Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Automated Usage Characterization of Mining Vehicles For Life Time Prediction

E. Jakobsson [*,**], **E. Frisk** [**], **M. Krysander** [**], **and R.Pettersson** [*]

[*] *Epiroc Rock Drills AB, Örebro, 702 25, Sweden (e-mail: {erik.jakobsson, robert.pettersson}@epiroc.com).*
[**] *Linköping University, Linköping, 581 83, Sweden (e-mail: {erik.frisk,mattias.krysander}@liu.se)*

Abstract: The life of a vehicle is heavily influenced by how it is used, and usage information is critical to predict the future condition of the machine. In this work we present a method to categorize what task an earthmoving vehicle is performing, based on a data driven model and a single standalone accelerometer. By training a convolutional neural network using a couple of weeks of labeled data, we show that a three axis accelerometer is sufficient to correctly classify between 5 different classes with an accuracy over 96% for a balanced dataset with no manual feature generation. The results are also compared against some other machine learning techniques, showing that the convolutional neural network has the highest performance, although other techniques are not far behind. An important conclusion is that methods and ideas from the area of Human Activity Recognition (HAR) are applicable also for vehicles.

Keywords: Maintenance scheduling and production planning, Neural networks in process control, Measurement and instrumentation.

1. INTRODUCTION

Automated tracking of what task a construction vehicle is performing has many benefits when one wants to predict the wear and tear of the vehicle. In the simplest case, only running time is available, and from experience this gives some notion on the current wear state of the vehicle. A slightly more advanced case would be to characterize the vehicle activities, and for how long each activity has been carried out. Different activities causes wear on different subsystems, and by collecting such data more accurate models of wear state can be accomplished. This work focuses on the characterization of vehicle usage to enable such analysis, but does not target the actual life time prediction.

For highly computerized vehicles usage characterization is trivial, since for example speed and load information can be obtained from a multitude of sensors on the vehicle. A large part of the population of mining vehicles in the world are however still low-tech and manually operated, and for such less advanced vehicles it would be beneficial to categorize the activity from a simple standalone sensor setup, such as an accelerometer. This paper targets how to characterize vehicle activities in the case where no advanced vehicle sensors are available.

The target is similar to the field of Categorization of human activity and detection of transportation mode. Human Activity Recognition (HAR) is a well studied field, mainly as a result of the readily available activity tracking bracelets that are flooding the market. To detect what type of transportation people are utilizing has also received a lot of attention, given its benefits for city planning and the possibility to receive measurements from mobile phones. Many different classification algorithms based on accelerometer data are suggested in the literature. Some researchers focus on more traditional manual feature extraction followed by a classifier, such as Principal Component Analysis (Mantyjarvi et al., 2001). A good overview of such methods can be found in (Figo et al., 2010).

Other researchers focus on using a more automated feature extraction approach, most commonly using neural networks for feature extraction and classification. Zeng et al. (2014) use a single convolution layer followed by two fully connected hidden layers, keeping the three measured accelerometer channels separate until the fully connected layers. Liang and Wang (2017) use the magnitude of the acceleration, since the orientation of a body-worn accelerometer is often unknown. The one dimensional acceleration magnitude signal is fed to a multilayer network with multiple convolution steps. Also Ronao and Cho (2016) show that multiple convolution layers are efficient for capturing higher level features. In their case accelerometer and gyroscope readings were used, resulting in 6 individual channels.

This work is based on methodology presented in (Liang and Wang, 2017), since their work involves characterizing transportation mode, such as riding a bike, driving, going by train, etc. Accelerations of public transportation vehicles are likely more similar to an earthmoving vehicle than gestures of a person, where most of the HAR research is concentrated. Like Liang and Wang (2017) we also use multiple stacked convolution layers, each reducing the dimensionality of the data and capturing increasingly higher level features. The main reason for using Convolutional Neural Networks (CNN), is their property to handle translation invariance. We wish to capture certain acceleration patters, but we do not know where in the time segments they will occur. Another benefit of CNNs is the reduced need of feature engineering.

A main difference from Liang and Wang (2017) is that we apply the methodology on vehicle data directly rather than human subjects. Doing so we can keep the 3 accelerometer channels separate since the accelerometer orientation is fixed on

Fig. 1. The type of mine truck used for this study. The accelerometer is located centered on top of the rear wheel axis.

the vehicle. Using this setup, we target the following research questions (RQ):

RQ1: Can methods used in Human Activity Recognition work also for vehicles, i.e., what accuracy can such models reach?

RQ2: Do CNNs give an advantage compared to more conventional machine learning techniques such as random forests and support vector machines?

2. DATA

Data is available from a three axis accelerometer sampled at 50 Hz on a mine truck as seen in Fig. 1. The main oscillation frequency of the mine truck is below 2 Hz, and 50 Hz is therefore deemed sufficient. The data is collected during normal operation at an active mine site. For this study some meta data such as current payload mass and vehicle speed is available enabling automatic labeling of the data.

2.1 Partitioning

The acceleration signal is partitioned into 2 s segments using 50% overlap to extract more samples from the limited data set. The length is chosen as a trade off where a larger segment would give more information on what class it belongs to, but also would increase the risk of having multiple classes within the same segment. Longer segments would also give a lower resolution on the final classification results.

An alternative method would be to select 2 s segments from time periods containing only one class. This would remove the issue of learning from mixed-class examples, but one would need to be careful to avoid any edge effects, such as acceleration patterns that occurs at specific state transitions. If for example the opening of the box would always occur on the first time instance in an *Unloading* segment, this could fool the learning process by introducing false time dependencies in the model. When evaluating the model, no true state transitions are known, and the false time dependencies could cause poor results.

The data is divided into a training set (80%) and a test set (20%). During training and hyper parameter optimization, only the training set is used, and it is in turn split with 80% for training and 20% for validation of parameter selection.

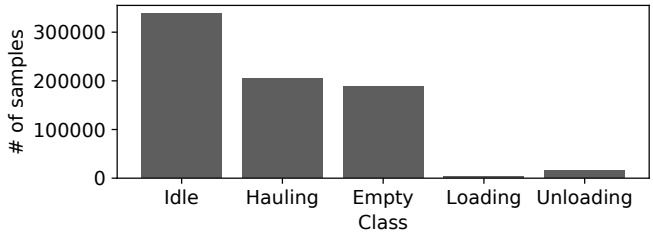

Fig. 2. The original dataset is unbalanced with a clear majority of *Idle* cases, and a minority of *Loading* cases.

2.2 Labeling

Automatic labeling is performed using this typically unavailable meta data, and the current task of the vehicle is decided from a set of logical rules. Five different tasks are defined:

- *Idle*, as the vehicle stand still doing nothing, but the vehicle control system is running.
- *Hauling*, when the vehicle transports material.
- *Empty*, when the vehicle is driving without payload.
- *Loading*, when payload is increasing.
- *Unloading*, when the vehicle dumps its load.

2.3 Normalization and balancing

The accelerometers are of dynamic type, and do not measure static level, and thus the signals are zero mean by default. To further normalize the signals, data is divided by the standard deviation of the entire training set to produce unit variance.

Unbalanced data can cause issues during neural network classification, since the majority classes are likely to dominate. The tasks of a mine truck are not equally distributed, and can differ significantly depending on application. For the data in this study, two classes are severely underrepresented as seen in Fig. 2. Both undersampling by discarding majority samples on random, and oversampling by duplicating minority samples on random are considered. The sampling techniques are implemented using the imbalance-learn package (Lemaître et al., 2017).

Two datasets are created. The undersampled dataset is limited by the minority class *Loading* and consists of 3,016 samples per class for training. The oversampled dataset is chosen to contain as many samples as the *Hauling* class, i.e., 204,527 per class. The undersampled set is used in this work, if not otherwise stated. For verification an undersampled set of 736 unique samples, i.e. not present in the training data, is used.

The dataset can be found on `https://gitlab.liu.se/erija11/mine_truck_usage_characterization_data`

3. MODELS

This section includes the different models evaluated, and also the hyper parameter search that is required to find suitable model configurations.

3.1 Baseline

A number of out-of-the-box machine learning techniques available in the Python framework Scikit learn (Pedregosa et al.,

2011), are investigated to create a baseline evaluating convolutional neural nets against other techniques. To include some time invariance and hence simplify for such techniques, also spectral features are included by calculating the frequency response of each 2 s time segment using the fast Fourier transform (FFT). The positive half of the spectrum is concatenated for the three accelerometer channels. The FFT can be regarded as a form of hand crafted feature, although a general one. The different techniques are described below.

Support Vector Machines (SVM) The evaluated SVM algorithm is available in Scikit learn based on (Chang and Lin, 2011). The SVM is tuned to use a rbf kernel, C=1000 and gamma=0.001, by performing a small hyper parameter search.

Random Forests (RF) The evaluated RF is available in Scikit learn, based on (Breiman, 2001). A small hyper parameter search showed that n=100 trees is a good choice.

K Nearest Neighbors (KNN) The evaluated algorithm is the standard KNN available in Scikit learn. Three neighbors are used, as a result from a small hyper parameter search.

Multi Layer Perceptron (MLP) A single hidden-layer MLP is evaluated. A small hyper parameter search gives that a hidden layer containing 200 neurons with Relu activation is sufficient to exploit the model structure. The output layer has 5 neurons with soft-max activation to generate a class probability.

3.2 Convolutional Neural Network (CNN)

The CNN architecture is based on the work done in (Liang and Wang, 2017), with some minor adaptations to fit the input data in this work. One important difference from their work, is that the orientation of the three axis accelerometer on a vehicle in our case is easy to keep fixed. As a result, there is no need to look at acceleration magnitude such as done in (Liang and Wang, 2017), and the acceleration channels are kept separate until the fully connected layer at the end of the network. This is similar to the late fusion of channels seen in for example (Yang et al., 2015), and identical to the fusion seen in (Zeng et al., 2014).

The overall structure of the CNN is shown in Fig. 3. The input consist of three concatenated acceleration vectors with 100 samples each, corresponding to 2 s of acceleration data. In the frequency response case, the input vector is reduced to 49x3, i.e., the positive half of the spectrum. Each convolution filter has the shape 15 x 3 and applies zero-padding to preserve size, typically referred to as *same-padding*. The filters are moved one step a time in the temporal direction. This keeps the acceleration channels separate and preserves the three channels. The number of feature maps, n, are left as a tunable hyper parameter. After each convolution, a relu activation is used, and this is followed by a max pooling operation of size 4 and stride 2 effectively reducing the temporal dimension to half the size. This is repeated 5 times. Each convolution layer also features l2-kernel regularization, where the penalty is left as a tunable hyper parameter. Also the learning rate and number of epochs are left as tunable hyper parameters. The choice of tunable parameters is based on what is common when training CNNs.

Finally a fully connected layer with 100 neurons merges the three channels, and a last layer containing 5 neurons with soft-max activations converts the result to class probabilities.

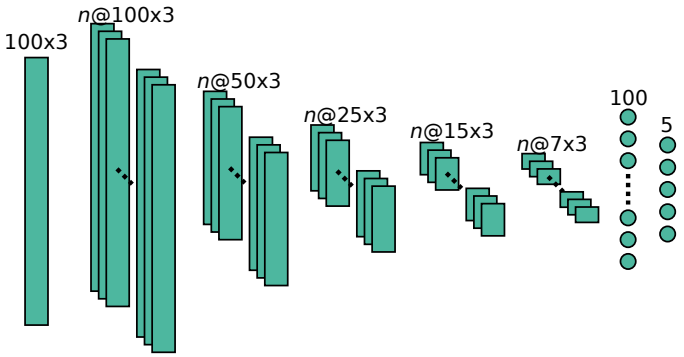

Fig. 3. Visualization of the convolutional neural network architecture for time signal input. A series of convolution and max pooling layers shrink the temporal dimension. Finally a fully connected layer merges the feature maps and a soft-max layer converts the result to class probabilities. Layers are shown as "feature maps @ data points x channels"

3.3 Hyper parameter search

Given the huge search space for hyper parameters, an optimization framework is a necessity to cover at least a small portion of it. For this work, Talos (Autonomio, 2019) is used since it offers seamless integration with Keras (Chollet et al., 2015), the software package used to implement the neural networks. Talos is used by wrapping the standard Keras syntax in a number of Talos functions, and then passing parameters as a dictionary. Some optimization strategies are provided in the framework, but for this work only full grid search was used.

During hyper parameter optimization none of the test data is used. Instead 80% of the training data is used for training, and 20% for validation. Once a set of model parameters are chosen, the full training dataset is used to train the final model.

4. RESULTS

In this section we show the results for the hyperparameter search, and the prediction results for the different models investigated.

4.1 Hyper parameter search

Results from a full grid hyper parameter search for the CNN are shown as a parallel coordinates plot in Fig. 4. To increase readability some noise is added to the plot, separating lines originating from the same point vertically. Reversing the regularization column and including accuracy further improves readability of relations between hyper parameters and accuracy. Low regularization and feature maps ≥ 64, combined with a high learning rate gives the highest accuracy. Using 500 epochs is deemed sufficient to reach the potential of the model.

The overall best model at 94% validation accuracy is selected, having the hyper parameters seen in Table 1.

Table 1. Best CNN hyper parameters.

Hyper parameter	Value
feature maps	64
kernel regularizer	0.001
epochs	500
learning rate	0.001

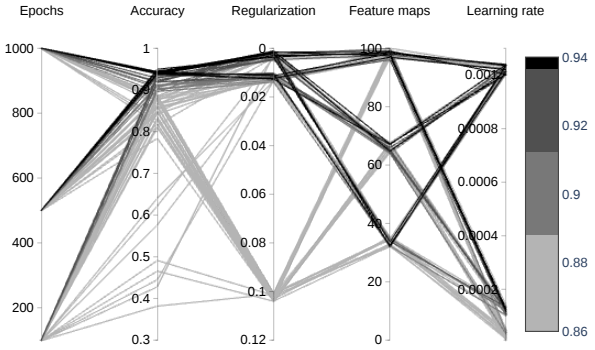

Fig. 4. The influence of hyper parameters is visualized using a parallel coordinates plot. By following the dark lines from the top of the accuracy column, it is seen how a low regularization and a high learning rate are most influential to achieve a high accuracy.

The results from the CNN are compared to the baseline classification from out-of-the box machine learning techniques as well as to the MLP network. The comparison is done on the balanced test set, making accuracy a good metric. Table 2. shows the results for both time input and spectral input.

Table 2. Accuracy results for the different machine learning techniques

Technique	Time	Spectral
SVM	0.61	0.88
KNN	0.70	0.88
RF	0.82	0.90
MLP	0.76	0.93
CNN	0.94	0.94

The CNN is able to handle the time invariance without assistance, and outperforms the other techniques for time signal input. By introducing a generic feature such as frequency response, the time invariance can be handled also by the other techniques, and the performance of the CNN is far less superior. The identical CNN performance for time and spectral input shows that the network structure is insensitive to transformations.

4.2 Confusion matrix

The best CNN model from the parameter search is evaluated on the balanced test set. The confusion matrix seen in Fig. 5 shows the performance to vary between the classes. Most confusion is seen between the *Loading*, *Unloading*, and *Idle* classes.

Fig. 6 shows examples of acceleration curves for the different classified cases. They are arranged as the confusion matrix, i.e., correct classifications are seen on the main diagonal and combinations of misclassification on the off-diagonals. Discussions on class similarities are left for section 5.2.

4.3 Oversampled dataset

The undersampling used throughout the work discards a large part of samples from all but the minority class. The samples

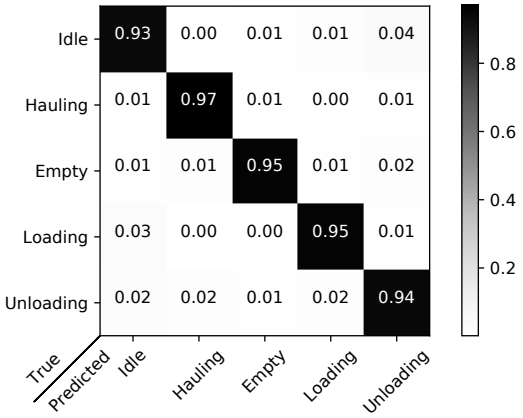

Fig. 5. Confusion matrix for the CNN evaluated on test data.

could contain useful cases for learning a more accurate model. Re-training the time signal CNN using the full dataset increases the accuracy on the test set to reach 0.96. Re-training the RF with spectral inputs shows no such improvement, and accuracy even drops to 0.87 in this case. This shows that the CNN structure is capable of using more information to further improve its performance. The decrease in RF accuracy is possibly due to over fitting on the larger dataset. The SVM method failed to converge for the larger dataset, and no result is presented.

4.4 Post processing

The classification algorithms are left blind when it comes to previous and following segments, but in reality, a lot of information is available from how the vehicle can operate. This information can be utilized. One example is short driving segments, like if the machine is in state *Hauling* and briefly shows state *Empty*. Sudden loss and gain of load is impossible, and such segments can be corrected.

A post processing scheme using information from adjacent segments is applied to remove such invalid changes between classes. The time series is partitioned without overlap, and each individual segment is classified using the CNN. This gives 5 signals showing the probability for each class over time. A 3rd order low pass Butterworth filter with a cut-off frequency of 7.5 Hz is applied forward and backward over the classifier output, resulting in zero phase shift filtering. Zero phase, or at least constant phase, is needed not to shift the different class probabilities w.r.t. each other. For each time instant, the class with maximum filtered probability is chosen as the predicted state. Fig. 7 shows the unfiltered and filtered probability signals for 3 states. At time 7 and 34 marked by star, the unfiltered unloading briefly has the largest class probability which is incorrect. After filtering, the correct class has largest probability. For the underbalanced test set, the method improves accuracy from 0.94 to 0.96.

This simple method is limited to spikes shorter than the shortest possible valid segments. This limits the usage to single outliers, since the loading segments can be as short as 4 s. More advanced post processing methods such as Bayesian Filters or Hidden Markov Models are left for future work.

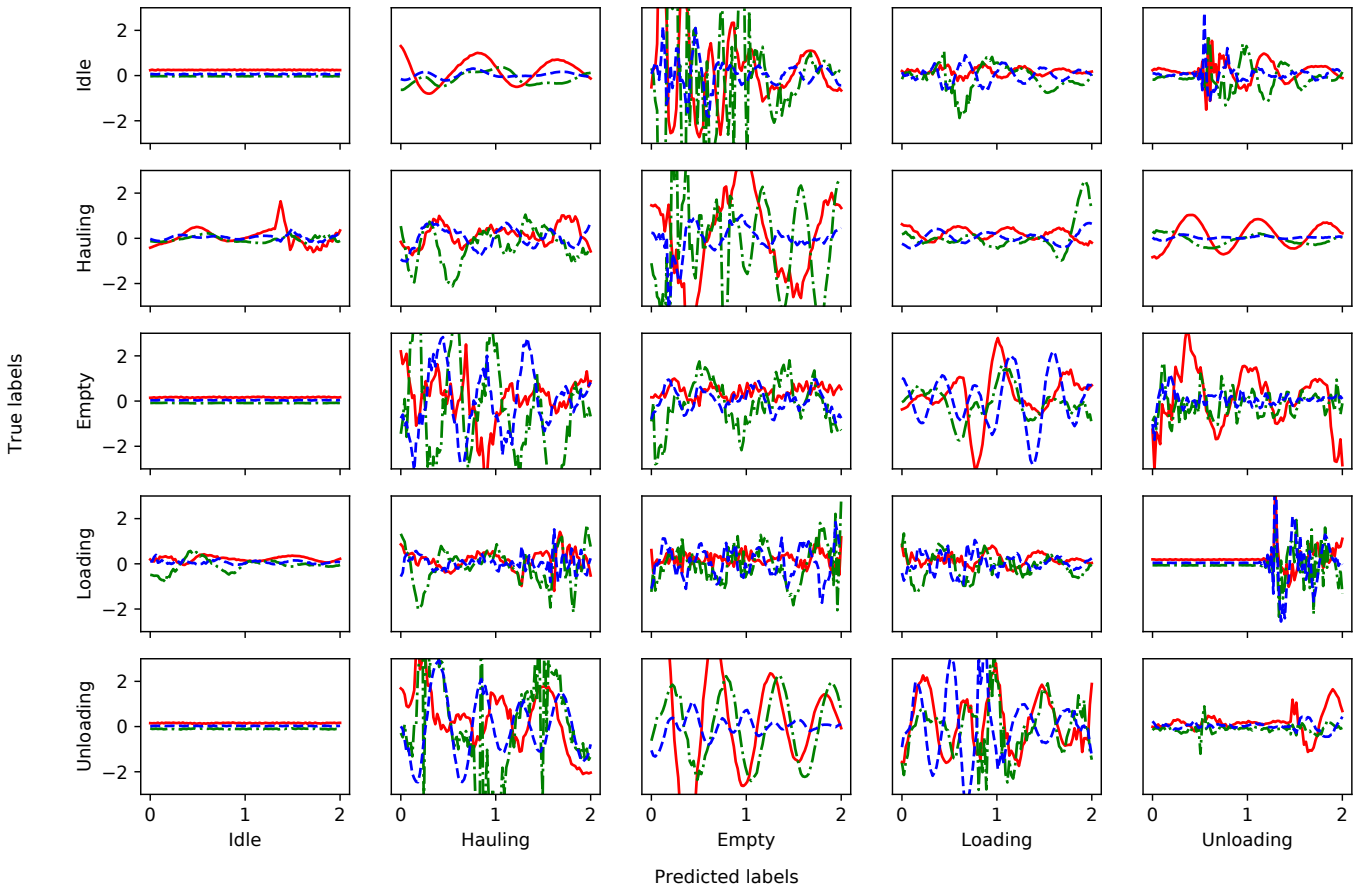

Fig. 6. Confusion matrix of different examples of acceleration segments, where red solid is in the driving direction, green dash-dot in the sideways direction, and blue dash in the vertical direction with respect to the vehicle. The main diagonal shows correctly classified samples, while the off diagonals shows different combinations of mis-classifications.

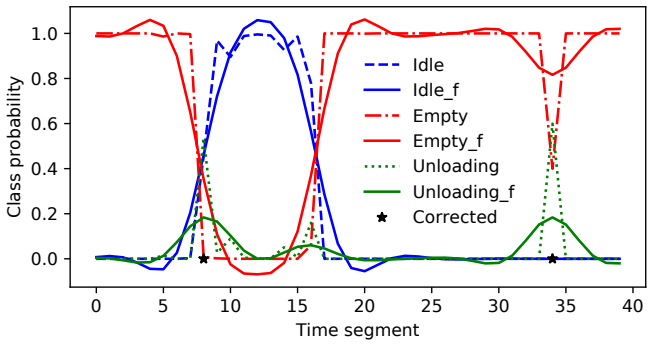

Fig. 7. The state probability time signals are filtered, allowing the removal of rapid state transitions by choosing the maximal probability from the filtered signals as seen at the stars.

5. DISCUSSION

In this section we share our thoughts on similarities between vehicle task recognition and HAR, class distinction performance, and future implementation in a life time prediction setting.

5.1 Similarities to HAR

Since the data used in this work has not been used before, it is difficult to evaluate the performance of the suggested approach. One way is to compare different techniques, as done in this paper. Another is that given the similarities to both data and techniques presented in (Liang and Wang, 2017) but with the differences:

- The accelerations are generated by vehicle movements mainly, not human gestures.
- The data is sampled at 50 Hz and contain experiments with approximately 2 s time windows.

it makes at least some sense to compare the results directly despite using different data sets.

(Liang and Wang, 2017) presents an accuracy of 75.48% for the CNN on 2 s time windows. Out of the other ML techniques investigated, they found random forests (74.09% accuracy) on spectral features to be almost as effective as CNN. This is completely in line with our work, although the absolute accuracy levels are lower in their work, which could possibly be explained by how similar the different classes are. It is reasonable that transportation mode classes such as *Taking subway* and *Taking train* are much more similar than construction vehicle classes such as *Unloading* and *Hauling*, thus complicating the classification task.

Their best results are obtained using longer time segments, around 10 s, and they reach over 94% accuracy in this case. For our data, it is not possible to increase the time segments given the short duration of some tasks.

5.2 Class distinction

An interesting point is why some classes are misclassified more than others. By looking at the acceleration signature of the misclassified samples in Fig. 6, some insights are found. For all cases classified as *Idle*, accelerations are low. Distinguishing such cases from true *Idle*, where the accelerations are (truly) low is difficult. The high mixing between *Idle*, *Loading*, and *Unloading* can possibly be explained by the presence of such low-acceleration segments in the other classes.

For the case where *Idle* segments are misclassified as some other class, Fig. 6 shows that considerable accelerations can occur also when the true label is *Idle*. Possible explanations could be mislabeled data, since the logical rules used for automatic labeling can not take acceleration into account. For example, vehicle oscillations can occur for a while after the machine has stopped. It is also possible some external force has affected the still-standing machine so accelerations have occurred.

By investigating the external signals for vehicle speed and mass for the misclassified segments, the following is observed:

- Nearly all cases misclassified as *Idle* have low velocity.
- Nearly all misclassification *Loading* as *Idle* is at the start/end of a *Loading* segment.
- Nearly all cases when *Hauling* and *Empty* are confused occurs in low velocity. This is reasonable since load-dependent accelerations are caused by vehicle movement.

A general observation is that a large portion of the misclassification occur at borderline cases.

5.3 Implementation for life time prediction

The high accuracy results enables usage in a life prediction setting which is not possible today on machines with a low number of sensors. Knowledge on the amount of time the machine spends in each state can for instance be used to adapt service intervals on a per-machine level. However, if the model is to be used to count the exact number of complete load-haul-dump cycles, more work is required since single misclassified segments would cause large errors.

The model complexity is important given implementation on automotive grade hardware. The CNN model uses 3.9 Mb memory, and predicts 1 sample in 4e-4 s on a typical laptop. The RF model uses 2.4 Mb , and predicts 1 sample in 3e-5 s. Even though the evaluation time is an order of magnitude smaller for the RF model, both would run on a typical one-card computer.

6. CONCLUSION

A CNN outperforms both the MLP and some conventional machine learning techniques for this case of vehicle activity recognition from accelerometer signals. This is in line with the results from similar networks used on HAR data sets. When a time invariant transform is introduced, other ML techniques are almost as good as CNN. Also this is consistent with earlier HAR research, and shows that the time invariance is a key difficulty with the classification task.

7. FUTURE WORK

An interesting extension would be to visualize what features the different convolution layers learn such as in (Zeiler and Fergus, 2014), and could give insights to improve the learning algorithm. Further interesting extensions would be to let the time window sweep the signal with a much finer granularity, and to see if state transitions can be found with a higher resolution.

ACKNOWLEDGEMENTS

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP)

We would like to thank Jari Hyvärinen, Ali Beg and Markus Bagge from Epiroc Rock Drills AB for their work on planning, collecting and cleaning the dataset used in this work.

REFERENCES

Autonomio (2019). Talos. `http://github.com/autonomio/talos`.

Breiman, L. (2001). Random forests, machine learning 45. *Journal of Clinical Microbiology*, 2, 199–228.

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

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Figo, D., Diniz, P.C., Ferreira, D.R., and Cardoso, J.M. (2010). Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14(7), 645–662.

Lemaître, G., Nogueira, F., and Aridas, C.K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17), 1–5. URL `http://jmlr.org/papers/v18/16-365.html`.

Liang, X. and Wang, G. (2017). A convolutional neural network for transportation mode detection based on smartphone platform. In *Mobile Ad Hoc and Sensor Systems (MASS), 2017 IEEE 14th International Conference on*, 338–342. IEEE.

Mantyjarvi, J., Himberg, J., and Seppanen, T. (2001). Recognizing human motion with multiple acceleration sensors. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 2, 747–752. IEEE.

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

Ronao, C.A. and Cho, S.B. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, 59, 235–244.

Yang, J., Nguyen, M.N., San, P.P., Li, X., and Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. In *Ijcai*, volume 15, 3995–4001.

Zeiler, M.D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, 818–833. Springer.

Zeng, M., Nguyen, L.T., Yu, B., Mengshoel, O.J., Zhu, J., Wu, P., and Zhang, J. (2014). Convolutional neural networks for human activity recognition using mobile sensors. In *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*, 197–205. IEEE.