Generic Online Animal Activity Recognition on Collar Tags

Jacob W. Kamminga
University of Twente
Enschede, The Netherlands
j.w.kamminga@utwente.nl

Nirvana Meratnia
University of Twente
Enschede, The Netherlands
n.meratnia@utwente.nl

Helena C. Bisby
University of Twente
Enschede, The Netherlands
h.c.bisby@student.utwente.nl

Paul J.M. Havinga
University of Twente
Enschede, The Netherlands
p.j.m.havinga@utwente.nl

Duc V. Le
University of Twente
Enschede, The Netherlands
v.d.le@utwente.nl

UbiComp/ISWC '17 Adjunct, September 11-15, 2017, Maui, HI, USA

ACM ISBN 978-1-4503-5190-4/17/09…$15.00
https://doi.org/10.1145/3123024.3124407

Abstract
Animal behaviour is a commonly-used and sensitive indicator of animal welfare. Moreover, the behaviour of animals can provide rich information about their environment. For online activity recognition on collar tags of animals, fundamental challenges include: limited energy resources, limited CPU and memory availability, and heterogeneity of animals. In this paper, we propose to tackle these challenges with a framework that employs Multitask Learning for embedded platforms. We train the classifiers with shared training data and a shared feature-representation. We show that Multitask Learning has a significant positive effect on the performance of the classifiers. Furthermore, we compare 7 types of classifiers in terms of resource usage and activity recognition performance on real-world movement data from goats and sheep. A Deep Neural Network could obtain an accuracy of 94 $\%$ when tested with the data from both species. Our results show that a Deep Neural Network performs the best among the compared classifiers in terms of complexity versus performance. This work supports the development of a robust generic classifier that can run on a small embedded system with good performance, as well as sustain the lifetime of online activity recognition systems.

Author Keywords
Online Animal Activity Recognition; Multi Species; Machine Learning; Resource Usage; Multitask Learning

ACM Classification Keywords
I.5.m [Pattern Recognition]: Miscellaneous

Introduction
Animal behavior is a strong indicator of welfare and can provide information about social interaction between animals and herds. Additionally, through monitoring animal behaviour, it is possible to detect environmental events such as forest fires [22], poaching activities [19], and environmental problems [18]. Moreover, activity recognition has been implemented to aid the conservation and protection of animals such as rhinoceroses [19, 17]. For many years, collaring technology with Inertial Measurement Units (IMUs) and Global Positioning System (GPS) has been widely used to study animal behavior. Most existing works on animal behaviour recognition are offline and centralized approaches in which sensor data is stored on the tag and retrieved later or is transmitted wirelessly [7]. However, for livestock and wildlife in widespread and remote areas, activity recognition needs to be executed online (while activities are being performed) and locally (on the collar tag). On one hand, collar tags have limited energy supply, memory, processing power, and transmission bandwidth. Local activity recognition will significantly prolong the battery life since it consumes less power for data transmission, which typically consumes more energy than data processing. On the other hand, online activity recognition enables the monitoring system to efficiently adapt its resource usage to a situation (e.g., the device can sleep when the animal is sleeping). In addition, it is dangerous, expensive, stressful for the animal, and sometimes impossible to re-capture the animals for data retrieval and battery replacement.

In this paper, we propose a framework that employs Multitask Learning (MTL) [3] for embedded platforms. We address online activity recognition with collar-based platforms for large heterogeneous groups of animals in real-world environments. Collected behavioural data from a fraction of animals are used for offline training at a central server using MTL, which performs classification tasks across individual animals to learn the significant commonalities. The learned models can then be implemented on the collars for online activity recognition of other animals. Consequently, we can significantly reduce the costs of acquiring labeled animal activity data for numerous heterogeneous wild animals as well as power consumption. Summary results of activity recognition are then sent to a sink node, using a low-cost and long range communication link such as a Low Power Wide Area Network (LPWAN).

In order to further reduce resource consumption while maintaining a high classification performance, we aim at minimizing the complexity of the classification. We extract the most informative features from raw accelerometer data of the collar tags. Since the collars are likely to shift and rotate throughout the day, we also select features that are insensitive to sensor orientation in order to reduce the effects of orientation on classification accuracy. The extracted features are used locally to classify the animal activities using a classification technique that possesses a lightweight inference. As a result, the proposed approach can be employed for various (sub)families of species. Our approach allows for the development of an application in which collars can be placed on a group of animals without the necessity of training the classifier and fine-tuning parameters for every individual. Collars can then be readily deployed on animals with a lower risk of system failure due to mislabelling and configuration.

The main contributions of this paper are:

- This work extends research on human activity recognition and provides a robust technological basis for

Table 1: Observed activities during the day

Activity	Description
Stationary	The animal is lying on the ground or standing still, occasionally moving its head or stepping very slowly.
Foraging	The animal is eating fresh grass, hay from a pile or twigs on the ground.
Walking	The animal is walking. The pace of walking varies from very slowly to nearly trotting.
Trotting	This is the phase between walking and running. The animal is not galloping rapidly but walking very quickly and is therefore in a trot state.
Running	The animal gallops.

online animal activity recognition
- We show that Multitask Learning significantly improves the performance of a generic classifier
- We verify the classification performance of our approach among 7 types of classifiers
- We make our dataset publicly available at [9] for the community

Related Work

In recent years, there has been a considerable rise in interest in activity monitoring of livestock and wildlife using sensors and embedded devices. Recent studies acknowledge the potential of collaring applications and have evaluated offline activity recognition of cows [7, 5] , sheep [13], and vultures [16]. Existing approaches that identify animal behavior rely on data-loggers, the subsequent collection of data, and centralized processing [7, 16]. In real-world applications, these approaches require transferring data to a central location. However, the transmission demands high bandwidth which dramatically reduces the precious battery life of a collar tag due to the high energy consumption of radios. To the best of our knowledge, few studies currently focus on online animal activity recognition on collar tags. Moreover, to the best of our knowledge, there is currently *no* research on generic online activity classifiers for animals. In fact, the only online classifier for animals, to the author's knowledge, was implemented by Petrus in 2016 [19], who therein states that 'the live transmission of on-animal classified behaviour has not been done before', and distinguishes five activities in sheep. Online activity recognition systems with wearables have been widely studied in humans, which has an overlap with animal activity recognition [24].

Multitask Learning (MTL) is an approach to inductive transfer in which multiple learning tasks are solved at the same time [3] and exploits the deep, subtle connections among tasks [4]. To overcome the challenge of heterogeneity in animals we envision a one-fits-all generic classifier; similar challenges have been found in many other research areas such as human activity recognition [8] and computer vision [23].

In this paper, we investigate the applicability of MTL for generic animal activity recognition. Moreover, we do this with a focus on embedded, real-time classification that is robust against heterogeneity in animals. We primarily investigate the impact of sharing a feature-representation and instances between animals on the performance of 7 types of classifiers, ranging from Decision Trees to Deep Neural Networks.

Data Acquisition and Pre-processing

In this section, we first present our data acquisition system, which comprises collar tags containing motion and orientation sensors. We then describe how the sensor data are pre-processed. All experiments involving the animals complied with Dutch ethics law concerning working with animals; thus, an ethics approval was not required.

Data collection

We collected a dataset that comprises multiple sensor data from 4 goats and 2 sheep. The animals differ in size, weight, and age but belong to the same subfamily *Caprinae*. We randomly attached the sensors in various orientations on each individual animal. The various positions of the collars on three animals are indicated in Figures 1 and 2, respectively. The collars were prone to rotation around the animals' necks during the day. We intensively collected data for the duration of 1 day. The tags were synchronized with a precision of <100 ns. We used ProMove-mini [27] tags from Inertia Technology that contain 3D accelerometer and 3D gyroscope sensors. All sensors were sampled at 200 Hz.

Figure 1: Sensor placement on a sheep, the red arrows indicate the location of the sensors

Figure 2: Sensor placement on two goats, the red arrows indicate the location of the sensors

Figure 3: Screenshot of the labelling application

The animals were videotaped from various angles throughout the day. The activities that were observed during the day are listed in Table 1.

Data labeling

We used a labeling application based on a Matlab GUI [14] to annotate our data (see Figure 3). The clock timestamps from the tags were used to obtain a coarse synchronization. The offset between the videos and the sensor data was adjusted in the application to improve the synchronization. The l^2-norm of the accelerometer's 3 axes, expressed in Equation (1), is displayed in the application to visualize the sensor data. A single annotator labeled the data by clicking at the point representing a change in behavior on the graph. The high synchronization achieved with the video and the visualization of the sensor data allowed the annotator to accurately label the activity associated with the sensor data. The stop marker for one activity was also the start marker for the following activity if that activity was of any other type than *unknown*. Transitions between activities were not excluded from the data, thus some labeled data include a transition phase to another activity. All data of all animals was annotated according to the behaviors listed in Table 1. All efforts were put in to ensure high quality of the labeling process. The size of the dataset is shown in Table 2.

Multitask Learned Online Activity Recognition

Multitask Learning (MTL) is an approach to inductive transfer in which multiple learning tasks are solved simultaneously [3] and exploits the deep, subtle connections among tasks [4]. In the context of activity recognition, MTL can be used to train one generalized classifier to predict the behaviour of multiple individuals [26]. We tuned a single classifier with Multitask Learning (MTL). Pan et. al [20] provided the following definitions for domain and task: A domain D is a 2-tuple $(\chi, P(X))$, where χ is the feature space of D, and $P(X)$ is the marginal distribution where $X = x_1, \ldots, x_n \in \chi$. A task T is a 2-tuple $(Y, f())$ for a given domain D, where Y is the label space of D and $f()$ is an objective predictive function for D. $f()$ is not given, but can be learned from the training data [20]. In this case, each species is a domain D and each behavioral activity is a task T. We examined three scenarios, for each of which the set of source domains $SD = D_{s1}, \ldots, D_{sn}$ and target domains $TD = D_{t1}, \ldots, D_{tn}$ comprise either individual animal data, data from one species, or data from both species. In doing so, we investigate the effect of sharing knowledge across species on the generic performance of a generic classifier.

Figure 4 shows a graphical representation of our approach. Each colored box denotes an inner loop in the process. First, we calculated only 3D-vector features from the accelerometer data since they are theoretically robust against sensor orientation [25, 21]. Firstly, we used the Relief algorithm [10] to select the 3 most relevant 3D-vector features for all data (in both Source Domain (SD) and Target Domain (TD)) so that a shared feature-representation was established. Secondly, for each combination k of multiple animals, we mixed the instances into 3 data sets: training, cross-validation, and test data respectively. The training set T_k was always used to train the classifier for combination k. The cross-validation set C_k was used to optimize the parameters of the classifier. Thirdly, all data was standardized by means of a Z-transformation, obtaining a standard score of each feature value. Fourthly, parameter optimization was applied to make a comparison between various classifiers fairer. Finally, the optimally-tuned classifier was used to assess the performance of each combination k of animal data with the test set V_k. Each of the steps is described in more detail in the following subsections. These steps were repeated for each type of classifier.

Table 2: Number of instances per individual for each activity

	Stationary	Grazing	Walking	Running	Trotting
Sheep 1	3071	6039	974	432	409
Sheep 2	6376	5196	1714	386	478
Goat 1	4466	2552	1956	214	252
Goat 2	7346	2044	1850	224	66
Goat 3	7468	3998	1842	30	120
Goat 4	10418	1386	1398	6	69

Table 3: Features that were calculated for each window of data from all sensors and all their axes

Feature	Description
Maximum	Maximum value
Minimum	Minimum value
Mean	Average value
Standard deviation	Measure of dispersion
Median	Median value
25th percentile	The value below which 25 % of the observations are found
75th percentile	The value below which 75 % of the observations are found
Mean low pass filtered signal	Mean value of DC components
Mean rectified high pass filtered signal	Mean value of rectified AC components
Skewness of the signal	The degree of asymmetry of the signal distribution
Kurtosis	The degree of 'peakedness' of the signal distribution
Zero crossing rate	Number of zero crossings per second
Principal frequency	Frequency component that has the greatest magnitude
Spectral energy	The sum of the squared discrete FFT component magnitudes
Frequency entropy	Measure of the distribution of frequency components
Frequency magnitudes	Magnitude of first six components of FFT analysis

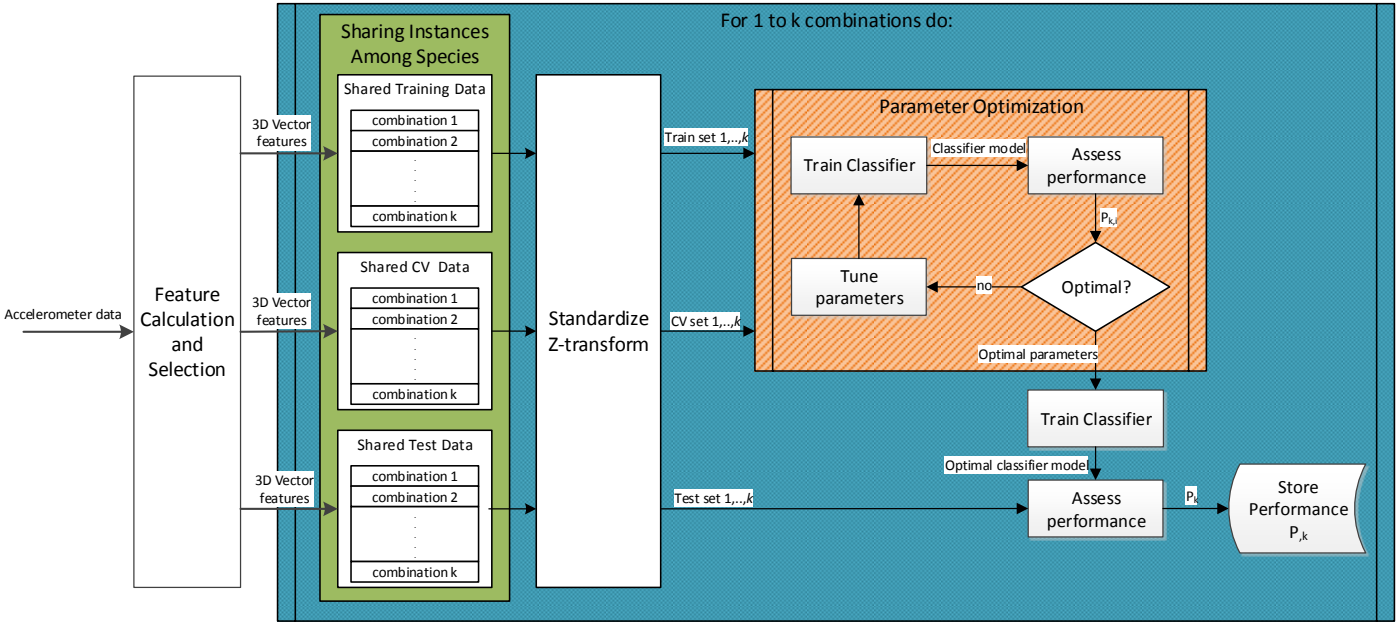

Figure 4: Training and testing with mixed data. Each colored box denotes an inner loop

Feature Calculation and Selection

For each window of data we calculated features that are typically used for activity recognition [1, 13, 24], see Table 3. To acquire orientation independent features, we calculated a 3D vector (the l^2-norm) from the sensors' individual axes. The orientation-independent magnitude of the 3D-vector is defined as:

$$M(t) = \sqrt{s_x(t)^2 + s_y(t)^2 + s_z(t)^2}\,, \qquad (1)$$

where s_x, s_y, and s_z are the three respective axes of the sensor. $M(t)$ was calculated from the accelerometer data.

We calculated the relevance of each feature with the Relief algorithm [10]. The Relief algorithm evaluates instances and compares the value of the current feature with both the closest instances of the same class and of the nearest different classes. Each feature was scored according to its variation and the separation between its own class and nearest class. The weights were normalized so that the top contributing features could be extracted. The *min*, *standard deviation*, and *25th percentile* features were the most relevant and used to characterize the movement data of all animals.

Table 4: Non-mixed training and testing combinations. G_i and S_i denote a goat and sheep, respectively.

Animals in 60% training and 40% CV		Animals in 100% testing			
G1	G2	S2	S1		
G1	G3	S2	S1		
G1	G4	S2	S1		
G2	G3	S2	S1		
G2	G4	S2	S1		
G3	G4	S2	S1		
S1	S2	G1	G2	G3	G4

Table 5: Mixed training and testing combinations. G_i and S_i denote a goat and sheep, respectively.

Animals in 60% training and 40% CV				Animals in 100% testing	
G1	G2	G3	S2	G4	S1
G1	G2	G4	S2	G3	S1
G1	G3	G4	S2	G2	S1
G2	G3	G4	S2	G1	S1
G1	G2	G3	S1	G4	S2
G1	G2	G4	S1	G3	S2
G1	G3	G4	S1	G2	S2
G2	G3	G4	S1	G1	S2

Sharing Instances Among Species

We examined 3 scenarios, for each of which the set of source domains $SD = D_{s1}, \ldots, D_{sn}$ and target domains $TD = D_{t1}, \ldots, D_{tn}$ comprise either individual animal data, data from one species, or data from both species. When only the individual data was used, it was split into $60\,\%$ training, $20\,\%$ cross-validation and $20\,\%$ training. In the second scenario, we shared data among individuals of the same species. Table 4 shows the combinations of animals' data. For each combination k, two animals from one species were divided into $60\,\%$ training and $40\,\%$ cross-validation, while 2 animals' data of the other species were used for testing. In the third scenario we mixed the data among both species. Table 5 shows the combinations of animals' data. For each combination k, 4 animals' data were divided into $60\,\%$ training and $40\,\%$ cross-validation, while 2 animals' data of both species were used for testing.

Parameter Optimization

In order to make a fair comparison between different classifiers, we performed parameter optimization prior to the performance assessment. Parameter optimization finds the optimal values for a set of parameters. We used an evolutionary approach that iteratively adjusted the various parameters of the classifiers until an optimal configuration was found. The optimal configuration of each classifier was then used to assess the performance.

Evaluation

In this section we first describe the classifiers that were implemented together with their most important parameter settings. Then, we discuss the resource usage measurements of the various classifiers. Finally, we evaluate the effect of multitask learning with our real-world dataset.

Classifier Implementations

All classifiers were implemented in RapidMiner [15]. We used the following 7 classifiers in the experiments:

Decision Tree (DT) A decision tree consists of branches and leaves which are navigated depending on feature values [11]. Throughout, the information gain ratio of features was used as the splitting criterion. The maximal depth of the decision tree was varied between 1 and 100. Pruning was enabled, with a confidence level varying between $1 \times 10^{-}7$ and 0.5. Pre-pruning was enabled throughout with varying parameters. Firstly, the threshold feature gain value for splitting was varied between 1 and 100. Secondly, the minimum number of examples at a node in order for the node to be split was varied between 1 and 100. Lastly, the minimum number of examples at a leaf node was varied between 1 and 100.

Neural Network (NN) The neural network used here is a multilayer perceptron (MLP), which is a type of feed-forward neural network which maps input data to output classes using a number of hidden layers which contain neurons [11]. In this case, the activation function used was *sigmoidal*. The number of hidden layers necessary depends on the size of the data set. Since we are dealing with a relatively small data set, only one hidden layer was used. The number of neurons in each hidden layer was defined by:

$$\psi = \frac{\gamma + \rho}{2} + 1\,, \tag{2}$$

where ψ is the amount of neurons in a layer, γ the number of features, and ρ the number of classes. The learning rate was varied between $4.9 \times 10^{-}324$ and 1, while the momentum was varied between 0 and 1. The error epsilon was set to $1 \times 10^{-}5$, and a maximum of cycles used for training was 500.

Support Vector Machine (SVM) The support vector machine is a training algorithm that aims to maximize the margin between data points and the decision boundary. A LibSVM C-SVC [11] model was used with linear kernel throughout. The epsilon value was fixed at 0.001 while the cost function was varied between 0 and 100.

Naive Bayes (NB) Naive Bayes uses Bayes theorem in order to build a model which determines probabilities of each outcome [11]. The traditional naive Bayes algorithm does not use adjustable parameters.

Linear Discriminant Analysis (LDA) LDA [11] aims to discover the combination of features which best distinguish classes. LDA typically has no adjustable parameters.

k-Nearest Neighbors (k-NN) k-Nearest Neighbors (k-NN) determines the class of a feature vector based on the majority vote of the values of the *k*-nearest neighbors in the example set. The measure of distance was Euclidean space, and the value of *k* was varied between 1 and 100.

Deep Neural Network (DNN) The Deep Neural Network (DNN) used is an implementation of the H2O 3.8.2.6 algorithm [2]. The DNN was used with 10 hidden layers which each contained 50 neurons. The start, ramp, and stable momentum values, as well as the number of epochs, were varied between 0 and 100. The annealing and decay learning rates, as well as L1 and L2 parameters, were varied between 0 and 1. The value of epsilon was set at 1.0×10^-8, and the value of rho at 0.99. A multinominal distribution function and a cross entropy loss function were used throughout.

Resource Usage

While most machine learning studies focus on the performance of a technique, many other criteria should be considered when selecting a classifier type [11], especially when selecting a classifier to be used on an embedded platform. The criteria include, but are not limited to: i. accuracy; ii. CPU and memory complexity; iii. sensitivity to irrelevant features; iv. sensitivity to continuous versus discrete features; v. sensitivity to noise; vi. bias and variance of classifiers; vii. storage space required during training and classification stages; viii. possibilities for use as an incremental learner (online ML); ix. ease of use, related to the number of model or run-time parameters to be tuned by the user; and x. the comprehensibility of the classifier.

Here, we focus mainly on the CPU and memory complexity because these components consume the most energy. In order to be able to discuss the trade-off between accuracy and complexity, we measured the CPU run-time and memory usage for all 7 classifiers. Memory and CPU measurements were taken using a PC with an Intel core i7-2600 with 4GB of RAM and a clock speed of 3.40GHz running Windows 7 64-bit. Memory consumption was measured using JProfiler v10.0 [6], and CPU execution times were measured using the log operator in RapidMiner Studio v7.4. The memory consumption was measured by running the same task (training or inferring a model) 5 times, while forcing the garbage collection operation before and after the performed task in order to measure the amount of heap memory consumed by the algorithm. The average of these 5 measurements was taken in order to account for anomalous fluctuations. The training and inference CPU execution times were measured and averaged across 20 runs for each of the algorithms. Using only 3 features made the inference phase too fast on a PC to be able to take high-resolution measurements. In an embedded system there

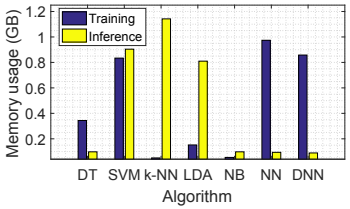

Figure 5: Memory usage for training and inference phases of various classifiers

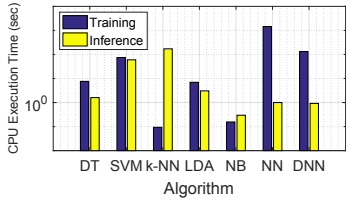

Figure 6: CPU execution time for training and inference phases of various classifiers

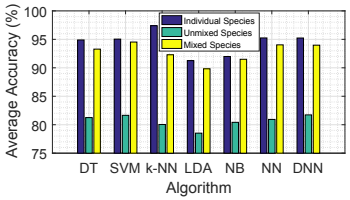

Figure 7: Average accuracies per classifier. The accuracies are averaged over all individuals or combinations

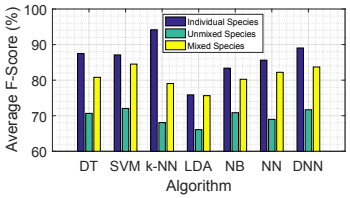

Figure 8: Average F-scores per classifier. The F-scores are averaged over all individuals or combinations

will be a lower CPU speed and less memory than in a PC. To scale the comparison to an embedded system we decided to use higher-dimensional data during the comparison of the classifiers. We used the principal components that account for the top 99 $\%$ of the variability in the data, resulting in 109 components.

Figures 5 and 6 show the memory usage and CPU execution time, respectively. The most important comparison should be made in the inference phase because this is the phase that takes place on the embedded system. It can be seen that the Decision Tree (DT), Naive Bayes (NB), Neural Network (NN), and DNN classifiers consumed an equal amount of memory in the inference phase. The NB classifier was the best performing in terms of CPU usage, closely followed by the NN and DNN classifiers. Therefore, the NB classifier is the cheapest to use in terms of resource usage.

Effect of Multitask Learning

Figures 7 and 8 show the accuracy values and F1 scores of each classifier for the three scenarios described in the previous section. The F1 score is also referred to as F-score or F-measure and can be interpreted as a weighted average of the precision and recall. An F1 score of 1 is optimal and 0 is worst. The figures show the average performances of all individuals or the combinations of mixed and unmixed species, denoted in Tables 4 and 5, respectively.

When the classifier was trained with data from the same individual, this data was within the same domain, thus $SD = TD$. As expected, each classifier performed the best in this scenario. Because there were too few instances for some animals in some classes (see Table 2) the performance of this scenario can be improved by collecting more data for each individual. When the species' data were unmixed, there were data of one species in the training set, and data of the other species in the test set, thus $SD \neq TD$. Figures 7 and 8 both show that all classifiers performed significantly worse in this scenario because the difference between the two species is too large to use only a shared feature representation. When mixing the data from both species in both the training and testing sets, we see a significant improvement in the performance of the classifiers. The performance approaches that of the individual scenario, with the exception of the k-NN classifier's F-score. The big gap in performance between the unmixed and mixed instance scenarios shows that the two species share sufficient characteristics so that a generic classifier can be trained. Our results show that, when taking into account the complexity versus performance trade-off, the DNN classifier is the best among the compared classifiers.

Conclusions and Future Work

We have discussed our novel approach towards a generic animal activity recognition classifier that has a high performance across different species. We analyzed the complexity in terms of memory and CPU usage between 7 classifier types. When taking into account the complexity versus performance trade-off, the DNN classifier is the most promising for our approach. We have shown a significant increase in performance when instances of two species are shared to train a classifier. Our results support the development of a generic classifier that can run on a small embedded system with good performance, as well as sustain the lifetime of online activity recognition systems.

In future work, we intend to extend our dataset with movement data from quadruped animals such as horses, cows, dogs, and cats. Thereby, we want to investigate the range of species in which generic activity recognition is possible and optimize the approach. We are planning to improve the performance of the generic classifier by means of incremental learning and online change detection [12].

Acknowledgements

This research was supported by the Smart Parks Project, which involves the University of Twente, Wageningen University & Research (WUR), ASTRON Dwingeloo, and Leiden University. The Smart Parks Project is funded by the Netherlands Organisation for Scientific Research (NWO). The authors would like to thank the Proosdij farm, Jan Pieter Meijers, and Henjo de Knecht (WUR) for their help with collecting the dataset that was used for the experiments.

REFERENCES

1. Ling Bao and Stephen S. Intille. 2004. Activity Recognition from User-Annotated Acceleration Data. *Pervasive Computing* (2004), 1 – 17. DOI: http://dx.doi.org/10.1007/b96922
2. Arno Candel, Viraj Parmar, Erin LeDell, and Anisha Arora. 2016. Deep Learning with H2O. *H2O. ai Inc.,* (2016).
3. Rich Caruana. 1997. Multitask Learning. *Learning to Learn* 75 (1997), 95–133. DOI: http://dx.doi.org/10.1007/978-1-4615-5529-2_5
4. Diane Cook, Kyle D. Feuz, and Narayanan C. Krishnan. 2013. Transfer learning for activity recognition: A survey. *Knowledge and Information Systems* 36, 3 (2013), 537–556. DOI: http://dx.doi.org/10.1007/s10115-013-0665-3
5. Ritaban Dutta, Daniel Smith, Richard Rawnsley, Greg Bishop-Hurley, James Hills, Greg Timms, and Dave Henry. 2015. Dynamic cattle behavioural classification using supervised ensemble classifiers. *Computers and Electronics in Agriculture* 111 (2015), 18–28. DOI: http://dx.doi.org/10.1016/j.compag.2014.12.002
6. ej-technologies GmbH. 2001-2017. JProfiler. http://www.ej-technologies.com. (2001-2017).
7. L A González, G J Bishop-hurley, R N Handcock, and C Crossman. 2015. Behavioral classification of data from collars containing motion sensors in grazing cattle. *Computers and Electronics in Agriculture* 110 (2015), 91–102. DOI: http://dx.doi.org/10.1016/j.compag.2014.10.018
8. Jin Hyuk Hong, Julian Ramos, and Anind K. Dey. 2016. Toward Personalized Activity Recognition Systems with a Semipopulation Approach. *IEEE Transactions on Human-Machine Systems* 46, 1 (2016), 101–112. DOI: http://dx.doi.org/10.1109/THMS.2015.2489688
9. Jacob Kamminga. 2017. Goat Orientation Data. online. (05 2017). http://ps.ewi.utwente.nl/Datasets.php
10. K Kira and LA Rendell. 1992. The feature selection problem: Traditional methods and a new algorithm. *Aaai* (1992), 129 – 134. DOI: http://dx.doi.org/10.1016/S0031-3203(01)00046-2
11. S B Kotsiantis. 2007. Supervised Machine Learning : A Review of Classification Techniques. *Informatica, An International Journal of Computing and Informatics* 3176, 31 (2007), 249–268.
12. Viet Duc Le, Hans Scholten, and P. J M Havinga. 2014. Online change detection for energy-efficient mobile crowdsensing. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8640 LNCS (2014), 1–16. DOI: http://dx.doi.org/10.1007/978-3-319-10359-4_1
13. Jacques Marais, Solomon Petrus, Le Roux, Riaan Wolhuter, and Thomas Niesler. 2014. Automatic classification of sheep behaviour using 3-axis

accelerometer data. *Pattern Recognition Association of South Africa* (2014), 1–6.

14. MATLAB. 2015. *version 8.6.0 (R2015b)*. The MathWorks Inc., Natick, Massachusetts.
15. I Mierswa, M Wurst, R Klinkenberg, M Scholz, and T Euler. 2006. YALE: Rapid prototyping for complex data mining tasks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 2006 (2006), 935–940. DOI: http://dx.doi.org/10.1145/1150402.1150531
16. Ran Nathan, Orr Spiegel, Scott Fortmann-Roe, Roi Harel, Martin Wikelski, and Wayne M Getz. 2012. Using tri-axial acceleration data to identify behavioral modes of free-ranging animals: general concepts and tools illustrated for griffon vultures. *The Journal of experimental biology* 215, Pt 6 (2012), 986–96. DOI: http://dx.doi.org/10.1242/jeb.058602
17. Paul O'Donoghue and Christian Rutz. 2015. Real-time anti-poaching tags could help prevent imminent species extinctions. *Journal of Applied Ecology* 53, 1 (2015), 5–10. DOI: http://dx.doi.org/10.1111/1365-2664.12452
18. Julie K Petersen. 2002. *Understanding Technologies Surveillance Spy Devices, Their Origins & Applications*. CRC Press.
19. Solomon Petrus. 2016. *A Prototype Animal Borne Behaviour Monitoring System*. Ph.D. Dissertation. Stellenbosch University.
20. Qiang Yang. 2010. a Survey on Transfer Learning. 1, 10 (2010), 1–15. DOI: http://dx.doi.org/10.1109/TKDE.2009.191
21. Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. 2010. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks* 6, 2 (2010), 1–27. DOI: http://dx.doi.org/10.1145/1689239.1689243
22. Yasar Guneri Sahin. 2007. Animals as Mobile Biological Sensors for Forest Fire Detection. *Sensors* 7 (2007), 3084–3099. DOI: http://dx.doi.org/10.3390/s7123084
23. L Shao, F Zhu, and X Li. 2014. Transfer Learning for Visual Categorization: A Survey. *Tnnls* PP, 99 (2014), 1. DOI: http://dx.doi.org/10.1109/TNNLS.2014.2330900
24. Muhammad Shoaib, Stephan Bosch, Ozlem Incel, Hans Scholten, and Paul Havinga. 2015. A Survey of Online Activity Recognition Using Mobile Phones. *Sensors* 15, 1 (2015), 2059–2085. DOI: http://dx.doi.org/10.3390/s150102059
25. Pekka Siirtola and Juha Röning. 2012. Recognizing Human Activities User-independently on Smartphones Based on Accelerometer Data. *International Journal of Interactive Multimedia and Artificial Intelligence* 1, 5 (2012), 38. DOI: http://dx.doi.org/10.9781/ijimai.2012.155
26. Xu Sun, Hisashi Kashima, and Naonori Ueda. 2013. Large-scale personalized human activity recognition using online multitask learning. *IEEE Transactions on Knowledge and Data Engineering* 25, 11 (2013), 2551–2563. DOI: http://dx.doi.org/10.1109/TKDE.2012.246
27. Inertia Technology. 2017. ProMove mini. online. (2017). http://inertia-technology.com/