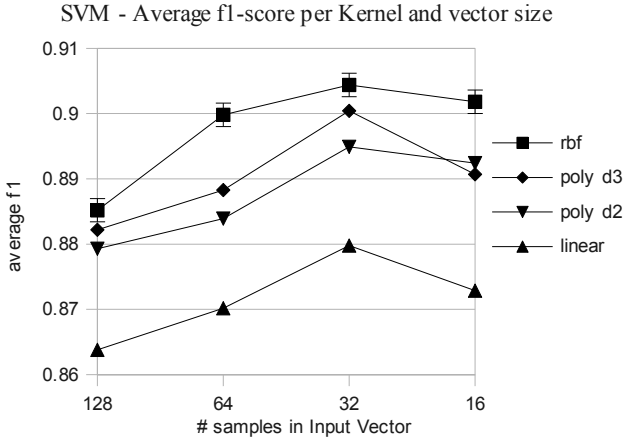

Fig. 5: Support Vector Machine, influence of Kernel function

For the SVM classifier, the best results are obtained with the following choices:

- Use the start/stop timings of a single channel.
- Downsample the signal to 32 samples
- Gaussian (RBF) and degree 3 polynomial kernels perform best.
- For the polynomial kernel, One-versus-one performs better than One-versus-all

We summarize our results in the following table

TABLE I. Support Vector Machine parameters / performance

# samples	kernel	C	gamma	multi cl	f1-avg	f1-min
32	RBF	500	1.4	OvR	0.902	0.807
32	RBF	500	1.4	OvO	0.899	0.804
32	Poly d=3	500	0.1	OvO	0.895	0.764

D. Adaptive Boosting

Adaptive Boosting is a meta-algorithm that improves the performance of a large number of base classifiers. A commonly used base classifier is a decision stump, which looks at a single component of the input vector $x=(x_1, x_2, \ldots, x_n)$

to produce a classifier $h(x,\theta)=sign(ax_k-b)$

where the training parameters are $\theta=\{k, a, b\}$

Using decision stumps, our first approach is to form an input vector with the signal waveform and statistical properties of the

signal. For the statistical properties we divide the signal in 3 regions of equal duration, and calculate the minimum, maximum, mean, variance, skewness and kurtosis for each region.

Our second approach is to use a pool of random linear discriminants as base classifiers. Using the above notation for the input vector, we derive a classifier from two components

$$h(x,\theta)=sign(u(x_j-x_k)-b)$$

where the training parameters are

$$\theta=\{j,k,u,b\} \quad \text{and} \quad u\in\{-1,+1\}$$

and the indices j,k are chosen from a set of random values.

The best results were obtained with the One-versus-one method and the following settings:

TABLE II. ADABOOST PARAMETERS / PERFORMANCE

Classifier	samples	IV length	# estim	L. rate	f1-avg	f1-min
Signal + statistics	64	82	128	0.5	0.869	0.636
Linear discriminants	64	256	256	1	0.853	0.656

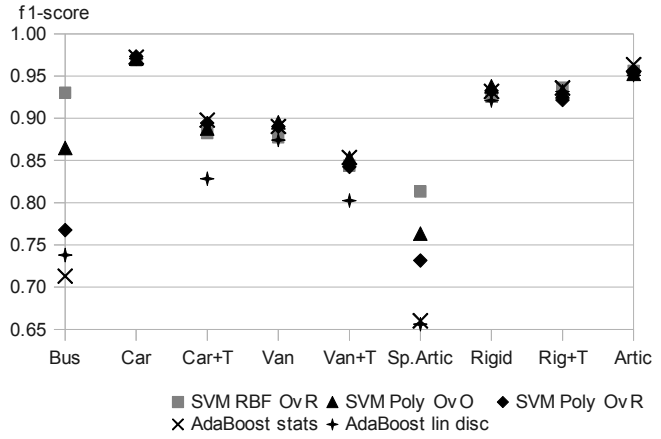

Fig. 6: AdaBoost vs. SVM. Compared f1-score per vehicle class.

In figure 6, we compare the results with the Support Vector Machines by plotting the f1-score per class. We can see that the SVM classifier performs better for minority classes.

E. Class-imbalance

To reduce the effect of class imbalance, the previous results were obtained by limiting the number of cars to the first 500 samples. This is a crude form of random undersampling. The following experiments are based on the full dataset. At this stage we focused on Support Vector Machines

We tested the effect of undersampling, oversampling, Synthetic Minority Oversampling Technique (SMOTE), and the combination of undersampling with either oversampling or SMOTE. Undersampling was applied to the Cars category with ratios of 1:2 or 1:4. Oversampling and SMOTE were applied to the Bus (class 1) and Special Artic. (class 7) to increase the number of samples to approximately 100.

The SMOTE algorithm is an oversampling technique that calculates synthetic samples by linear interpolation between the nearest vectors, selected using the k-Nearest neighbours (kNN) algorithm. Low values of kNN, around 1 and 2 appear to perform best.

The first results were disappointing and lead us to discover outliers in the manual classification. It is well-known random oversampling or SMOTE are sensitive to outliers, as these techniques will duplicate existing errors [13] .

Our experimental results show that the performance of the polynomial kernel is significantly improved by the combination of undersampling and SMOTE. This result is of interest with regards to the computational cost of an embedded implementation.

TABLE III. SVM CLASS IMBALANCE PARAMETERS

Kernel	Under-sampling	Over-sampling	C	gamma	f1-avg	f1-min
RBF OvR		Class 1, 4, 6, 7	500	1.4	0.908	0.834
Poly OvO	1:2	SMOTE 1 & 7	50	0.125	0.899	0.799
Poly OvR	1:2	SMOTE 1 & 7	5	0.25	0.896	0.802

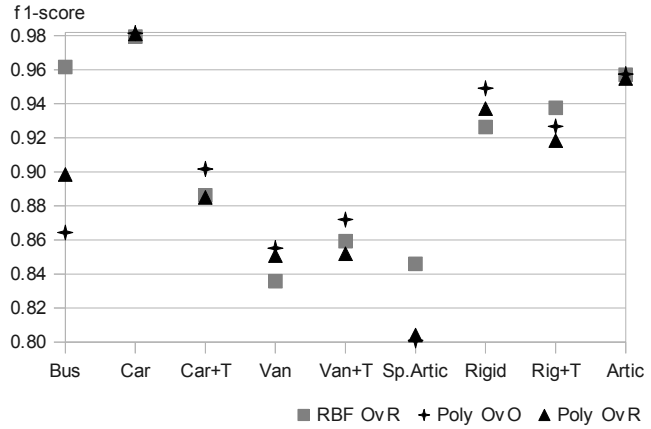

Fig. 7: SVM with class imbalance techniques. f1-score per class.

We show complete numerical results in table IV.

TABLE IV. SVM PERFORMANCE PER VEHICLE CLASS.

Support Vector Machine, Gaussian Kernel, One vs. Rest

Stratified Shuffle Split, training 80%, test 20%, 1000 iterations

	precision	recall	f1-score
Bus	96%	81%	88%
Car	99%	96%	98%
Car + Trailer	86%	91%	89%
Heavy van	76%	93%	83%
Heavy van + Trailer	87%	82%	84%
Special Artic	92%	76%	83%
Rigid	93%	93%	93%
Rigid + Trailer	94%	92%	93%
Articulated	95%	96%	95%
Average weighted	95%	95%	95%
Average unweighted	91%	89%	90%

F. Comparison with published results

We will now compare our results with those published in previous works. Most authors don't use the f1-score and classify into 4 or 5 classes of vehicles: Bus, Car, Van and Truck, the fifth class being Motorcycles.

Meta & Cinsdikici [5] and Ki & Baik [2] publish recall values, as well as a confusion matrix. We assumed that Feng & Minzhe [4], who use the term "classification correct" for their metric, also report recall values.

For comparison, we mapped the SWISS10 categories as follows and calculated a weighted average of the recall rate obtained with our method.

SWISS 10	Literature
Bus	Bus
Car	Car
Car + Trailer	
Heavy van	Van
Heavy van + Trailer	
Special Artic	Trucks
Rigid	
Rigid + Trailer	
Articulated	

TABLE V. COMPARISON WITH PUBLISHED RESULTS

	Our results	Meta & Cinsdikici	Feng & Mingzhe	Ki & Baik
	2014	2010	2009	2006
Sensor	**2 ILD**	2 ILD	AMR	1 ILD
Algorithm	**SVM**	PCA + BPNN	SVM	BPNN
		recall		
Bus	**81.0%**	100.0%	80.0%	91.6%
Car	**95.8%**	98.0%	89.0%	92.0%
Van	**90.2%**	89.7%	81.0%	79.4%
Trucks	**94.0%**	92.8%	86.0%	100.0%
Average	**90.3%**	95.1%	84.0%	90.8%

V. ESTIMATE OF COMPUTATIONAL COST

Vehicle classification is performed on low-power embedded systems which deliver in the order of 10 MIPS. Power consumption is a decisive criteria in the design of such equipment, as many installations are solar powered. We will evaluate the computation time on Classic ARM, and on a more modern Cortex-M4 architecture, at 8MHz clock speed.

These processors don't offer floating point units. We produce an implementation of the classifier in fixed-point arithmetic in order to assess the required resolution and derive an instructions count in ARM Assembly language.

A. Fixed-point implementation and resolution

We implemented the decision function for a Support Vector Machine, with a polynomial kernel, for two classes only. This implementation is incomplete, but is sufficient to assess issues of precision and computational cost. We ran fixed-point calculations in parallel with IEEE double precision arithmetic in order to assess the range of the variables and the error propagation in fixed-point.

The SVM decision function, for two classes, can be expressed as follows

$$f(x)=\sum_i \alpha_i y_i K(x_i, x)+b$$

where x_i are the support vectors, $y_i \in \{-1,1\}$ represent class membership, α_i and b are calculated at the training stage, and K is the kernel function [14] .

This function suffers from cancellation effect. The average output amplitude for the decision function is around 10, whilst the range required for the accumulator is around 10^5 (2^{18}). A 32 bits implementation failed to provide sufficient resolution, good results were obtained with 32 bits variables and 64 bits for the multiply-accumulate register.

B. Estimate of ARM CPU cycle count

The best approach to measure CPU timings is to implement the classifier in C or assembly language and execute the code on a development board. We limited ourselves to estimate the number of assembly instructions by derivation. We used Sloss & Wright's "Instruction Cycle Timings" [15] and the "Cortex-M4 Technical Reference Manual" [16] to derive a CPU cycle count from the assembly language instructions.

The code is built form 3 subroutines: 1. Kernel computation, 2. SVM decision function, 3. Multi-class decision with voting. The assembly instructions and CPU cycle count is performed in a spreadsheet from following parameters: input vector size, average number of support vector per class, number of classes, CPU cycle count of multiplications, number of terms in the Taylor expansion of the exponential function

Our estimates are based on an average of 200 support vectors per class and an approximation of the exponential with a 22 terms Taylor expansion. Our estimates are reported in the following table.

TABLE VI. ESTIMATED COMPUTATION TIME ON ARM ARCHITECTURE

Architecture	**Polynomial Kernel**		**RBF Kernel**	
Classic ARM (8MHz)	k CPU cycles	536	k CPU cycles	942
	ms	70	ms	120
Cortex-M4	k CPU cycles	275	k CPU cycles	548
	ms	35	ms	70

VI. CONCLUSION AND FURTHER WORK

A. Conclusion

We have compared the performance of machine learning algorithms in order to classify vehicles in a 9 category scheme, within the SWISS-10. In this comparison, Support Vector Machines gave better results than Adaptive Boosting, the best results being obtained with a Gaussian Kernel.

When reduced to 4 categories, our results are comparable in performance with previous publications. On 9 categories, the Gaussian Kernel delivers an f1-score between 98% and 83% a per class, with an average of 90%.

A combination of undersampling and Synthetic Minority Oversampling Technique (SMOTE) significantly improves the results obtained with a polynomial kernel. As a side-effect, the SMOTE algorithm will indicate the presence of outliers in the affected class.

We have estimated that the computation time of an SVM classifier with a Gaussian Kernel on ARMv4 architecture at 8Mhz is in the order of 120ms. Although this estimate is obtained from derivation rather than implementation in Assembly Language, it indicates the potential for running the classifier on an embedded platform for multi-lane classification.

We have also shown that the classifier can be implemented in fixed-point arithmetic, with sufficient resolution, provided the multiply-accumulate registers are 64 bits wide.

B. *Further work*

Class imbalance issues and performance should be assessed with a larger dataset.

We should derive a CPU execution time from an implementation executable on ARM architecture. This can be combined with a refinement of the fixed-point implementation.

Outlier detection would be of immense benefit to verify the results of the manual classification used for the training set.

ACKNOWLEDGMENT

The authors would like to thank Clearview Traffic Limited for providing the data used in this paper.

REFERENCES

[1] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, Jan. 2002.

[2] Y.-K. Ki and D.-K. Baik, "Vehicle-Classification Algorithm for Single-Loop Detectors Using Neural Networks," *IEEE Trans. Veh. Technol.*, vol. 55, no. 6, pp. 1704–1711, Nov. 2006.

[3] S. Kaewkamnerd, J. Chinrungrueng, R. Pongthornseri, and S. Dumnin, "Vehicle classification based on magnetic sensor signal," in *The 2010 IEEE International Conference on Information and Automation*, 2010, pp. 935–939.

[4] Z. Feng and W. Mingzhe, "A New SVM Algorithm and AMR Sensor Based Vehicle Classification," in *2009 Second International Conference on Intelligent Computation Technology and Automation*, 2009, vol. 2, pp. 421–425.

[5] S. Meta and M. G. Cinsdikici, "Vehicle-Classification Algorithm Based on Component Analysis for Single-Loop Inductive Detector," *IEEE Trans. Veh. Technol.*, vol. 59, no. 6, pp. 2795–2805, Jul. 2010.

[6] P. Bajaj, P. Sharma, and A. Deshmukh, "Vehicle Classification for Single Loop Detector with Neural Genetic Controller: A Design Approach," in *2007 IEEE Intelligent Transportation Systems Conference*, 2007, pp. 721–725.

[7] H. A. Oliveira, F. R. Barbosa, O. M. Almeida, and A. P. S. Braga, "A vehicle classification based on inductive loop detectors using artificial neural networks," in *2010 9th IEEE/IAS International Conference on Industry Applications - INDUSCON 2010*, 2010, pp. 1–6.

[8] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.

[9] R. Rifkin, "Multiclass Classification - 9.520 Class 06." Massachusetts Institute of Technology, 2008.

[10] N. V. Chawla, "Data Mining for Imbalanced Datasets: An Overview," in in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Springer US, 2005, pp. 853–867.

[11] T. E. Oliphant, "Python for Scientific Computing," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 10–20, 2007.

[12] 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," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[13] R. Batuwita and V. Paladey, "Class Imbalance Learning Methods for Support Vector Machines," in in *Imbalanced Learning: Foundations, Algorithms, and Applications*, H. He and Y. Ma, Eds. Wiley-IEEE Press, 2013.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.

[15] A. Sloss, D. Symes, and C. Wright, "Instruction Cycle Timings," in in *Computer Organization and Design: The Hardware/Software Interface*, 5th ed., Morgan Kaufmann, 2013.

[16] ARM Ltd., "Cortex-M4 Technical Reference Manual.," 2014. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0439b/CHDDIGAC.html. [Accessed: 20-Jul-2014].