ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

Data Analytics for Drilling Operational States Classifications

Galina V. Veres and Zoheir A. Sabeur *

University of Southampton IT Innovation Center - Electronics and Computer Science, Faculty of Physical Sciences and Engineering,
Gamma House, Enterprise Road, Southampton, SO16 7NS, UK

Abstract. This paper provides benchmarks for the identification of best performance classifiers for the detection of operational states in industrial drilling operations. Multiple scenarios for the detection of the operational states are tested on a rig with various drilling wells. Drilling data are extremely challenging due to their non-linear and stochastic natures, notwithstanding the embedded noise in them and unbalancing. Nevertheless, there is a possibility to deploy robust classifiers to overcome such challenges and achieve good automated detection of states. Three classifiers with best classification rates of drilling operational states were identified in this study.

1 Introduction

Offshore industrial engineering involves the management of highly complex operations in drilling rigs. Critical situations such as "*Kick*s", "*Fluid loss*" or "*Stuck pipe*" may occur during drilling operations. Such conditions are gradually reached following various stages of criticalities in time. Therefore, it is important that those stages of operations are detected and controlled during drilling processes. One way of achieving it is to automate the detection of drilling Operational States (OS). It involves the breaking of a drilling process into ten well-defined and exclusive drilling OS [1]: *1) Drilling Rotary (DrlRot); 2) Drilling sliding (DrlSld); 3) Clean Downwards (CleanDN); 4) Clean Upwards (CleanUP); 5) Wash Upwards (WashUP); 6) Wash Downwards (WashDN); 7) Move in hole (MoveDN); 8) Move out of hole (MoveUP); 9) Circulation on (CirclHL); and 10) Make Connection (MakeCN)*. The OS have been successfully detected on a drilling run using machine learning techniques with five additional principal states [1]. Further, Echo State Networks were adjusted to cope with unbalanced datasets in order to perform well in the classification of OS at a given well [2]. However, knowledge of labeled data for training was assumed to be available during the drilling process. Therefore, the challenge is to consider a real operational scenario which considers a drilling plan at multiple wells when labelled data becomes available after drilling, at least in one well on the rig.

* This work was partly supported by the European Commission FP7 IST Programme under contract TRIDEC IP FP7-258723.

A framework for the selection of the best performing OS classifiers is proposed in this study. The classifiers are trained by using a portion of available labeled data of OS, i.e. the training is done for a given well, while testing is performed on other wells for the detection of unseen OS.

2 Data analytics

Two types of drilling data are generated: Sensor *measurements* data (time series); and data created by drilling experts as *observations*, so-called OS labels. The analyses of both observations and measurements data are addressed in this section.

2.1 Measurement data analyses (Time series)

Measurement data are generated by sensing devices. The data exhibit *complex behaviour* (Fig. 1.) which led to further data analysis for selecting suitable classifiers. The complexity of the drilling time series (data behaviour) required two tests in the classification process: *Linearity* and *Normality* tests.

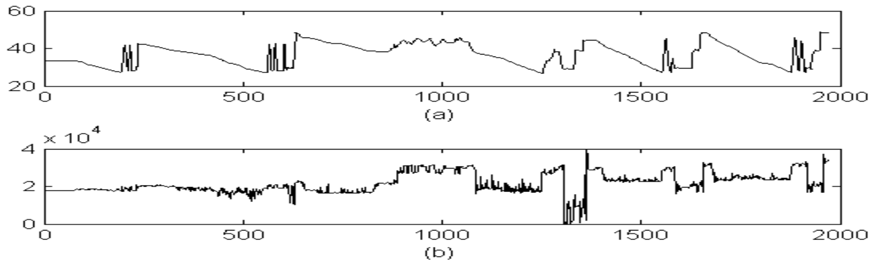

Fig. 1. The complexity of data dynamics: (a) Block Position and (b) Hook Load

Scatter plots for each time series data have been produced in order to check on data linearity. From the ten available measurement time series data only two showed linearity trends. These include Bit Depth and Hole Depth measurements. The test for data *Normality* was performed using Mardia's goodness-of-fit test for multivariate normality [3]. The results have shown that the drilling measurement time series data are non-Gaussian.

2.2 Observation data analysis (OS labels)

OS data are generated by drilling engineers as real-time observations, using expert knowledge assessments and *Morning Reports*. The latter are filled up when phases of drilling operations are complete and passed to the next operating drilling teams. The OS labels are consequently noisy and subjective. The statistical analysis of the 9 wells showed that 15% to 25% of OS labels were missing for each well. Also, the generated labeled OS occurred at different durations and frequencies, i.e. they are statistically imbalanced. Table 1 illustrates such issue (Well140).

Data complexity measures [4] were recently proposed to quantify the characteristics of data which affect accuracy of classification such as 1) *Overlaps of* *classes in feature space*; 2) *Separability of classes;* and 3) *Class density in overlap region*.

CircHL	CleanDN	CleanUP	DrlRot	DrlSld	MakeCN	MoveDN	MoveUP	WashDN	WashUP
9.2%	5.8%	1.3%	33.8%	21.3%	16.7%	3.5%	3.8%	2.4%	2.2%

Table 1: Distribution of OS label, Well140

Fisher's discriminant ratio (F1), the ratio of average intra/inter class nearest-neighbour distance (N2), and class density in overlapped regions (D3) represent a useful set of indicators for the good classification characteristics of the dataset. The generalization for L classes which also considers all feature dimensions was suggested in [5] for F1 and can be calculated as follows:

$$F1 = \frac{\sum_{i=1}^{L} n_i \cdot \delta(\boldsymbol{\mu}, \boldsymbol{\mu}_i)}{\sum_{i=1}^{L} \sum_{j=1}^{n_i} \delta\left(\mathbf{x}_j^i, \boldsymbol{\mu}_i\right)}, \tag{1}$$

where n_i is the number of samples in class i, δ is a similarity metric, $\boldsymbol{\mu}$ is the overall mean, $\boldsymbol{\mu}_i$ is the mean of class i, and $\mathbf{x}_j^i$ corresponds to the sample j of class i. When $F1 = 0$, a complete overlap exists between classes, while $F1 > L - 1$ means that there is no overlap. The intermediate values of $F1$ show the level of overlap between some classes. In this study, $F1 = 1.97$ for training data set of 10 classes. This shows that although there is no complete overlap between all classes, some classes may still overlap.

N2 measures class separability in the following way:

$$N2 = \frac{\sum_{i=1}^{N} \text{intra}(\mathbf{x}_i)}{\sum_{i=1}^{N} \text{inter}(\mathbf{x}_i)}, \tag{2}$$

N is the number of data samples, $\text{intra}(\mathbf{x}_i)$ is the distance to the nearest neighbour within a class for a sample i; and $\text{inter}(\mathbf{x}_i)$ is the distance to the nearest neighbour of any other class. Low values of N2 suggest that samples of the same class are well separated from other classes, whereas large values of N2 indicate that they are dispersed. Table 2 shows N2 values, calculated for each OS (class) for drilling data. The DrlRot class is the best separated class from the rest of classes. Hence one expects that this class can be easily classified. CleanDN and MakeCN classes exhibit good separability, while the MoveDN has the worst separability measure, followed by the DrlSld and MoveUP classes. These last three states mentioned may consequently present some confusion during the classification process.

The aim of the class density D3, as introduced in [5], is to determine the relative density of each class within an overlapping region. The lower the values of D3, the less number of samples lie within the overlapping region. Table 2 shows D3 values of samples in overlapping regions for all OS. The OS with the smallest D3s include DrlRot, MakeCN and CircHL. However, CleanUP, DrlSld and WashUp have shown D3s exceeding 60%. The rest of the OSs has shown significant high proportion of the overlapping regions. This analyses shows the type of challenges exists in this classification problem with overlapping classes and high class densities overall. As a result, the selected classification algorithms need to overcome the multi-class imbalance and complexity of the drilling data.

OS	N2	D3 (%)
CircHL	1.6	9.5
CleanDN	0.7	34.3
CleanUP	1.5	66.6
DrlRot	0.1	4.1
DrlSld	4.7	62
MakeCN	0.2	6.1
MoveDN	8.3	48
MoveUP	4.2	28.9
WashDN	1.5	49.0
WashUP	1.3	66.0

Table 2: Measures of data complexity N2 and D3

3 Selection of classifiers for the best performance

Following the above data analysis, reference to one of the most comprehensive review in [6] and the authors' experience with complex data classification, eight machine learning algorithms were selected. These include: *1) k-Nearest-Neighbour (kNN), 2) Support Vector Machines (SVM), 3) Linear Discriminant Analysis (LDA) 4) Echo State Network (ESN), 5) Random Forest (RF), 6) AdaBoostM2, 7) RusBoost and 8) Subspace*. Each of the algorithms were evaluated using micro-averaged and macro-averaged F-measures [2]; together with the Matthews Correlation Coefficient (MCC) [7] for their respective overall performances. Correct Classification Rates (CCR) were adopted for the assessment of individual OSs. The larger the F-measure is the higher the classification rate. Micro-average F-measure gives equal weights to each label and tends to be dominated by the classifier's performance on common classes. Macro-average F-measure gives equal weights to each class regardless of its frequency. It is influenced more by the classifier's performance on rare classes. Both measurement scores are used to analyze how well classifiers perform under common and rare classes. MCC summarizes the confusion matrix into a single value and is regarded as a good measure for problems with unbalanced classes. It returns a value between -1 and 1, where 1 is a perfect prediction, 0 no better than a random prediction and -1 indicates a total disagreement between prediction and observation. The selected eight classifiers were trained using sensor measurements with given OS at **Well140**. Their testing was subsequently performed on two other Wells of the same Rig (**Well80** and **Well85**). Only **Well80** is presented in this instance.

Ten sensor measurement were considered: *1) Block Position; 2) Bit Depth; 3) Hole Depth; 4) Weight on Bit; 5) Mud Flow; 6) Pump Pressure; 7) Rate of Penetration; 8) Rotary Torque; 9) Hook Load and;10) Rotary Speed*. Six additional features were also considered: *1) Hole Depth - Bit Depth; 2) Hole Depth + Block Position; 3) Bit Depth + Block Position; 4) Rotary Torque * Rotary Speed; 5) Pump Pressure * Mud Flow and; 6) Rate of Penetration * Weight on Bit.* Three experimental scenarios were designed according to various utilizations of the amount of labeled data from **Well140** for classifiers training: 1) 100% of data are considered for training (All); 2) 30% of data are considered for training using uniform sampling without replacement (30_UWR) and; 3) 30% of data are considered for training using a hybrid sampling (30_HS). These scenarios were considered to assess the possibility of reducing training sets without losing in the classification accuracy on the testing sets. The investigation on the sensitivity of the classifiers to various sub-sampled sets and the comparison of confidence intervals under scenarios 2) and 3) were performed using ten different Monte Carlo samples which were respectively drawn for each sampling scheme. Each algorithm was fine-tuned in order to achieve best performance. Table 3 shows the algorithms performance for **Well80**. Three algorithms such as RF, AdaBoostM2 and RUSBoost show best performance for the different training sets. They achieved similar performance according to all three assessment criteria: F_1 and F_2 measures reached values above 80% and 55% respectively; while MCC was above 0.7 for all these algorithms. The kNN and SubSpace algorithms consistently performed poorly Table 3 shows that the volume of the training datasets can be reduced by a third without significantly reducing the classifiers performances.

Method	ALL (%F_1,%F_2,MCC)	30_UWR (%F_1,%F_2,MCC)	30_HS (%F_1,%F_2,MCC)
kNN	(56,35.2,0.42)	(66.3±3.2,36.8±1.1, 0.53±0.05)	(50±2.6,33.6±0.7, 0.37±0.02)
SVM	(71.2,47.2,0.58)	(73.7±0.6,46.9±0.4, 0.62±0.008)	(70.6±2.4,45.8±1.0, 0.58±0.03)
LDA	(70.4,37.2,0.58)	(71.2±0.5,38.4±1.2, 0.59±0.006)	(69.9±0.2,37±0.4, 0.56±0.002)
ESN	(67.4,34.5,0.56)	(58.8±7,33±1.7, 0.48±0.06)	(66.2±2.3,36.3±2, 0.53±0.03)
RF	(84.1,57.1,0.77)	(85.2±0.3,57.4±0.5, 0.79±0.006)	(83.5±0.5,57.5±1.6, 0.74±0.01)
AdaBoostM2	(85.2,61.4,0.77)	(85.3±0.6,60.2±1.9, 0.77±0.01)	(85±0.4,60.5±0.6, 0.76±0.005)
RUSBoost	(85.3,61.6,0.76)	(84.2±0.5,59.5±1.1, 0.75±0.005)	(83.1±1.7,57.9±2.3, 0.73±0.03)
SubSpace	(65.1,22.3,0.61)	(64.6±1.4,21.1±2.7, 0.59±0.02)	(56.4±0.7,20.8±1.1, 0.44±0.01)

Table 3: Comparison of algorithms, Well 80

Uniform sampling Without Replacements (30_UWR) led to good overall performances for **Well80**. Though HS produced more balanced classes for classification, it did not preserve data structure. DrlRot, DrlSld and MakeCN operational states should not be misclassified, since they are critical for decision-making during normal operations. However, the accurate classification of WashUP/WashDN, CleanUP/CleanDN or MoveUP/MoveDN could become more important, when critical situations. As shown in Figure 2 below, three classifiers fulfill best results. These are **RF**, **AdaBoostM2** and **RUSBoost**. These nominated classifiers achieved high CCRs which are greater than 90% in the majority of cases of the critically important states under normal conditions such as DrlRot, DrlSld and MakeCN.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

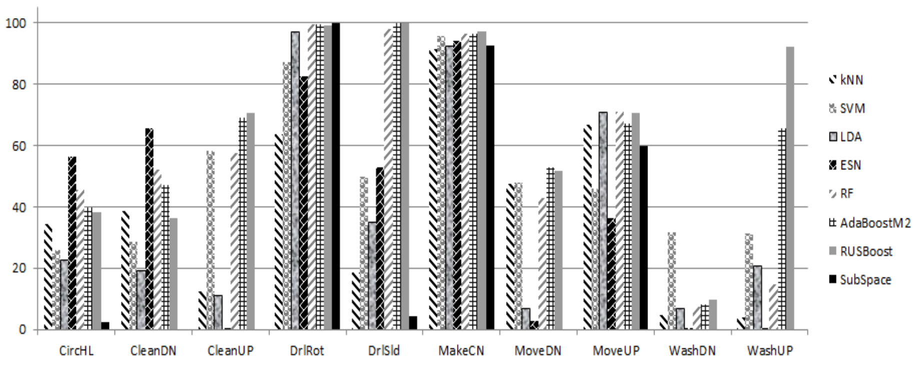

Fig. 2: Classification results for algorithms selection

4 Conclusions

A thorough benchmarking study has been achieved for the selection of the most performing classifiers for the detection of operational states in drilling operations. Strategies were put in place to filter out the less performing classifiers and maintain those which efficiently coped with complex drilling operation data and multiple states classification. Prior knowledge on the geophysical strata of the operating rig can potentially assist on further inferences for improving the identified performing classifiers. RF, AdaBoostM2 and RUSBoost were found highly reliable for achieving real-time automated detection of operational drilling states. They are proposed as the best classifiers for building the next generation decision-support information systems for achieving safer drilling operations in industrial rigs.

References

[1] B. Esmael, R. Fruhwirth, A. Arnaout, and G. Thonhauser. "Operations Recognition at Drill-Rigs," *EGU General Assembly 2012,* Vienna, Austria, April, 2012.

[2] G.V.Veres and Z. A. Sabeur. "Automated operational states detection for drilling system control in critical conditions". In Proceedings of 21st European Symposium on *Artificial* Neural *Networks, Computational Intelligence and Machine Learning (ESANN2013)*, Burges, Belgium, 24-26 April, 2013. i6doc.com publ., ISBN 978-2-87419-081-0.

[3] K. Mardia. "Measures of multivariate skewness and kurtosis with applications". *Biometrika*, vol. 57, pp. 519-530, 1970

[4] T.K. Ho, M. Basu, "Complexity measures of supervised Classification Problems". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 289-300, 2002

[5] J.S. Sanchez, R.A. Mollineda, J.M. Sotoca, "An analysis of how training data complexity affects the nearest neighbor classifiers". *Pattern Analysis and Applications*, vol. 10, no. 3, pp. 189–201, 2007

[6] X. Wu , V. Kumar , J. R. Quinlan , J. Ghosh , Q. Yang , H. Motoda , G. J. McLachlan , A. F. M. Ng , B. Liu , P. S. Yu , Z.-H. Zhou , M. Steinbach , D. J. Hand and D. Steinberg "Top 10 Algorithms in Data Mining", *Knowledge and Information Systems*, vol. 14, no. 1, pp.1 -37 2008

[7] J. Gorodkin. "Comparing two K-category assignments by a K-category correlation coefficient"**.** *Comp Biol and Chem*, vol. 28, pp. 367-374, 2004.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

Prediction of Concrete Carbonation Depth using Decision Trees

Woubishet Zewdu Taffese, Esko Sistonen and Jari Puttonen

Aalto University, Department of Civil and Structural Engineering
P. O. Box 12100, FI-00076 Aalto, Finland

Abstract. In this work, three carbonation depth predicting models using decision tree approach are developed. Carbonation, in urban areas is often a reason for reinforcement steel corrosion that causes premature degradation, loss of serviceability and safety of reinforced concrete structures. The adopted decision trees are regression tree, bagged ensemble and reduced bagged ensemble regression tree. The evaluation of the predictions performance of the developed models reveals that all the three models perform reasonably well. Among them, reduced bagged ensemble regression tree showed the highest prediction and generalization capability.

1 Introduction

Corrosion of reinforcement steel in concrete induced by carbonation is the foremost cause of premature degradation, loss of serviceability and safety of reinforced concrete structures [1, 2]. Carbonation of concrete is a natural physicochemical process caused by the penetration of carbon dioxide from the surrounding environment into the concrete through pores in the matrix where the carbon dioxide reacts with hydrated cement. Calcium hydroxide ($Ca(OH)_2$) in contact with carbon dioxide (CO_2) forms calcium carbonate ($CaCO_3$). This chemical reaction reduces the alkalinity of the pore fluid from pH value around 13 to pH value of below 9. Consequently, the passive oxide layer steel reinforcement is destroyed and eventually corrosion of the steel bars will be initiated [2, 3].

Concrete carbonation depth at a given time in steady state conditions can reasonably be estimated using Eq. (1) for usual life-time of concrete structures. This equation is based on Fick's second law of diffusion and it is well known [2].

$$x = C\sqrt{t} \qquad (1)$$

where, x is the depth of carbonation at time t [mm] , C is coefficient of carbonation [mm/$d^{0.5}$], and t is the duration of carbonation [d].

Coefficient of carbonation is a decisive factor in determining carbonation depth. It is analyzed either by an accelerated carbonation test or by measuring the development of the carbonation depth from an existing concrete structure. Since carbonation is a slow process, it is usually investigated by performing accelerated test with a higher CO_2 concentration in a controlled environment at the age of 28 days [4]. Then, the measured carbonation depth is used to calculate the equivalent carbonation coefficient using Eq. (1). Carbonation coefficient is mainly controlled by diffusion of CO_2 into the concrete pore system. CO_2 diffusion through concrete depends on several factors such as CO_2 concentration, environmental condition, and concrete characteristics. Therefore, carbonation coefficient may significantly vary from one concrete structure to another depending on environment and microstructural parameters which are linked with concrete composition and type of materials used.

Developing analytical carbonation depth prediction model is a challenging task since it is a function of many parameters that are complex to describe mathematically. Hence, building a model that can learn from readily available real data using a machine learning algorithms is a better alternative. Even though this approach is becoming a common practice in various engineering fields, its application in concrete durability is yet limited. Among several machine learning techniques, only artificial neural network is widely used in this research area, for instance, chloride penetration in concrete [5] and hygrothermal forecasting in thick-walled concrete [6].

This paper presents a machine learning method, namely a decision tree, for prediction of concrete carbonation depth.

2 Data understanding and preparation

2.1 Data understanding

Experimental data obtained from [7] is used to develop a model for predicting the depth of carbonation. This data were prepared for Finnish DuraInt-project. The project was carried out in cooperation between Aalto University and VTT Technical Research Centre of Finland. The data consists of concrete mixture ingredients and fresh and hardened properties of 46 specimens. Carbonation depths for half of the concrete specimen were conducted at the age of 28 days and the remaining half at the age of 56 days. The accelerated carbonation tests were performed by applying CO_2 of 1% in a controlled environment (temperature 21°C and relative humidity 60%) in accordance with EN 13295. The data contain both numerical and categorical inputs. In this work, only data of the concrete mixture ingredients and the carbonation depth is used, which is in total 15 features. These are: cement type, water to binding ratio (w/b), cement, blast-furnace slag (BFS), fly ash (FA), total effective water, total aggregate, aggregate < 0.125mm, aggregate < 0.25mm, aggregate < 4mm, product name of plasticizer, plasticizer, product name of air-entraining agent, air-entraining agent, carbonation period and carbonation depth.

2.2 Data preparation

An input matrix of [46x15] predictor values from concrete mixture parameters was arranged. Each column of an input matrix represents one variable, and each row represents one observation. A numeric column vector, carbonation depth, with the same number of rows as input matrix was prepared and assigned as a target. Each entry in output vector is the response to the data in the corresponding row of the input matrix. Since the environmental conditions for all test specimens were identical, this parameter is not included in the predictor matrix. The dataset were used for both training and testing datasets with 10-fold cross-validation.

3 Modeling carbonation depth using decision trees

Decision tree is a nonparametric hierarchical data structure which implements the divide-and-conquer strategy. It is composed of internal decision nodes and terminal leaves as illustrated in Figure 1. The left panel plots the data points and partitions and the right panel shows the corresponding decision tree structure. Each decision node implements a test function with discrete outcomes labeling the branches. Given an input, at each node, a test is applied and one of the branches will be chosen depending on the outcome. This process starts at the root and is repeated recursively until a leaf node is hit, at which point the value written in the leaf constitutes the output [8].

In this work, three different decision trees are used to predict concrete carbonation depth. These are regression tree, ensemble bagged regression tree and bagged regression tree after features reduced. All the trees were developed using Matlab.

3.1 Regression tree

The structure of the regression tree is the same as that of the tree presented in Figure 1. The only difference is the leaves which contain real numbers instead of class labels. The regression tree is trained over the training dataset. The performance of the developed tree is measured by mean square error (MSE) and mean absolute error (MAE) on both training and testing dataset. MSE, the mean square error between predicted output ($\hat{Y}_i$) and target (Y_i), is the most common measure of accuracy, Eq. (2). The MAE of Eq. (3) is the more intuitive measure and is less sensitive to outliers.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \hat{Y}_i\right)^2 \qquad (2)$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|Y_i - \hat{Y}_i\right| \qquad (3)$$

where $\hat{Y}_i$ is the predicted output value, Y_i is the measured target value, and N is the number of observations.

The resulting MSE values for training and test dataset were 0.0416 and 4.3108, respectively. Significant difference in MAE of training and testing dataset is also observed. All these show that the developed regression tree generalized the test data poorly because it overfitted the training data as seen in the regression plot, Figure 2.

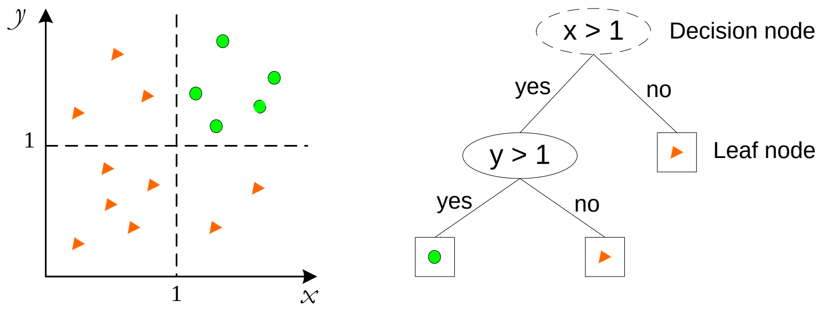

Fig. 1: Example of a dataset and the corresponding decision tree.

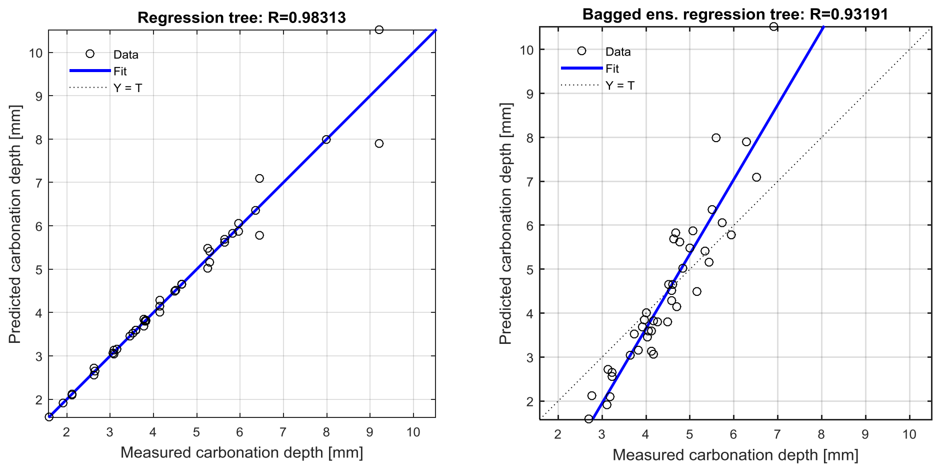

Fig. 2: Regression plot of predicted vs measured carbonation depth on training dataset for regression tree (left) and bagged ensemble regression tree (right).

3.2 Bagged ensemble regression tree

Bagging is one of the most effective methods that can be used to improve the predictive performance of a tree model by reducing the variance associated with prediction. This technique draws multiple bootstrap samples from the training dataset and generates multiple predictor trees, and then, the results are combined by averaging to obtain the overall prediction [9, 10].

An ensemble of bagged regression tree was developed with an initial default tree and leaf size. The performance evaluation indicates that ensemble of bagged regression tree has a high generalization capacity than the regression tree presented in Section 3.1. The MSE of the training and testing dataset was 0.9701 and 2.7223. Regression plot of predicted vs measured carbonation depth on training dataset for bagged ensemble regression tree is shown in Figure 2.

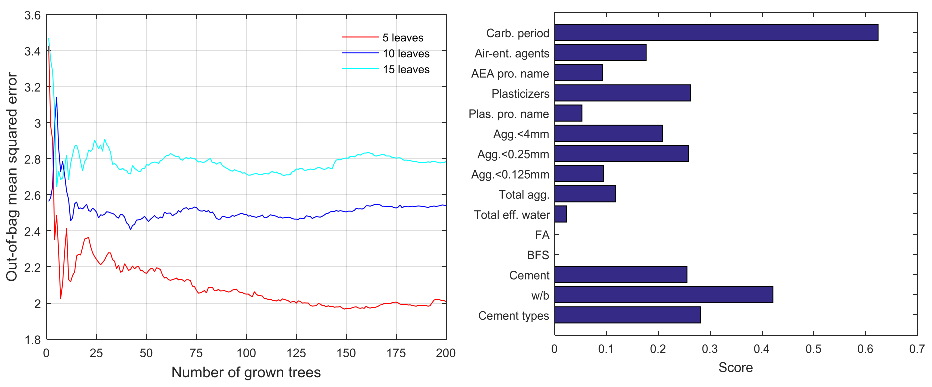

Fig. 3a: Out-of-bag mean square error vs number of grown trees (left). 3b: Relative importance of the input variables of the bagged ensemble regression tree (right).

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

3.3 Reduced bagged ensemble regression tree

In order to minimize the prediction error of the bagged ensemble, we compute predictions for trees with different leaf sizes on its out-of-bag observations, Figure 3a. It can be observed that the out-of-bag error decreases well with the number of grown trees for leaf size of five. The relative importance of the input variables of the bagged ensemble regression tree is illustrated in Figure 3b. It can be clearly seen that the carbonation period and w/b are the foremost influential predictors for this dataset. Next to these variables, amount and types of cement, plasticizer and the distribution of aggregate play considerable role in predicting the carbonation depth for this dataset. This is a useful finding because plasticizer and aggregate distribution were overlooked in several existing analytical models.

After determining good predictors and an ensemble size from the out-of-bag error, a new bagged ensemble regression tree was constructed to enhance its performance further. In this case, the optimal number of leaf and trees was chosen as 5 and 150, respectively. Two parameters, BFS and FA, were reduced out of the total 15 features since they are unimportant to predict the carbonation depth in this dataset. The MSE of training and testing dataset of this model was 0.9536 and 2.2990. Figure 4 illustrate the predicted and the measured carbonation depth with the predicted error.

3.4 Performance comparison

An average of five round statistical performance measurements of all the carbonation prediction models are listed in Table 1. As shown in this table, reduced ensemble bagged regression tree is statistically outperformed all the other models for this dataset. The MAE values of this model for training and test dataset are 0.4755 and 0.5261, respectively. These indicate that this model reasonably fits the measured data and has relatively better generalization capability. All the performance measurements of the models are valid only for the considered specific dataset. If a different dataset is employed, the performance may differ noticeably. Generally, this study revealed the applicability of decision tree based models to predict concrete carbonation depth. As part of future work, the model will be evaluated using more experimental data.

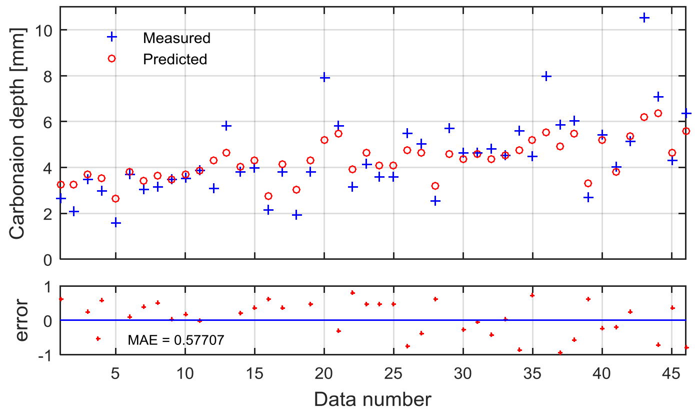

Fig. 4: Measured and predicted carbonation depth using bagged ensemble regression tree with the prediction error.

Models	MSE		MAE	
	Train	Test	Train	Test
Regression tree	0.0416	4.3108	0.0740	1.3437
Bagged ensemble regression tree	0.9701	2.7223	0.4927	0.6283
Reduced bagged ensemble regression tree	0.9536	2.2990	0.4755	0.5261

Table 1: Performance comparison of carbonation depth prediction models.

4 Conclusions

Three concrete carbonation depth prediction models based on decision tree method are presented. To develop the models, three different decision trees were adopted. They are regression tree, bagged ensemble regression tree and reduced bagged ensemble regression tree. The models prediction capacity was examined based on mean square errors and mean absolute error. Models developed using bagged ensemble with and without features extraction predict the carbonation depth with reasonably low error. The model developed using the former method has superior performance with relatively better generalization capability. This confirms the advantage of feature and ensemble size selection in improving performance. Furthermore, the bagged ensemble regression tree identified important variables that influenced the carbonation rate which was not considered in the existing analytical models. The models have potential to be part of a service life management system.

References

[1] K. Y. Ann, S.-W. Pack, J.-P. Hwang, H.-W. Song and S.-H. Kim, Service life prediction of a concrete bridge structure subjected to carbonation, *Construction and Building Materials,* 24:1494–1501, Elsevier, 2010.

[2] fib, Structural concrete: Textbook on behaviour, design and performance, 2nd edition, Volume 1, Technical Report, Fédération internationale du béton (*fib*), Lausanne, Switzerland, 2009.

[3] E. Sistonen, *Service life of hot-dip galvanised reinforcement bars in carbonated and chloride-contaminated concrete*, *Doctoral Thesis, Helsinki University of Technology, Espoo,* Finland, *2009.*

[4] P. Schiessl and S. Lay, Influence of concrete composition, In H. Böhni, editor, *Corrosion in reinforced concrete structures*, pages 91-134, Woodhead Publishing Ltd, Cambridge, 2005.

[5] H.-W. Song and S.-J. Kwon, Evaluation of chloride penetration in high performance concrete using neural network algorithm and micro pore structure, *Cement and Concrete Research*, 39:814–824, Elsevier, 2009.

[6] W. Z. Taffese, F. Al-Neshawy, J. Piironen, E. Sistonen and Jari Puttonen, Monitoring, evaluation and long-term forecasting of hygrothermal performance of thick-walled concrete structure, In *proceedings of OECD/NEA WGIAGE Workshop on the Non-Destructive Evaluation of Thick-walled Concrete Structures,* OECD, Pages 121-143, September 17-19, Prague, (Czech Republic), 2013.

[7] H. Kuosa, Concrete durability field testing: field and laboratory results 2007-2010 in DuraInt-project, Research Report, VTT Technical Research Centre of Finland, Espoo, Finland, September 2011.

[8] J. Gama, *Knowledge discovery from data streams,* Taylor and Francis Group, Boca Raton, 2010.

[9] E. Alpaydin, *Introduction to machine learning*, 2nd edition, The MIT Press, Cambridge, 2010.

[10] C. D. Sutton, Classification and regression trees, bagging, and boosting, In C. R. Rao, E. J. Wegman and J. L. Solka, editors, *Data mining and data visualization*, pages 303-330, Elsevier B.V., Amsterdam, 2005.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

Powered-Two-Wheeler safety critical events recognition using a mixture model with quadratic logistic proportions

Ferhat Attal, Abderrahmane Boubezoul, Allou Samé and Latifa Oukhellou

Université Paris-Est - IFSTTAR

Abstract. This paper presents a statistical methodology that uses both acceleration and angular velocity signals to detect critical safety events for Powered Two Wheelers (PTW). The problem of recognition of critical events has been performed towards two steps: (1) the feature extraction step, where the multidimensional time trajectories of accelerometer/gyroscope data were modeled and segmented by using a specific mixture model with quadratic logistic proportions; (2) the classification step, which consists in using the k-nearest neighbor (k-NN) algorithm in order to assign each trajectory characterized by its extracted features to one of the three classes namely Fall, near Fall and Naturalistic riding. The results show the ability of the proposed methodology to detect critical safety events for Powered Two Wheelers.

1 Introduction

In recent years, road safety has become a priority for the governments of European countries. While statistics show a substantial decline in the number of fatalities on the road for four-wheeled vehicles, in the category of Powered Two-Wheelers (PTWs), the statistics show only a minor reduction with a constant decrease (approximately 2%). In France, PTW traffic represents approximately 2% of the overall traffic flow; however, it represents 40% of all serious injuries and approximately 20% of all deaths.

Driver behavior and driver errors are major causes of vehicular accidents. Therefore, observing and understanding driver behavior has attracted much attention from researchers. Many studies have been made to determine what factors are associated with critical events that occur before crashes. One method among these tools is naturalistic driving/riding studies (NDS/NRS). Thus, several vehicles are equipped with embedded sensors. Each participant drives his/her vehicle for an extended period of time. In this way, NRS data can provide knowledge about rider behavior, i.e., how the rider interacts with her/his vehicle. Additionally, by identifying critical events, useful contextual information can be provided to intelligent transportation systems (ITS) developed for PTWs, thereby improving their effectiveness. An event is defined as an undesirable riding event, such as hard braking, lane changing and sharp turning.

This paper focuses on automatic incident detection based on the same idea as [1]. The authors applied a robust outlier detection methodology based on the Mahalanobis distance to detect critical incidents. As this method is a threshold

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

based method, the main drawback of such approaches is the difficulty of determining the thresholds of regular and irregular riding behavior. In this work our aim is to distinguish between the regular and irregular riding behavior and to detect the switching time between these two profiles. In this paper, the problem of incident detection is performed via two steps: (1) **the segmentation step**, this step consists in segmenting in an unsupervised context the multidimensional time series of accelerometer/gyroscope data. Therefore, each segment is modeled by a regression model and logistic functions are used to model the transitions between segments. (2) **the classification step**, this step consists in classifying segments by using the k-nearest neighbor (k-NN) algorithm, each segment being characterized by its mean and its variance. This approach was applied on real experiments conducted by different subjects driving instrumented PTWs and its performance was assessed by performing comparison with another algorithm, the standard Multiple Hidden Markov Model (MHMM) [2]. To start with, the next section is dedicated to the proposed statistical model for the PTW events detection problem.

2 PTW safety critical events detection model

2.1 PTW experimental data

In this work we have used the data coming from 3D Inertial Measurement Unit (accelerometers/ gyroscopes) mounted on the PTW. In order to evaluate the reliability and the robustness of the proposed methodology, three kinds of scenarios were conducted: **(i) Fall scenarios (F)**: Five falling scenarios were selected and replayed by a stuntman. A set of 8 trajectories were performed by the stuntman including five scenarios known: *Standstill fall*, *Fall in a curve*, *Fall on a slippery straight road section*, *Fall with leaning of the motorcycle "intentional maneuver"* and *Fall in a roundabout*. For more details on this experimentation the reader is invited to see [3]. **(ii) Near fall scenarios (Nf)**: A set of 10 trajectories were performed during extreme cases of riding behavior including aggressive riding such harsh braking, accelerating and swerving. These extreme manoeuvres were carried out on track by a professional rider. The purpose of performing these manoeuvres is to study the robustness of the proposed algorithm in such limit handling behavior. **(iii) Naturalistic riding scenarios (N)**: This experiment was performed in urban area near to the city of Paris, under different weather conditions (sunny, rainy and foggy). 11 trajectories were rode by five riders with different profiles and riding experiences. The participants were given an instruction to drive like they usually do. For more details on this experimentation the reader is invited to see [4].

The signals have been collected with a sampling frequency of 1 Khz. A filtering task was carried out by using the Wavelet Filter (WF) with a six level of wavelet decomposition and Daubechies mother wavelet (Db20) [4].

The collected discrete observations constitute a sample of multivariate trajectories:

$$\boldsymbol{\mathcal{D}} = \{\boldsymbol{x}_i\}_{i=1:N}, \tag{1}$$

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

where N is the total number of trajectories. Each trajectory $\boldsymbol{x}_i = \{\boldsymbol{x}_{it}\}_{t \in \mathcal{T}_i}$ is supposed to be observed at the time vector $\mathcal{T}_i = \{t_{i1}, \ldots, t_{iT_i}\}$, where T_i represents the trajectory length. In this study, only the data recorded from the 3D-accelerometer/gyroscope were used. Therefore, $\boldsymbol{x}_{it}$ is defined as:

$$\boldsymbol{x}_{it} = \{a_x, a_y, a_z, r_x, r_y, r_z\} \in \mathbb{R}^6,$$

where, a_x, a_y and a_z are the longitudinal, the lateral and the vertical accelerations respectively. And r_x ,r_y and r_z are the roll, the pitch and the yaw angular velocities respectively.

2.2 A Gaussian mixture model with quadratic logistic proportions

This section aims at describing the model that will be used for accelerometer/gyroscope signals segmentation. For this purpose a specific mixture model with time varying proportions is formulated, which is a specific case of the regression model with hidden logistic process (RHLP) [5].

2.2.1 Model definition

The signal segmentation model used in this work assumes that each observation $\boldsymbol{x}_{it}$ $(t = 1, \ldots, T_i)$ of the trajectory $\boldsymbol{x}_i$ $(i = 1, \ldots, N)$ is distributed according to the following mixture of two Gaussian distributions:

$$p(\boldsymbol{x}_{it}; \boldsymbol{\theta}) = \pi(t; \boldsymbol{w}) \mathcal{N}(\boldsymbol{x}_{it}; \boldsymbol{\beta}_1, \boldsymbol{\Sigma}_1) + (1 - \pi(t; \boldsymbol{w})) \mathcal{N}(\boldsymbol{x}_{it}; \boldsymbol{\beta}_2, \boldsymbol{\Sigma}_2) \qquad (2)$$

where $\mathcal{N}(.; \boldsymbol{\beta}, \boldsymbol{\Sigma})$ is the Gaussian probability density function with mean $\boldsymbol{\beta}$ and covariance matrix $\boldsymbol{\Sigma}$. The parameters $(\boldsymbol{\beta}_\ell, \boldsymbol{\Sigma}_\ell)_{\ell=1,2}$ are the means and covariance matrices of the Gaussian components, and $\pi(t; \boldsymbol{w})$ is the first component proportion defined by:

$$\pi(t; \boldsymbol{w}) = \frac{\exp(w_0 + w_1 t + w_2 t^2)}{1 + \exp(w_0 + w_1 t + w_2 t^2)}, \qquad (3)$$

whose parameter is $\boldsymbol{w} = (w_0, w_1, w_2) \in \mathbb{R}^3$. These quadratic mixture proportions allow a specific segmentation of the trajectories. They are particularly suitable for segmentation problems with reswitching transitions, especially encountered in the case of riding behaviour.

The parameter $\boldsymbol{\theta} = \{\boldsymbol{w}, \boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2\}$ of this model is estimated by maximizing the log-likelihood through the Expectation-Maximization algorithm, as detailed in [5]. After this step, a set of features is calculated based on the parameter $\boldsymbol{\theta}$ estimated for each trajectory. The set of features $\mathcal{F}$ is calculated as follow:

$$\mathcal{F} = \{\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \|\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2\|, diag(\boldsymbol{\Sigma}_1), diag(\boldsymbol{\Sigma}_2), \|diag(\boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_2)\|\}$$

where $\|.\|$ is the norm associated the Euclidean distance, and $diag(\boldsymbol{\Sigma})$ is the vector of the diagonal elements of $\boldsymbol{\Sigma}$. It should be noticed that the segments were rearranged according to temporal order which corresponds to the real riding activity. This set of features is used as learning database for classifying segments by using the k-nearest neighbor (k-NN) algorithm. By varying the value k between 1 and 10, we selected the value $k = 3$. The aim of this step is to judge if the transition between two contiguous segments is a consequence of an event occurring, which can bee seen as an important change between the mean and the variance from one segment to another.

3 Results and discussion

As mentioned above, the PTW events detection methodology is evaluated on a real database. This database is constituted by 29 trajectories in which we have 8 falling trajectories representing 5 scenarios, 10 near-falling trajectories representing 6 scenarios and 11 naturalistic riding trajectories.

In this section we present the results of the segmentation of different trajectories using the proposed approach GMMQLP (Gaussian Mixture Model with Quadratic Logistic Proportions) and MHMM. As we can see on the example presented on Figure 1, the proposed approach performs well the data segmentation compared to the MHMM. In the particular case of fall scenario, it can be clearly observed that the data is correctly segmented by the GMMQLP algorithm. We can also notice that the data is still correctly segmented by the MHMM algorithm with some "pseudo" transitions, Figure 1(g). In the case of near fall scenario, the same observation can be made for the two algorithms, with more "pseudo" transitions in the case of the MHMM algorithm. In the case of naturalistic riding scenario, the GMMQLP algorithm does not segment the data, which is not the case of the MHMM algorithm where many "pseudo" transitions can be observed, this result can be explained by the flexibility of the logistic process that govern the switching from one segment to another.

The Table 1 represents the obtained results in terms of correct segmentation rate for each scenario. These results are obtained by matching the segmentation results to the true labels (given by an expert). We can notice that the GMMQLP performs better segmentation than the MHMM. We have to recall that the aim

	Correct segmentation rate			
	F trajectories	Nf trajectories	N trajectories	Global correct segmentation rate
GMMQLP(%)	98.77	95.37	95.72	96.62
MHMM(%)	92.50	85.81	76.93	85.08

Table 1: Correct segmentation rate obtained with GMMQLP and MHMM algorithms.

of this study is to develop an automatic incident detection approach. This approach aims is to distinguish between the regular and irregular riding behavior and to classify the segment into three classes (fall, near fall or naturalistic) riding

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

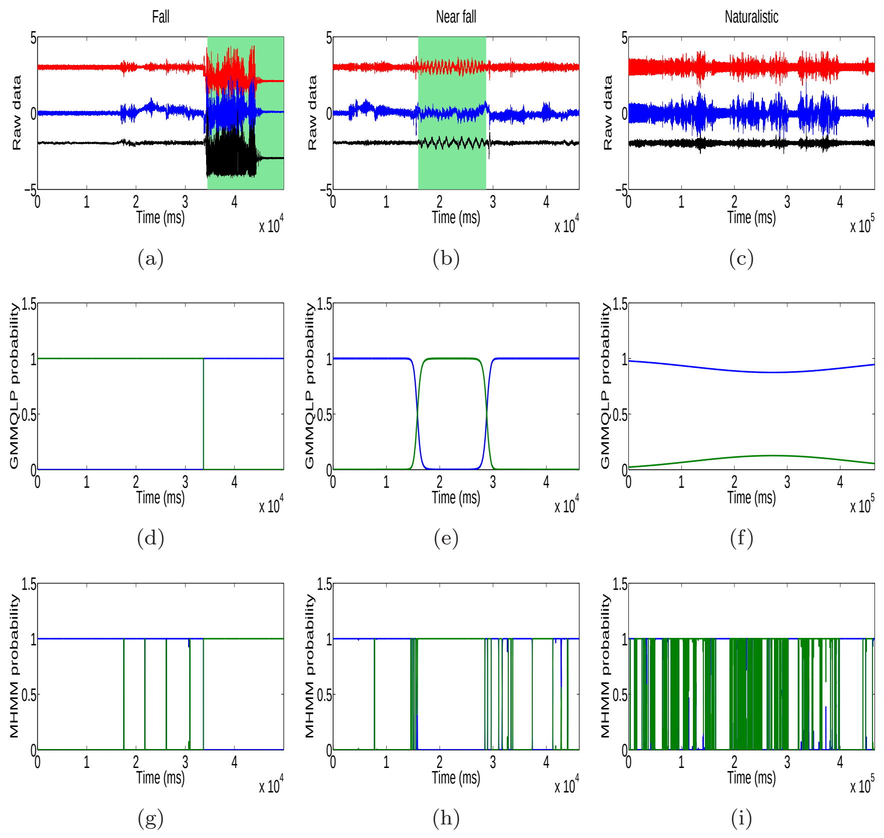

Figure 1: Results obtained by applying the proposed GMMQLP model (middle) and the MHMM approach (bottom) on the acceleration time series measured during fall, near fall and naturalistic riding cases (from left to right). In a, b and c the red, blue and black signal represent a_z, a_x and a_y, respectively. In d, e, f, g, h the estimated probabilities obtained with the two approaches are presented.

		Predicted classes		
		F	Nf	N
	F (%)	100	0	0
Real	Nf (%)	0	90	10
classes	N (%)	0	0	100

Table 2: Global confusion matrix for the GMMQLP algorithm.

		Predicted classes		
		F	Nf	N
	F (%)	100	0	0
Real	Nf (%)	0	70	30
classes	N (%)	0	0	100

Table 3: Global confusion matrix for the MHMM algorithm.

events. This classification task is performed by the k-nearest neighbor (k-NN) algorithm on the database $\mathcal{F}$, each segment being characterized by its mean and its variance, as stated before. The global confusion matrices for the GMMQLP and the MHMM algorithms are given in Table 2 and Table 3. AS expected we can observe that the confusions occur especially for near fall scenarios, less for GMMQLP algorithm than for MHMM algorithm. These confusions can be explained by the fact that in theses situations the features extracted from GMMQLP algorithm $\mathcal{F}$ are more discriminative than the features extracted from the MHMM algorithm.

4 Conclusion and Further Work

In this paper we presented a new method for PTW events detection by using a specific mixture model with quadratic logistic functions. Among data collected from an instrumented motorcycle we have only used the 3D Inertial Measurement Unit (accelerometers/ gyroscopes) as input data. The obtained results show the effectiveness of the proposed algorithm to solve such problem. The increasing computational capabilities of on-board computers makes the proposed methodology suitable for realtime events detection problem, this idea will be investigated as a future work.

References

[1] Eleni I Vlahogianni, George Yannis, and John C Golias. Critical power two wheeler driving patterns at the emergence of an incident. *Accident analysis and prevention*, 2013.

[2] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. volume 77 (2), pages 267–296, 1989.

[3] Abderrahmane Boubezoul, Bruno Larnaudie, Stéphane Espié, and Samir Bouaziz. A simple fall detection algorithm for powered two wheelers. *Control Engineering Practice*, 21(3):286 – 297, 2013.

[4] Attal Ferhat, Boubezoul Abderrahmane, Oukhellou Latifa, and Espie Stephane. Riding patterns recognition for powered two-wheelers users' behaviors analysis. In *Intelligent Transportation Systems (ITSC), 2013 16th International IEEE Conference*. IEEE, 2013.

[5] Faicel Chamroukhi, Allou Samé, Gérard Govaert, and Patrice Aknin. Time series modeling by a regression approach based on a latent process. *Neural Networks*, 22(5):593–602, 2009.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

Real-time activity recognition via deep learning of motion features

Kishore Konda[1], Pramod Chandrashekhariah[2], Roland Memisevic[3] and Jochen Triesch[1,2] *

1- Goethe University Frankfurt, Germany

2- Frankfurt Institute for Advanced Studies, Germany

3- University of Montreal, Canada

Abstract.
Activity recognition is a challenging computer vision problem with countless applications. Here we present a real time activity recognition system using deep learning of local motion feature representations. Our approach learns to directly extract energy based motion features from video blocks. We implement the system on a distributed computing architecture and evaluate its performance on the iCub humanoid robot. We demonstrate real time performance using GPUs, paving the way for wide deployment of activity recognition systems in real world scenarios.

1 Introduction

Activity recognition is of primary interest in various applications such as video surveillance, medical care, human-computer interaction etc. [1, 2]. In the recent past, there has been an increasing interest in activity analysis from areas such as elderly care and health monitoring of patients. Traditionally, patients are required to wear a variety of sensors to identify their daily live activities [3, 4]. Vision based recognition schemes come in handy in such applications as they allow the use of passive, non-contact sensors. This, however, requires a system that not only learns and recognizes the activities in new scenarios but also provides real-time performance. In this work we develop a end-to-end learning based activity recognition system that learns from a minimal data set for a given scenario providing high speed and real-time recognition performance. We integrate and demonstrate the system on a robotic platform.

Using local motion features for activity recognition is a popular approach employed in many of the previous works [5, 6, 7, 8]. Approaches like [8] use traditional handcrafted features like HOG3D, HOF etc., as local motion features whereas so-called energy models [7, 6, 5] learn motion features from the input data. In traditional energy models, motion, or the spatial transformation between two frames of a sequence, is represented as the sum of squared quadrature Fourier or Gabor coefficients across multiple frequencies and orientations [5]. Summing over squared quadrature pairs also induces invariance to content, allowing the model to represent pure motion. In [6] it has been shown that

*This work was supported in part by the German Federal Ministry of Education and Research (BMBF) in projects 01GQ084(0/1) (BFNT Frankfurt), by an NSERC Discovery grant and by a Google faculty research award. JT was supported by the Quandt foundation.

learning the spatial transformations and invariance can be viewed as two independent aspects of learning. Based on that view they introduced a single layered autoencoder based model named *synchrony* autoencoder(SAE) for learning motion representations. In this work we use the SAE for learning motion features exploiting its training efficiency. In the next section we briefly explain the SAE model followed by details on the real time activity recognition system in later sections.

2 Learning motion features

In [6] it is shown that the detection of a spatial transformation can be viewed as the detection of synchrony between the image sequence and a sequence of features undergoing the transformation. This is done in the SAE model by allowing for multiplicative (gating) interactions between filter responses applied to the frames in a video. The following is a brief description of the SAE model for sequences.

Let $\vec{X} \in \mathbb{R}^N$ be the concatenation of T vectorized frames $\vec{x}_t \in \mathbb{R}^M, t = 1, \ldots, T$. Let $\mathbf{W}^x \in \mathbb{R}^{Q \times N}$ denote a matrix containing Q feature vectors $\vec{W}^x_q \in \mathbb{R}^N$ stacked row-wise. Each feature is composed of individual frame features $\vec{w}^x_{qt} \in \mathbb{R}^M$ each of which spans one frame $\vec{x}_t$ from the input sequence. The filter responses, or "factors", are defined as $\vec{F}^X = \mathbf{W}^x \vec{X}$. A simple representation of motion can then be defined as

$$H_q = \sigma((F^x_q)^2), \tag{1}$$

Learning in an autoencoder is generally achieved by minimizing the sum of a reconstruction cost and a regularization term, using gradient descent. In this work we use contractive regularization [9]. The cost function for the SAE model together with the regularization is given by

$$\mathcal{J}_C = \|(\vec{X} - \hat{\vec{X}})\|^2 + \lambda \|J_e(\vec{X})\|^2_E, \tag{2}$$

where $\|J_e(X)\|^2_E$ denotes the Frobenius norm of the Jacobian of the hidden units with respect to the inputs [9], which for σ defined as *logistic sigmoid* is given by

$$\|J_e(\vec{X})|^2_E = \sum_j (H_j(1 - H_j))^2 (F^x_j)^2 \sum_i (W^x_{ij})^2. \tag{3}$$

The hyper-parameter λ is set via a grid search. The SAE model is used as the feature extraction module of the activity recognition pipeline explained in the following section.

3 Activity pipeline

Our activity analysis pipeline is based on the bag-of-words approach used in [5, 6]. The pipeline consists of a feature extraction module followed by K-means

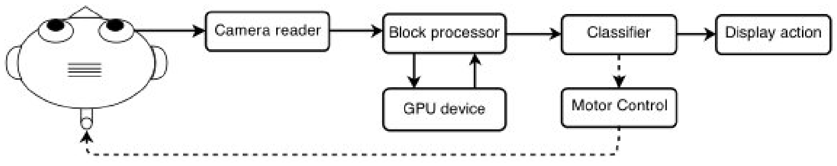

Fig. 1: Block diagram of real time activity recognition system on iCub.

vector quantization and finally a χ^2 kernel SVM for classification. The model described in Section 2 is used for motion feature extraction. The model is trained on PCA-whitened input patches of size $10 \times 16 \times 16$ ($time \times space \times space$). The total number of training samples is $200,000$. The size of the latent hidden layer representation from the model is fixed at 300.

It has been observed that spatially combining local features learned from smaller input patches leads to better representation than features learned on larger patches [5, 10]. In this regard, for computing a local feature describing a larger region of input video, sub blocks of the same size as the patch size are cropped from "super blocks" of size $14\times20\times20$ [5, 6]. The sub blocks are cropped with a stride of 4 on each axis giving 8 sub blocks per super block. The feature responses of sub blocks are concatenated and dimensionally reduced using PCA to form the local feature. On these local features a K-means layer with 3000 centers is learned with $500,000$ samples for training. This gives a dictionary of 3000 words where each word can be thought of as a motion pattern. The pipeline can be viewed as layer-wise trained deep neural network.

For inference the super-blocks are extracted densely from the input video with a 50% overlap. The resulting local features, from convolution operation, from the entire video are quantized using the learned K-means vocabulary resulting in a 3000 dimensional histogram or bag-of-words feature vector. The histograms from the training videos are used for training the SVM classifier. During inference the histogram computed for a given test sequence is classified into a activity label.

4 Real-time recognition system

In this section we describe the real-time implementation of the activity recognition system. The overall architecture of the system is as shown in Figure 1. platform. The individual modules shown in Figure 1 are run on different machines that are connected through an open-source platform called YARP (a robotic platform). We use a C++ implementation that uses OpenCV (computer vision library) and GPUs (Graphical Processing Units) for fast computations to cater to the real-time performance. We explain in detail the individual modules in the rest of this section.

Camera reader: This module handles the video stream from the cameras of the iCub humanoid robot [11]. The *camera reader* receives a continuous stream of frames from the cameras on the robot, bundles them into groups of 14 and then sends them to the *block processor* module. There is an overlap of 7 frames netween subsequent video blocks.

Block processor: This module implements the activity recognition pipeline explained in Section 3 which computes a histogram or motion descriptor (output of the K-means quantization step) given a video block. The input to the *block processor* are video blocks of size 14 frames from the *camera reader*. The computed histogram for each input video block is passed on to the *classifier* module. Our implementation has the ability to parallelize block processing by running multiple instances of *block processor* on multiple GPUs.

Classifier: The module handles the output histograms from the *block processor* by maintaining a first-in-first-out buffer of size 10 in our case. For every new incoming histogram the module updates the buffer and predicts an action label by summing over histograms in the buffer i.e., over past 10 blocks (4 secs of video) unlike the offline case that uses the entire sequence to predict. The label information can be further used in detection and tracking of the user involved by controlling the gaze of the humanoid robot (see dotted lines in Fig. 1). We plan to do this as part of our future work.

4.1 Implementation details

Hardware configuration and speed General-purpose computing on GPUs has gained popularity in recent years especially in the field of computer vision for speeding up the algorithms that involve intense computations on images. We use GPUs for the *block processor* module. Our current implementation ran on a system with 2.6 GHz CPU, 12 GB RAM and GTX 480 NVIDIA GPU devices. We observed that the time taken for processing one video block of 14 frames varied from 350 to 450 milliseconds that corresponds to an overall processing speed of 15 to 20 frames per second *fps*, considering overlap of 7 frames between the video blocks. When using four GPUs the speed further increased to 42 *fps* which higher than what is considered is as real time performance.

YARP: a robotic architecture: We develop the system on a distributed architecture to share the load onto different machines. We modularize the algorithm into different parts that are simultaneously run and coordinated through YARP. YARP (Yet Another Robot Platform) is a robot software architecture that can run a collection of programs on different machines and lets them communicate in a peer-to-peer way [12]. In our work we run the *camera reader*, *block processor*, *classifier* and *Display* modules on different cores/machines.

iCub robot: iCub is an open-system robotic platform that is generally considered an interesting experimental platform for analyzing cognitive, visual and sensorimotor behaviors. iCub is designed with physical dimensions resembling a 3 year old child. The head in particular has two dragon fly cameras with VGA resolution that are mounted as eyes used for video capture in our system. The eyes can produce images at resolution of 320×240 at a rate of ~ 20 *fps* .

5 Dataset and results

As mentioned in the previous section we recorded a dataset on iCub robotic platform for parameter training and testing of our system. The dataset includes

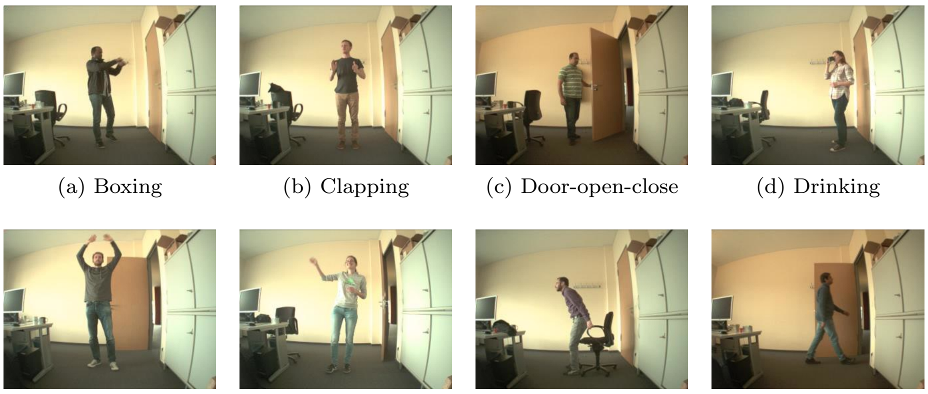
(a) Boxing (b) Clapping (c) Door-open-close (d) Drinking
(e) Waving-double (f) Waving-single (g) Sit-Stand (h) Walking

Fig. 2: Example snapshots of different actions from the collected dataset.

	Boxing	clapping	DOC	drinking	HWD	HWS	sitstand	walking
Boxing	46.6/49.8	26.6/9.6	0./0.4	0./7.2	0./1.2	20./15.4	0./10.6	6.6/5.5
clapping	11.1/1.6	61.1/52.7	0./0.	5.5/20.8	0./1.11	5.5/10.8	16.6/8.3	0./4.4
DOC	0./0.	0./0.	100./91.4	0./8.5	0./0.	0./0.	0./0.	0./0.
drinking	0./0.	0./0.	0./2.7	100./95.8	0./0.	0./0.9	0./0.	0./0.5
HWD	0./0.	11.1/6.5	0./0.	0./0.3	50./75.6	22.2/17.5	16.6/0.	0./0.
HWS	0./0.	0./0.	0./0.	0./11.7	0./0.	80./88.2	20./0.	0./0.
sitstand	0./0.	0./0.	0./0.	11.9/47.8	0./0.	0./0.	88.1/52.1	0./0.
walking	0./0.	0./0.	0./0.4	0./6.2	0./0.	0./0.	0./0.	100./93.3

Table 1: Confusion matrix of classification experiments. Offline test/Realtime test.

videos from 8 different people $P0$ to $P7$ performing 8 different actions. The actions are namely "clapping", "door open close"(DOC), "drinking", "hand waving double"(HWD), "hand waving single"(HWS), "sit stand", "walking" and "boxing". These actions are chosen as they are a subset of most observed human behaviors in an indoor workplace environment. Example snapshots of different actions are shown in Figure 2.

A total of 574 videos are collected and are divided into a training set of 355 clips from persons $P1, P2, P3, P5, P7$ and a testing set of 219 clips from $P0, P4, P6$. Each person performed an action multiple times with two different types of clothing (jacket on and jacket off). Since it is very likely that a person performs an action similarly in multiple tries, the dataset is split into training and testing set based on person rather than choosing random subsets of the total set. The classification accuracy of the activity pipeline (Section 3) on the testing set is 85.39%. The confusion matrix of the classification experiment is shown in Table 1.

In order to validate the real time performance of the system, apart from demonstrating it live on the iCub, we also run the system on longer test videos collected from users $P0, P4, P6$ to quantify the results. The *classifier* module of the system is set to predict for every new incoming video block which implies a

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

new prediction at an interval of seven frames (due to overlap). The classification performance of the real time system is 74.91 and the corresponding confusion matrix is reported in Table 1. The deviation from the offline performance on the testing set is mainly due to inability of the current system to deal with user activities other than the ones system is trained on.

6 Conclusion

The real-time system presented in this work achieves high processing speeds and competitive performance by utilizing a very limited set of data samples. The learned local motion features used by the system are well generalized there by no additional parameter training (except the classifier) is necessary to be able to recognize new sets of actions. The parallel architecture and utilization of GPUs give the system the ability to achieve higher processing speeds when required. In future we plan to utilize action class information for better detection and tracking of the user via gaze control of the humanoid robot.

References

[1] Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

[2] Pavan Turaga, Rama Chellappa, Venkatramana S Subrahmanian, and Octavian Udrea. Machine recognition of human activities: A survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(11):1473–1488, 2008.

[3] Tam Huynh, Ulf Blanke, and Bernt Schiele. Scalable recognition of daily activities with wearable sensors. In *Location-and context-awareness*, pages 50–67. Springer, 2007.

[4] Seon-Woo Lee and Kenji Mase. Activity and location recognition using wearable sensors. *IEEE pervasive computing*, 1(3):24–32, 2002.

[5] Q.V. Le, W.Y. Zou, S.Y. Yeung, and A.Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011.

[6] Kishore Reddy Konda, Roland Memisevic, and Vincent Michalski. Learning to encode motion using spatio-temporal synchrony. In *Proceedings of ICLR*, April 2014.

[7] Graham W. Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *Proceedings of the 11th European conference on Computer vision: Part VI*, ECCV'10, 2010.

[8] Heng Wang, Muhammad Muneeb Ullah, Alexander Kläser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *University of Central Florida, U.S.A*, 2009.

[9] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *ICML*, 2011.

[10] Adam Coates, Honglak Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Artificial Intelligence and Statistics*, 2011.

[11] Giorgio Metta, Lorenzo Natale, Francesco Nori, Giulio Sandini, David Vernon, Luciano Fadiga, Claes Von Hofsten, Kerstin Rosander, Manuel Lopes, José Santos-Victor, et al. The icub humanoid robot: An open-systems platform for research in cognitive development. *Neural Networks*, 23(8):1125–1134, 2010.

[12] Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. Yarp: yet another robot platform. *International Journal on Advanced Robotics Systems*, 3(1):43–48, 2006.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

Designing Semantic Feature Spaces for Brain-Reading

L. Pipanmaekaporn[1]*, L. Tajtelbom[2], V. Guigue[2] and T. Artières[3] †

1- King Mongkut's University of Technology North Bangkok, Thailand

2- Laboratoire d'Informatique de Paris 6 (LIP6) Paris, France

3- Laboratoire d'Informatique Fondamentale (LIF) Marseille, France.

Abstract.
We focus on a brain-reading task which consists in discovering a word a person is thinking of based on an fMRI image of their brain. Previous studies have demonstrated the feasibility of this brain-reading task through the design of what has been called a semantic space, i.e. a continuous low dimensional space reflecting the similarity between words. So far the best results have been achieved by carefully designing this semantic space by hand which limits the generalization of such a method. We propose to automatically design several semantic spaces from linguistic resources and to combine them in a principled way and achieve results comparable to that of manually built semantic spaces.

1 Introduction

Neuroimaging has gained much interest in the last decade in many fields ranging from philosophy and psychology to neuroscience and artificial intelligence. Among brain imaging techniques, functional Magnetic Resonance Imaging (fMRI) has become a primary tool to detect mental activity with great spatial resolution [1]: an fMRI image contains approximately 20,000 voxels (volumantic pixels) that are activated when a human performs a particular cognitive function (e.g., reading, mental imagery) [2]. With fMRI, it became possible to associate brain areas with cognitive states: specific conceptual words and pictures trigger specific activity in some parts of the brain and studies began to focus on the extraction of meaningful brain activation patterns [3, 4].

A pioneering work [5] showed that it was possible to predict the brain activation pattern (a fMRI image) in response to a given conceptual stimulus (e.g. a word). Reciprocally [6] demonstrated on the same dataset the feasibility of identifying the concept from the brain activation pattern (fMRI image). The proposed approaches for these two reciprocal tasks share the definition of an intermediate semantic (or representation) space to represent the concepts, the underlying idea being that it allows the problem of inferring the concept from the fMRI (and vice versa) to come down to a standard regression problem from the fMRI voxel space to the semantic space (and vice versa). Importantly if

*This work was done while the author was at LIP6 for an internship.

†We want to thank the French government for having funded the stay of L. Pipanmaekaporn in France within the Franco-Thai Junior Research Fellowship program from May to July 2014.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

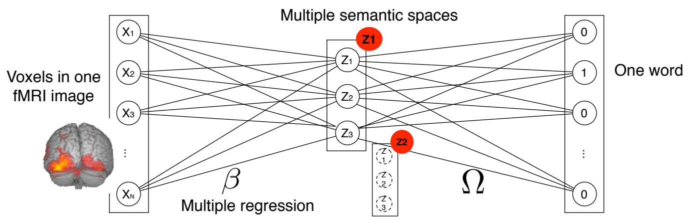

Fig. 1: Brain-reading processing chain

the representation space is designed in such a way that one can get the representation of any new word, such a strategy naturally allows the recognition of concepts from an fMRI image even if there was no training fMRI image for this word. This may be done in two steps: first, starting from an input fMRI image, a point in the semantic space is computed using the regression model; then, the word whose representation is the closest to this point is found: this is the zero-shot learning setting defined in [6].

Previous studies defined this semantic space "by hand". [6] manually designed a 218 dimensional representation space in which a concept representation is defined according to the answers to 218 questions such as 'is it manmade?' or 'can you hold it?'. Such a semantic space was designed for a particular set of concepts. Later on, to extend these methods to deal with a larger number of concepts, researchers tried to leverage information from lexical and corpus resources to automatically design a universal and accurate semantic space. For instance, [6] built a 5000 dimensional semantic space from the Google n-gram corpus, [7] found that co-occurrences counts with very high frequency words were an informative representation of words for semantic tasks, [8] examined various semantic feature representations of concrete nouns derived from 50 million English-language webpages, etc.

This work deals with the problem of automatically designing a semantic space for [6]'s task, i.e. predicting the concept from the fMRI image in the zero-shot learning setting. Since previous studies have shown the superiority of manually designed semantic spaces we propose to combine multiple and diverse semantic spaces, either automatically learned from huge corpora, following recent works in the machine learning and representation learning community [9], or designed from various linguistic resources (e.g. *WordNet* [10]). In order to exploit these semantic spaces efficiently, we propose to use an effective blockwise regularized learning algorithm [11] that prevents overfitting and focuses on relevant information contained in the fMRI images.

2 Learning Models for Brain Decoding

Our idea consists in combining multiple semantic spaces, some of them being designed automatically using linguistic resources while others are learned using representation learning ideas such as the one in [9]. Our system for inferring a concept from an fMRI image is illustrated in figure 2. It relies on two multilinear mapping functions: Ω maps a single word w in a continuous p-dimensional space so that $\Omega(w) = \mathbf{z} \in \mathbb{R}^p$. We refer to this space as a semantic space. Ω is built using external resources [9, 12] and the representation spaces considered are detailed in section 3. β enables us to make the link between the fMRI image vector $\mathbf{x} \in \mathbb{R}^d$ (an image made of d voxels) and the word semantic representation $\mathbf{z} \in \mathbb{R}^p$. We first consider a simple strategy by learning independently multiple ridge regressions: this will be our baseline when considering multiple semantic spaces in our experiments. We then investigate a more advanced multitask strategy using the multitask blockwise regularized LASSO from [11]: by regularizing jointly all regression models, we can take into account globally the relevance of every voxel with respect to the task. This strategy is explained here. Note that one independent model is learned for each subject.

Let $X = \{\mathbf{x}_i\}_{i=1,\dots,N}$, $\mathbf{x}_i \in \mathbb{R}^d$ be the collection of fMRI images for a subject and $Z = \{\mathbf{z}_i\}_{i=1,\dots,N}$, $\mathbf{z}_i \in \mathbb{R}^p$ the collection of associated word semantic representations. The Ridge Regression (RR) consists in learning $\beta \in \mathbb{R}^{d\times p}$ coefficients that map efficiently from the voxel space to the semantic space. For the Multitask LASSO (MTL), the global blockwise regularized problem is formulated as:

$$\underset{\beta}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i=1}^{N} \|\mathbf{z}_i - \mathbf{x}_i\beta\|^2 + \lambda \sum_{j=1}^{p} \|\beta_j\|_\infty \right) \text{ with } \|\beta_j\|_\infty = \max_{\ell} |\beta_{\ell j}| \tag{1}$$

During training, entire rows of the resulting β matrix will "vanish" so as to focus only on relevant voxels.

After training K models $\beta^{(k)}$, corresponding to K different semantic spaces, we still have to build a decision criterion to choose the word to be associated to the fMRI. Each word w is mapped in the kth semantic space using the $\Omega^{(k)}$ function, thus we get $\Omega^{(k)}(w) \in \mathbb{R}^p$. In parallel, we obtain K semantic representations associated to the fMRI images $\mathbf{x}$ using $\beta^{(k)}$ coefficients. Their cosine similarity can then be computed in the intermediate space and the results are merged using a linear combination, as follows:

$$sim(\mathbf{x}, w) = \sum_{k=1}^{K} \lambda_k \frac{\langle \mathbf{x}\beta^{(k)}, \Omega^{(k)}(w)\rangle}{\|\mathbf{x}\beta^{(k)}\| \; \|\Omega^{(k)}(w)\|} \text{ s.t. } \sum_k \lambda_k = 1 \tag{2}$$

Obviously, the word with the highest similarity to an fMRI image is chosen.

3 Experiments and Discussion

3.1 fMRI Dataset and Task

The fMRI data was collected from nine participants while subjected to a pair of stimuli: a line-drawing depicting a particular concept alongside a text label [5][1] .

[1] [5]'s data is publicly available at http://www.cs.cmu.edu/afs/cs/project/theo-73/www/science2008/data.html. [6, 8, 11] also test their models on that same dataset

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

There were 60 concepts (classes) belonging to 12 semantic categories (mammals, body parts, buildings, furniture, etc.), each presented 5 times to the participants. Every fMRI image is comprised of about $20,000$ voxels representing the cortex activity. Following [5] we considered in our experiments subsets of 500 to 10000 voxels using the same selection procedure they did, based on a stability criterion.

We investigated the zero-shot learning setting defined in [6]. Models' evaluation was done in a leave-2-out cross-validation setting: the training data consisted in 58 classes, and the models learned were tested on the remaining 2 (thus, the classes of the test set are completely unknown to the models).

3.2 Word Semantic Features

We now describe the three considered approaches to design a semantic space.

WordNet based semantic space (WN) WordNet provides easy ways for designing a semantic space. This lexicon is organized as a hierarchical tree of concepts and subconcepts. As a consequence, it is possible to compute a path in the tree between two concepts (words). Intuitively the smaller this path, the closer these two concepts are [10]. Based on such a metric, a given word can be represented in a fixed p-dimensional space by computing its distance to a given set of p representative words: we considered the most common words in Wikipedia. We will call such a semantic space WN_{path}. Alternative metrics have been proposed in the literature that lead to other semantic spaces: the closeness of two concepts can be measured in respect to their closest common ancestor [12] (let us note the corresponding semantic space WN_{anc}) and [13] defines a criterion inspired from mutual information, comparing the weights of subtrees associated to each concept (this semantic space will be denoted WN_{mi}).

Word2Vec semantic space (W2V) Representation learning has emerged in the recent years as a key research field in the machine learning community. Word2Vec is an efficient tool that learns continuous and dense representations of words from text data [9]. It is a supervised learning approach based on neural networks in which a hidden layer is used to encode a vector representation that captures syntactic and semantic patterns of words.

Human218 semantic space (H_{218}) The last semantic space we considered is a baseline noted H_{218}. As explained before it is a manually designed space which has been obtained from crowdsourcing [6]. For each concept considered a 218 dimensional representation is defined according to the answers from a set of volunteers to 218 questions like *is it manmade?* or *can you hold it?*.

3.3 Results and Discussion

In a preliminary experiment, the semantic space dimension p was optimized using a large set of voxels (we fixed $d = 2000$). Results are reported in Fig 2 : a dimension $p = 150$ seems to offer a good trade-off between complexity and

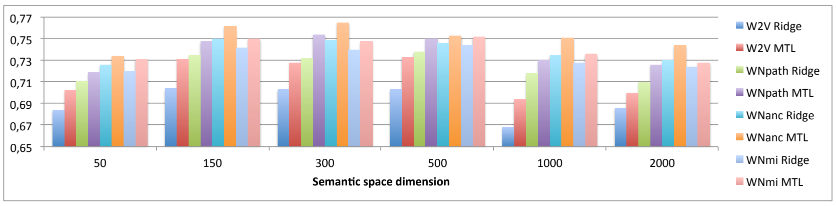

Fig. 2: Accuracy (zero-shot learning) wrt the semantic space dimension, for various semantic spaces (W2V, WN, ...) and for the two training strategies (MTL = Multitask LASSO/ Ridge = Ridge Regression)

accuracy, and this value was kept for further experiments. Also, in every semantic space configuration, we notice that MTL (multitask LASSO) systematically overcomes Ridge regression which validates our regularization strategy for identifying and neglecting unnecessary voxels. Hence we will focus on this model in further experiments.

We then performed a combined experiment to study the impact of voxel preprocessing (reducing the voxel space using the stability criterion proposed in [5]) as well as the interest of mixing different semantic spaces. All results are provided in Fig 3. Best results are obtained for a voxel space size of 2000: we can see MTL procedure can't deal efficiently with large dimensional noisy data such as fMRI images (with their 20000 voxels), and that such a preprocessing to select relevant voxels is recquired.

Our most important result lies in the overall performance on this difficult brain-reading task: up to now, state-of-the-art results relied on Human218 (H_{218}) resources [6], which is hand-made for this task and questions the ability to generalize the process to a larger vocabulary. We demonstrate here the interest of combining different lexical and learned resources to outperform this strategy. While H_{218} reaches an accuracy of 80.3% (last column of Fig. 3), being far above the best single model ($WNanc$) that reaches 76.2%, it is outperformed by our combination schemes. Combining 2 resources provides a significative improvement to catch up with H_{218}: $W2V + WNanc$ model reaches 80.3% accuracy. Adding a third resource ($WNpath$), we reach 80.7% accuracy.

The comparison of various combinations confirms our assumption: it is more relevant to combine heterogeneous spaces like $W2V$ and WN than to work with a single resource.

4 Conclusion

Predicting a concept stimulus from an fMRI image is a hard task which is traditionally tackled through defining a manual semantic space and learning a regression model. While this approach has proved effective for a limited set of concepts, the manual design of the semantic space prevents the approach to be extended to a larger number of concepts. We tackled the problem by relying on multiple

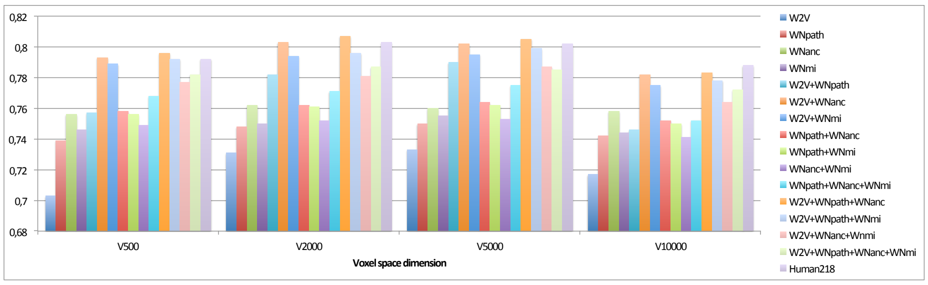

Fig. 3: Accuracy (zero-shot learning) with Multitask LASSO wrt the voxel space dimension and for various semantic space combinations.

semantic spaces automatically designed from resources and trained from large corpora. Given the dimension of fMRI images, it is necessary to implement a robust learning strategy: the multitask LASSO we designed allows us to efficiently select relevant voxels. MTL, combined with Word2Vec and WordNet, catches up with the state-of-the-art performance in brain-reading relying on hand-made resource. It is a promising step towards more advanced brain-reading tasks.

References

[1] R.A. Poldrack. The role of fmri in cognitive neuroscience: where do we stand? *Current opinion in neurobiology*, 18(2):223–227, 2008.

[2] F. Pereira and M. Botvinick. A systematic approach to extracting semantic information from functional mri data. In *NIPS*, pages 2267–2275, 2012.

[3] K.N. Kay, T. Naselaris, R.J. Prenger, and J.L. Gallant. Identifying natural images from human brain activity. *Nature*, 452(7185):352–355, 2008.

[4] D.R. Hardoon, J. Mourao-Miranda, M. Brammer, and J. Shawe-Taylor. Unsupervised analysis of fmri data using kernel canonical correlation. *NeuroImage*, 37(4), 2007.

[5] T.M. Mitchell, S.V. Shinkareva, A. Carlson, K.M. Chang, V.L. Malave, R.A. Mason, and M.A. Just. Predicting human brain activity associated with the meanings of nouns. *science*, 320(5880):1191–1195, 2008.

[6] M. Palatucci, D. Pomerleau, G.E. Hinton, and T.M. Mitchell. Zero-shot learning with semantic output codes. In *NIPS*, 2009.

[7] J.A. Bullinaria and J.P Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3), 2007.

[8] B. Murphy, P. Talukdar, and T. Mitchell. Selecting corpus-semantic models for neurolinguistic decoding. In *ACL JC on Lexical and Computational Semantics*, 2012.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv*, 2013.

[10] C. Leacock and M. Chodorow. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2), 1998.

[11] H. Liu, M. Palatucci, and J. Zhang. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *ICML*, 2009.

[12] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. *IJCAI*, 1995.

[13] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *ACL*. ACL, 1994.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

Learning objects from RGB-D sensors using point cloud-based neural networks

Marcelo Borghetti Soares, Pablo Barros, German I. Parisi, Stefan Wermter *

University of Hamburg, Department of Computer Science,
Vogt-Koelln-Strasse, 30, 22527, Hamburg, Germany

Abstract. In this paper we present a scene understanding approach for assistive robotics based on learning to recognize different objects from RGB-D devices. Using the depth information it is possible to compute descriptors that capture the geometrical relations among the points that constitute an object or extract features from multiple viewpoints. We developed a framework for testing different neural models that receive this depth information as input. Also, we propose a novel approach using three-dimensional RGB-D information as input to Convolutional Neural Networks. We found F1-scores greater than 0.9 for the majority of the objects tested, showing that the adopted approach is effective as well for classification.

1 Introduction

The efficient recognition of parts of the environment is relevant for a variety of scenarios ranging from lower-level grasping and manipulation to higher-level robotic assistance. The way the human brain performs these activities is an attractive source of inspiration, due to its capacity to deal with noise, to operate in cluttered scenarios and to generalize from few examples. Concerning object recognition, some researchers have focused on viewpoint-independent recognition, assuming that this process can be accomplished using invariant features that must match three-dimensional representations of objects stored in the memory [1]. Others state that the brain stores multiple viewpoint representations of objects in the brain and uses strategies to interpolate and to generalize the results for viewpoints that are not initially presented [2].

Additionally, depth information can play an important role in recognition [3] and many approaches for scene understanding and object recognition have been developed in the last years taking into account the rich information provided by depth sensors. The impact of depth and RGB data sources was explored in [4], where the improvement obtained for object recognition tasks performed by a mobile robot was shown. Similar approaches have been tested in which different neural models were employed to treat different representations such as color, shape or depth [5],[6]. This paper differs from these previous approaches since we developed and evaluated different neural models that take into account the geometry of the point cloud. We consider that this three-dimensional information processed by the brain is particularly useful to distinguish parts, faces, etc, of an object under different environment conditions. This is potentially important to recognize categories

*This work is partially supported by CAPES Brazilian Federal Agency for the Support and Evaluation of Graduate Education (Process number 10441-13-1).

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

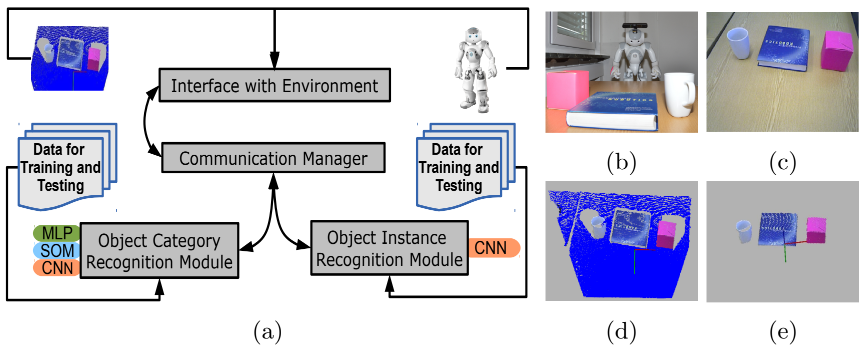
Interface with Environment
Communication Manager
Data for Training and Testing
Data for Training and Testing
MLP
SOM
CNN
Object Category Recognition Module
Object Instance Recognition Module
CNN
(a)
(b)
(c)
(d)
(e)

Fig. 1: (a) Scheme of the proposed framework divided in modules. (b) Scenario with a NAO robot and three objects. (c) RGB image from the RGB-D device. (d) Point cloud from RGB-D device. (e) Segmented objects.

and instances (different books, different cups, etc) of objects. Our contribution can be summarized as follows: i) a novel approach using three-dimensional RGB-D information as input to Convolutional Neural Networks and ii) an implementation that encompasses both invariant geometric features from the objects and features extracted from multiple viewpoints and that allows testing different neural models.

This article is organized as follows: Section 2 presents the framework used and the neural approaches developed, Section 3 is devoted to our experiments and Section 4 concludes and presents future directions.

2 Object recognition and interaction with the environment

A database was built containing RGB-D data from objects in different poses (defined as the objects (x, y, z) coordinates and $(roll, pitch, yaw)$ orientations in relation to the RGB-D device's viewpoint), since the robots should act in an environment and recognize objects regardless of the pose (no capture was made with robots in movement). Figure 1 shows a NAO humanoid robot (b) looking at examples of these objects on a table and the corresponding RGB image (c) and depth capture (d).

We are considering the objects located on the largest surface captured by the RGB-D device. We use RANSAC (RAndom SAmple Consensus) [7] to identify these planes. In our case, the z axis points to the direction of the objects. Therefore, for segmentation, the scene is reoriented in a way that the z axis becomes orthogonal to the normal vector of the plane. Thus, we identify all objects that have y coordinate values higher than the average value of the y coordinate of the points that compose the plane. Finally, we consider only objects located within a *tolerance distance* from the center of mass of the segmented plane (Figure 1d).

2.1 Framework for recognition

Figure 1a shows the system overview. The *Interface with Environment* provides an interface through which it is possible to manually select the objects previously segmented. The features extracted from the selected object are sent to the other modules. The *Communication Manager Module* is responsible for redirecting messages exchanged by different modules. Usually, the *Interface with Environment Module* will send descriptors captured from RGB-D data to recognition modules and these modules will send back the categorization/instantiation.

2.2 Recognition based on three-dimensional feature descriptors

In this case, the input to the neural network models is a feature vector obtained using VFH (Viewpoint Feature Histogram) [8] to collect a multidimensional descriptor representing the geometrical relations of the points that compose an object. We also use *Principal Component Analysis* to reduce the input vector. This descriptor is used for category recognition with two neural approaches: i) Feedforward Network (MLP+VFH) and ii) Self-organizing Map (SOM+VFH). The SOM adopts the labelling scheme of the output presented in [9].

2.3 Recognition based on features from multiple viewpoints

We developed an approach that receives multiple viewpoints as input to a Convolutional Neural Networks (CNN) [10]. Generally, CNNs are composed of multiple layers divided in i) a set of n feature maps (every map is the result of a convolution operation) and ii) a set of n subsampling maps obtained from the feature maps. We developed and tested two different CNN approaches: i) 2DCNN: CNN with two-dimensional kernel and ii) S-3DCNN: CNN with three-dimensional kernel, where a three-dimensional kernel convolves with a stack of images [11]. As typical in applications that use RGB images, this stack is populated with similar images. In our case, for each object we generate n sliced planes that are orthogonal to the z axis (imagine a bread sliced by a knife). As the normal vector to the slice is parallel to the z axis, each slice is represented as a projection in x-y plane. This sliced object fills the stack of images, each slice occupying one position. The stack is used by the S-3DCNN preserving the sequence, as the geometrical relation between the parts of the object matters. Finally, these slices are then convolved with the three-dimensional kernel and the weights of this kernel are adjusted taking into account the position and sequence in the object.

3 Experiments

For our database, it was important to have objects observed from different viewpoints on tables and on the floor to comprise different situations in which the robots could act, since we aimed also testing how the robots generalize the results for different scenarios. We captured 5 different categories with 5 different objects for each category (instances) in 6 different viewpoints. In addition, this database also contains one instance of each category per object (used only in experiments of

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

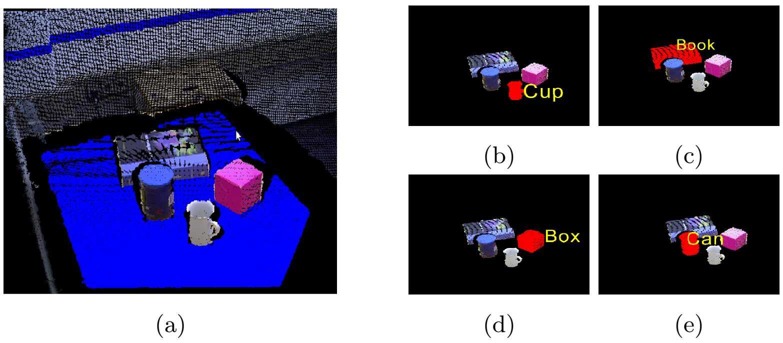

Fig. 2: (a) Online experiment with four objects: (b) Cup, (c) Book, (d) Box and (e) Can.

category recognition): 5 different categories in 8 different viewpoints, 5 times for each viewpoint (to take into account noisy captures). The total number of images acquired was 350. The samples used for training were composed by single objects, but online tests with multiple objects were also conducted.

The number of units for the MLP, CNN and SOM was 100, 250 and 360 respectively. In the CNN, the number of feature maps in the first and second layer was 20 and 30 respectively. We used a kernel of 5×5 pixels. For all neural networks, the learning rate was 0.01. The number of slices in the S-3DCNN was 10. The training and testing sets were divided in 60% and 40%. We performed a systematic search in the parameter space and defined the values that performed better. Each experimental result presented below was obtained from an average over 5 simulations. This number of runs was chosen due to the computational time required and since the methods presented very stable results.

Figures 2(a)-(e) show an example of online recognition with four objects on a table using MLP+VFH. It is important to note that the robot can select one object each time, which is particularly interesting in cluttered environments. Also, the object partially occluded can be recognized.

Figure 3a shows the recognition results based on three-dimensional feature descriptors. We can note that both methods have good accuracy with F1-scores greater than 0.96. To test the performance under different viewpoints, we applied to each sample rotation in roll, pitch and yaw ($3 \times 3 \times 3$). Thus, we have 28 samples per point cloud (27 generated and 1 original). Considering the 350 samples of our database the total amount is 9800 (28×350). For each sample generated, we also added noise in 10% of each of the clouds.

Figure 3b shows the recognition based on features from multiple viewpoints. The results have good accuracy with F1-scores greater than 0.89. In the case of the CNN, we also applied rotation (roll, pitch and yaw) to create a dataset of 9800 samples. The results obtained from Figure 3a and Figure 3b represent different methodologies (and theories) and should be analysed separately. The CNN approaches receive images (size 50×50 pixels) from different viewpoints that provide

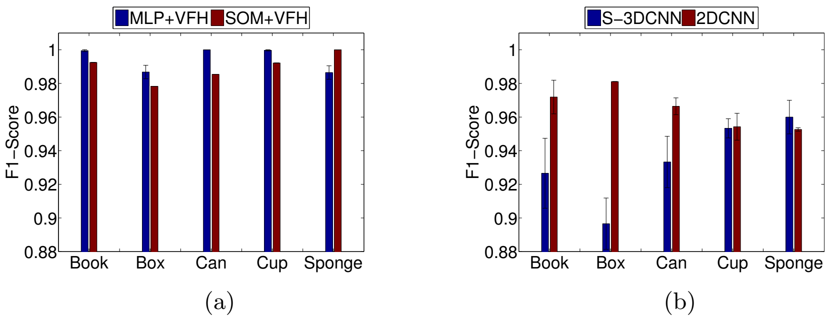

Fig. 3: F1-scores for each category: (a) MLP+VFH and SOM+VFH and (b) 2DCNN and S-3DCNN.

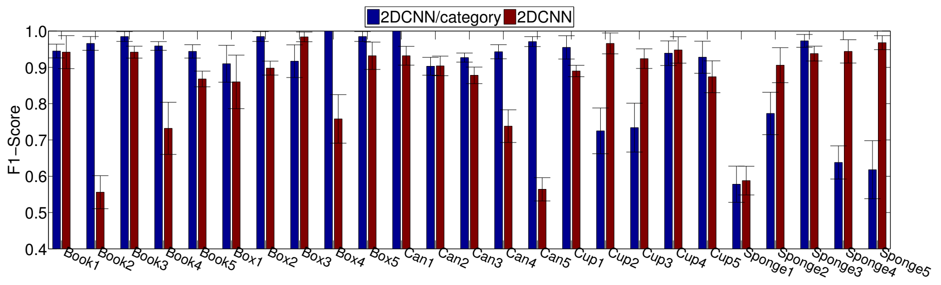

Fig. 4: F1-scores for each instance per category.

information in a slightly different way (projections of the cloud in the x-y plane). We demonstrated with this experiment that the recognition based on features from multiple viewpoints offers an efficient alternative approach. The partial knowledge contained in each viewpoint can be used further for a final decision about the classification of one object. It is possible to use multiple “readings” from different viewpoints to recognize an object when the object is not clearly visible due to noise or distance factors, like animals in general do. Similarly, S−3DCNN has also promising results and requires investigation about how to use the partial knowledge contained in each slice, since they can lead to a different classification.

Instance classification results are shown in Figure 4. We choose 2DCNN since it presented better results in the previous experiments. CNNs can also extract features based on textures, which is ideal for instance classification. We used two approaches: i) one 2DCNN to classify all different instances and ii) five different 2DCNN per category. In general, the results were better or similar for the second case, indicating that the division of labour works. But there are 6 cases for which F1-scores were worse (considering the standard deviation): `Box3`, `Cup2`, `Cup3`, `Sponge2`, `Sponge4`, `Sponge5`. We believe that this behaviour was caused by the fact that some of the instances (for example the sponges) do not have strong geometrical and textural features or that they are too similar to be recognized with a limited number of samples.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

4 Conclusions and Future Works

The framework presented in this paper was developed keeping in mind that robots should understand the interactions between objects and other parts of the environment, such as supporting surfaces. To learn the objects from its RGB-D devices we developed and tested several neural models. The results enabled us to draw the conclusion that features obtained from multiple viewpoints are a rich source of information to be explored. Each view or slice from S-3DCNN potentially stores valuable information that can be used to improve the recognition. As future steps, we plan to integrate multiple views captured over time and evaluate the improvement in the recognition accuracy.

References

[1] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[2] H. H. Bülthoff, S. Y. Edelman, and M. J. Tarr. How are three-dimensional objects represented in the brain? *Cerebral Cortex*, 5(3):247–260, 5 1995.

[3] D. DeAngelis. Roles of visual area MT in depth perception. In Michael S. Gazzaniga, editor, *The Cognitive Neuroscience*, pages 483–498. MIT press, 2009.

[4] L. C. Caron, Y. Song, D. Filliat, and A. Gepperth. Neural network based 2D/3D fusion for robotic object recognition. In *European Symposium on Articificial Neural Networks (ESANN)*, pages 431–436, 2014.

[5] L. A. Alexandre. 3D Object Recognition using Convolutional Neural Networks with Transfer Learning between Input Channels. In *13th International Conference on Intelligent Autonomous Systems*, volume 301 of *Advances in Intelligent Systems and Computing Series*, Padova, Italy, July 2014. Springer.

[6] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part V*, ECCV'12, pages 746–760, Berlin, Heidelberg, 2012. Springer-Verlag.

[7] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[8] R. B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3d Registration. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, ICRA'09, pages 1848–1853, Piscataway, NJ, USA, 2009. IEEE Press.

[9] G. I. Parisi, P. Barros, and S. Wermter. FINGeR: Framework for interactive neural-based gesture recognition. In *22th European Symposium on Artificial Neural Networks, ESANN 2014, Bruges, Belgium, April 23-25*, pages 443–447, 2014.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[11] P. Barros, S. Magg, C. Weber, and S. Wermter. A Multichannel Convolutional Neural Network for Hand Posture Recognition. In *International Conference on Artificial Neural Networks*, pages 403–410, 2014.