A New Frontier for Activity Recognition – The Sussex-Huawei Locomotion Challenge

Vito Janko
Jožef Stefan Institute
vito.janko@ijs.si

Nina Reščič
Jožef Stefan Institute
nina.rescic@ijs.si

Miha Mlakar
Jožef Stefan Institute
miha.mlakar@ijs.si

Vid Drobnič
Jožef Stefan Institute
vid.drobnic@gmail.com

Matjaž Gams
Jožef Stefan Institute
matjaz.gams@ijs.si

Gašper Slapničar
Jožef Stefan Institute
gasper.slapnicar@ijs.si

Martin Gjoreski
Jožef Stefan Institute
martin.gjoreski@ijs.si

Jani Bizjak
Jožef Stefan Institute
jani.bizjak@ijs.si

Matej Marinko
Jožef Stefan Institute
matej.marinko123@gmail.com

Mitja Luštrek
Jožef Stefan Institute
mitja.lustrek@ijs.si

UbiComp/ISWC'18 Adjunct, October 8-12, 2018, Singapore, Singapore
©2018 Copyright is held by the owner/author(s).
Publication rights licensed to ACM.
ACM ISBN 978-1-4503-5966-5/18/10...$15.00.
https://doi.org/10.1145/3267305.3267518

Abstract

The Sussex-Huawei Locomotion-Transportation recognition challenge presents a unique opportunity to the activity-recognition community – providing a large, real-life dataset with activities different from those typically being recognized. This paper describes our submission (team JSI Classic) to the competition that was organized by the dataset authors. We used a carefully executed machine learning approach, achieving 90% accuracy classifying eight different activities (Still, Walk, Run, Bike, Car, Bus, Train, Subway). The first step was data preprocessing, including a normalization of the phone orientation. Then, a wide set of hand-crafted domain features in both frequency and time domain were computed and their quality was evaluated. Finally, the appropriate machine learning model was chosen (XGBoost) and its hyper-parameters were optimized. The recognition result for the testing dataset will be presented in the summary paper of the challenge [13].

Author Keywords

Activity recognition; machine learning; feature extraction; XGBoost; competition

ACM Classification Keywords

H.2.8 [Database Applications]: Data mining; I.5.4 [Applications]: Signal processing.

Introduction

The grand vision of ubiquitous computing is that our devices will know as much as possible about our context in order to provide best service. What we are doing at any given moment is certainly an important part of our context, which is why activity recognition is intensely researched. Most research on activity recognition is focused on our bodies, dealing with activities such as walking, sitting and lying. However, since we spend a lot of time in vehicles – transportation studies show that the average commute time is up to 80 minutes a day [1], and we also travel for other purposes – this is probably an area that deserves more attention. If the grand vision of ubiquitous computing is to be realized, our devices should know not only whether we are walking or sitting, but also whether riding a train or driving a car. The University of Sussex and Huawei must have come to the same conclusion, as they have collected a dataset consisting of seven months of recordings of smartphone sensors during eight modes of locomotion.

The Sussex-Huawei Locomotion-Transportation (SHL) dataset [4, 9] can be used to develop methods for activity recognition, traffic analysis, localization, sensor fusion and other purposes. These methods can support mobile services such as travel and traffic advice, adapting phone operation to the mode of locomotion (notifications, volume, wi-fi and GPS ...), using this mode in games, for music selection and a myriad of other purposes application developers will think of. Most importantly for this paper, the dataset provides an excellent challenge for ubiquitous computing researchers – to apply existing and new methods for activity (locomotion) recognition on more data than most of them are able to collect on their own, and of course to win the SHL Challenge.

The paper – as the name of the authors' team JSI[1] Classic suggests – describes a rather classical approach to activity recognition, but executed on the basis of decades of experience in the activity recognition field. It starts with preprocessing the orientation data, continues with extracting a large number of expertly crafted features and selecting the best of them, and finishes with feeding the features into a classifier with tuned hyper-parameters.

Dataset

The SHL Dataset used for this work is publicly available and thoroughly described [4]. Here we quickly list the dataset characteristics relevant for this paper.

Data comes from a variety of sensors in a smartphone (worn on the hip): accelerometer, gyroscope, magnetometer, linear accelerometer, gravity, orientation (expressed with quaternions) and barometer; all sampled with the frequency of 100 Hz. The data is labeled with eight classes: Still, Walk, Run, Bike, Car, Bus and Subway. A class label was applied to each sensor sample. In aggregate, around 266 hours of labeled data was provided.

The goal of the competition was to train a model on this data, and then use it to classify an additional 100 hours of unlabeled test data.

Data split

The provided data was split into 1-minute intervals, which were then shuffled – with the original order for the labeled data provided as part of the dataset description. Since consecutive intervals can display substantial similarities (e.g., the phone may be in exactly the same orientation), it is essential that the dataset is placed in the original order before it is split into train, test and validation set.

[1] Jožef Stefan Institute

We used the first 25% of the labeled data for validation, the second 25% for testing, and the last 50% for training. After the parameters of our machine learning (ML) model were fixed, a different distribution – 90% for training and 10% for testing was used to get an insight on how the behavior of the classifier changes if given more training data. For training of the final model, all data was used. In all cases, we verified that the activity distribution remained similar in all listed splits.

Features

In order to apply classical ML, the data has to be preprocessed and some features describing each signal extracted.

Preprocessing

First, the data was down-sampled from 100 Hz to 50 Hz. This significantly reduced the computational load of the subsequent calculations, while not significantly affecting the classification accuracy. The additional practical benefit of the downsampling lies in the reduced consumption of the sensor battery, should the system be used in a real-time setting.

Second, "virtual" sensor streams were calculated based on the real ones. Those sensor streams had the same frequency and were used in the same manner as the original data streams for calculating features. They can be grouped into three categories:

- *Magnitudes.* Sensors with three axes had the magnitude ($m = \sqrt{x^2 + y^2 + z^2}$) calculated, which was used in addition to the sensor streams of individual axes.
- *De-rotated sensors.* These were computed by de-rotating acceleration and magnetometer data from body (phone) coordinate system to NED (North-East-Down) coordinate system by multiplying them with the rotation matrix R_{NB} created with quarternions $[q_w, q_x, q_y, q_z]$ as given below. This was done in order to obtain orientation-independent sensor information, which is important to avoid overfitting to specific orientations of the phone. However, since orientation information can also be relevant, both groups of sensor streams were retained.

$$R_{NB} = \begin{bmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_w q_z) & 2(q_x q_z + q_w q_y) \\ 2(q_x q_y + q_w q_z) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_y q_z + q_w q_x) & 1 - 2(q_x^2 + q_y^2) \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_N = R_{NB} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_B$$

- *Roll, pitch, yaw.* Finally, quarternions were used to compute Euler angles – pitch, roll and yaw. Orientation is usually presented with quarternions to avoid *gimbal lock* (point of singularity), however, Euler angles are better for extracting features since each of them has a clear real-world meaning, whereas queternion components are only really meaningful when taken all together. Because of that, we only used Euler angles in the subsequent steps.

$$\begin{aligned} pitch &= \arctan\left(\frac{2(q_w q_x + q_y q_z)}{1 - 2(q_x q_x + q_y q_y)}\right) \\ roll &= \arcsin\left(2(q_w q_y - q_z q_x)\right) \\ yaw &= \arctan\left(\frac{2(q_w q_z + q_x q_y)}{1 - 2(q_y q_y + q_z q_z)}\right) \end{aligned}$$

In the end, counting all separate sensor axes, we worked with 30 different data streams.

Feature extraction

Features were calculated from 1-minute windows of data. This window size was chosen as it was the largest possible given the limitations imposed by the nature of the competition, and we achieved highest classification accuracy using it. Since the transitions between activities are rare, long windows did not have a significant negative impact on the performance. Labels were calculated for each window as the most frequent label in that window.

Nearly all features were calculated on each individual data stream. Three categories of features were computed. First, domain features that have proven themselves in our previous work in similar domains [7, 8], including in the previously-won competition [10]. These features are described in the work by Cvetković et al.[8].

Second, we tried to generate all features using the tsfresh library [5]. While the library is capable of generating a large amount of features, is seemed too slow given the size of our dataset. Consequently, we only generated some features that seemed interesting and were not included in other categories – namely minimum, maximum, autocorrelation, number of samples above/below the mean and the average difference between two sequential data samples.

Finally, we calculated some features from the frequency domain, which describe the periodicity of the signals. These features were calculated using power spectral density (PSD), which is based on the fast Fourier transform (FFT). PSD characterizes the frequency content of a given signal and can be estimated using several techniques. The simplest one is to use a periodogram, which is obtained by taking the squared-magnitude of the FFT components. An alternative to periodogram is the Welch's method, which is also widely used and commonly considered superior to periodogram. It differs from a traditional periodogram in the fact that it computes the average of the periodograms of multiple overlapping segments of the signal to reduce the variance of the PSD. In our work, we have opted to use the Welch's method to obtain the PSD.

Using PSD is only suitable when the signal is clearly periodic. As we chose to use a 1-minute window, any periodic pattern in the signals is successfully captured, as shown in Figure 1.

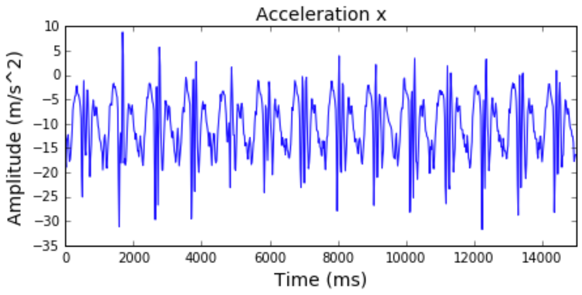

Figure 1: Periodic pattern in a 15-second accelerometer segment during walking.

We have implemented frequency-domain features as given in related work [12]. Some were slightly modified or expanded in accordance with our expert knowledge. The following features were computed.

- *Three largest magnitudes.* Three peaks with the largest magnitude from the PSD were considered. These tell us the dominant frequencies in the signal.

Both the magnitude values and the frequencies (in Hz) were taken as features.

- *Energy.* Calculated as the sum of the squared FFT component magnitudes. The energy was then normalized by dividing it with the window length.

$$energy = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2, \tag{1}$$

where $x(n)$ is the n-th FFT component.

- *Entropy.* Calculated as the information entropy of the normalized FFT component magnitudes. It helps discriminating between activities with similar energy features.

$$entropy = -\sum_{n=0}^{N-1} x(n) \log(x(n)) \tag{2}$$

- *Binned distribution.* A normalized histogram, which is essentially the distribution of the FFT magnitudes. First, the PSD is split into 10 equal sized bins ranging from 0 Hz to 25 Hz. Then, the fraction of magnitudes falling into each bin is calculated.
- *Skewness and kurtosis.* Calculated on the distribution-like PSD. Skewness and kurtosis describe the shape of the PSD. More precisely, skewness tells us about the symmetry of the distribution while kurtosis tells us about its flatness, as shown in Figure 2.

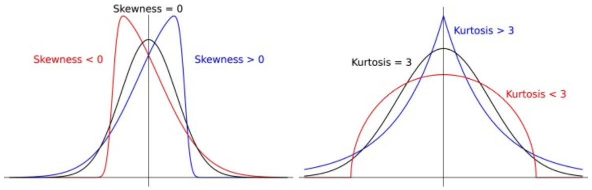

Figure 2: Distributions with different skewness and kurtosis.

In aggregate, a total of 1696 features were computed and used in the subsequent steps.

Feature selection
Since a relatively high number of features was computed, feature selection was used to remove the ones that do not contribute to the accuracy of the model in order to reduce overfitting and speed up the training process. Our feature selection consists of three steps.

In the first step, the mutual information between each feature and the label was estimated [2], where larger mutual information means higher dependency between the feature and the label.

After the features were sorted according to this value, correlated features were removed based on the Pearson correlation coefficient [3]. This has shown that roughly half of the features are redundant, which was expected due to the number of features and similar data streams. To make the process computationally feasible, only 100 features were taken at a time, starting with those with the highest mutual information with the label. Correlation was then calculated for each pair. If the correlation was higher than a certain threshold (experimentally determined as 0.8), the feature with lower mutual information was discarded. After that, next 100 features were added and the correlation between each pair was calculated again.

In the final step, features were selected using a greedy "wrapper" algorithm. A random forest (RF) classifier was first trained using only the best scoring feature. The trained model was used to predict labels for the test set and the prediction accuracy was calculated. Then the second-best feature was added and the model was trained again. If the accuracy on the test set was higher than without using this feature, the feature was kept. This procedure was repeated for all remaining features. This strict selection initially led to overfitting to the test set (accuracy was much higher compared to the validation set), so the condition for keeping a feature was made less strict: the feature was kept if the accuracy did not decrease by more than an experimentally set threshold. Using this rule, overfitting to the test set was reduced.

Best performing features came roughly evenly distributed from each of the three categories, with those coming from the de-rotated magnetometer being the most significant, followed by those from accelerometer and gyroscope. The magnetometer came as a surprise, since it is the accelerometer that usually takes the spotlight in similar problems. It might be explained by the the fact that the magnetometer is especially useful for transportation classification.

Machine learning algorithm

After obtaining the features we started building the ML model that would be used for predictions. Different ML algorithms were tried and we empirically determined the Extreme Gradient Boosting model (XGB) [6] as the best performing one. This was expected, since XGB is usually the best-performing algorithm in ML competitions.

XGB is an upgraded version of the gradient boosting algorithm. The implementation of XGB offers several advanced features for model tuning, computing environments (like parallelization across several CPU cores, distributed computing for large models, cache Optimization, ...) and algorithm enhancement (handling missing values, optimal usage of memory resources, ...). It is capable of performing the three main forms of gradient boosting (Gradient Boosting (GB), Stochastic GB and Regularized GB) and it is robust enough to support fine tuning and addition of regularization parameters. According to the author, the main difference is that XGB uses a more regularized model formalization to control overfitting, which gives it better performance.

But in order to obtain best results, XGB requires hyper-parameter tuning. For comparison we first trained the XGB model with default parameter values and then improved the model through tuning its parameters. XGB has three major groups of parameters:

- *General parameters* that relate to the boosting algorithm that we use, commonly a tree or a linear model. For this problem, we used the tree model.
- *Booster parameters* that depend on which booster is chosen. We chose the tree booster, so the parameters in this section will be tree-specific parameters.
- *Learning task parameters* that decide on the learning scenario, for example, which evaluation metric should be optimized. Regression tasks may use different parameters than classification tasks.

Since XGB has more than 30 hyper-parameters, the parameter tuning process could not be done in one step, so we did it iteratively. This means that we we optimized one parameter or maybe two combined parameters at a time, while keeping the other parameters on default values. After finding optimal value for the selected parameter, we fixed this value and started optimizing another parameter. For this iterative process, we had to start in a specific order, where first the more important parameters are optimized in order to guide the optimization toward the best overall parameter value settings.

The general approach was as follows:

1. We choose a relatively high learning rate. Learning rate defines the amount of "correction" we make at each step (each boosting round is correcting the errors of the previous round). So having a lower learning rate makes our model more robust to over-fitting and usually gets better results. But with a lower learning rate, we need more boosting rounds, which takes more time to train the model. And since we need to fit a lot of parameters we start with higher learning rate.
2. We tuned tree-specific boosting parameters (max depth, min child weight, gamma, subsample, colsample_bytree) for the chosen learning rate and number of trees.
3. We tuned regularization parameters for boosting (lambda, alpha), which can help reduce model complexity and enhance performance.
4. We lowered the learning rate to obtain the optimal parameter values.

We started by setting initial values for the parameters that were going to be optimized. We chose:

- *max_depth* $= 5$
- *min_child_weight* $= 1$
- *gamma* $= 0$
- *subsample*, *colsample_bytree* $= 0.8$
- *reg_alpha* $= 0.005$

Note that these were just initial estimates and were tuned later. We took the default learning rate of 0.1 and checked for the optimal number of trees using the cv function of XGB which performs cross-validation at each boosting iteration and thus returns the optimal number of trees required. The obtained number of thees was 140.

The first parameters tuned were *max_depth* and *min_child_weight* as they were expected to have the highest impact on model outcome. To start with, we set wider possible ranges and then we performed another iteration for smaller ranges around the best values obtained in the first step. The optimal obtained values were *max_depth* $= 6$ and *min_child_weight* $= 1$.

The accuracy increased after each step, but as the model performance increased, it became exponentially difficult to achieve even marginal gains in performance.

In the next step, we focused on tuning the *gamma* value. We used the parameters already tuned and searched for the optimal *gamma* value. The search showed that the initial value of *gamma* $= 0$ was the most suitable one.

Next, we tuned the parameters *subsample* and *colsample_bytree*. The optimal obtained values were *subsample* $= 0.65$ and *colsample_bytree* $= 0.75$.

Next, we applied regularization to reduce overfitting even though the *gamma* parameter provides a substantial way of controlling complexity. The best found value was *reg_alpha* = 0.0001.

Finally, we reduced the learning rate and increased the number of trees. We used *learning_rate* = 0.01 and *number_of_trees* = 5000. With this step, we added a significant boost in performance and the effect of parameter tuning became clearer. For comparison, the accuracy obtained with default parameter values was 88.3% and after parameter tuning we obtained the result of 90.2%.

Results

Different window lengths were tried in order to determine the optimal one. Window length of 1 minute was proven the best, as shown by the highest accuracy in Table 1. Only frequency-domain features were used in this experiment to train a RF model, as these were the fastest to compute and evaluate.

Window length	Accuracy [%]
5 seconds	67.1
30 seconds	72.5
60 seconds	74.9

Table 1: Accuracy using frequency-domain features computed on windows of length 5 seconds, 30 seconds and 60 seconds. Random Forest was used to train and evaluate the model.

All subsequent presented results were obtained by using the 1-minute-majority labels, however, per-sample accuracy was also computed. It was consistently around 0.5% lower, which is insignificant compared to the accuracy gains when using larger windows. This can be attributed to long-on-average activities, due to which activities very rarely changed in the middle of a given window.

Accuracies of the RF model after each stage of feature selection are given in Table 2. One can note that the accuracy on the test set increases with each step. The accuracy on the validation set is slightly higher than on the test set in the first three lines, presumably because those instances were easier to classify on average. In the last line ("Wrapper Strict"), the accuracy on the validation set drops, suggesting that we overfit the training data. The difference between "Wrapper" and "Wrapper strict" is in the threshold to select a feature. The threshold was higher in "Wrapper Strict", so fewer features were selected – apparently such that did not generalize very well beyond the test set. The same experiment was conducted with the XGB model, obtaining similar results: overall, the accuracy was higher, but the increase after each feature selection step was retained. The "Wrapper" feature set was thus chosen for the final model.

	Features	Accuracy [%]	
		Test	Validation
All features	1696	78.9	82.2
Correlation removed	816	80.7	83.0
Wrapper	359	82.0	83.6
Wrapper (Strict)	90	83.5	82.9

Table 2: Number of features kept and accuracy of Random Forest after each step of feature selection on both test and validation set.

After choosing the "Wrapper" feature set, we proceeded to test different classification models on the validation set. The results are shown in Table 3. XGB outperformed others by a significant margin.

Model	Accuracy [%]
k-nn	81.5
Random Forest	83.6
SVM	87.1
XGB	88.3
XGB optimized	90.2

Table 3: Accuracy of different ML models on the validation set.

Since the competition entries are rated based on the F-score metric, it should be noted that the best version of our classifier achieved an F-score of 0.90. The most common classification mistakes were between "Bus" and "Car", and between "Train" and "Subway", which was expected due to the similarity of activities within those two pairs.

In the end we trained the XGB classifier with preselected features and parameters (that worked best in previous experiments) on the whole dataset and used it to classify the unlabeled set. To get an insight in the behavior of the "fully trained" classifier, we also trained a model on 90% of the data and classified the remaining 10%, achieving the accuracy of 93.7%. We compared its classifications with the ones made with the classifier that was trained on only 50% of the data and found them very similar (they matched in 97.2% of cases).

On a "standard PC" (4 cores, 3.6 GHz and 16 GB of RAM) the training pipeline – preprocessing, extracting features, feature selection, training the model – required roughly 6 hours, and produced a model of size 43MB. Additional 3 hours were required to classify the data, mostly due to computationally expensive process of calculating the features on the whole set.

Conclusion

The SHL recognition challenge presents a uniquely large and sensor-rich dataset with data from real life. It provides a platform for creating and testing many activity-recognition algorithms, and is open to all researchers. By containing activities not common in activity recognition, such as "Train" and "Subway", it opens new challenges for the activity-recognition community.

Our approach achievedaccuracy of 90% on the validation set, far better than the baseline (70%) presented in the dataset author's paper [9]. This result is comparable with the state-of-the-art methods on similar domains. The high classification accuracy can be contributed to carefully executing every step of the standard machine learning pipeline, based on decades of experience. We started with choosing the right window size and creating new sensor streams, some of which were independent of phone's orientation. A complex set of features was computed, trying to describe the data from many different viewpoints. This process gained us roughly 10% of accuracy compared to the baseline. It was the most computationally expensive step and it required the most domain knowledge. In the next step, a complex feature selection procedure was used and increased the accuracy by 1-2%. The final steps were choosing the XGBoost model (5% accuracy gain) and optimizing its parameters (2% accuracy gain).

There are further possibilities to improve the accuracy, such as using ensemble methods, which are fairly standard in recent years. However, the focus of this project was achieving as good as possible results using only one "classical" ML method and tuning the input, description and all the parameters available. In our previous research we often recognized that such carefully hand-crafted approaches enable top results, competitive with more advanced methods [11]. In addition, a solution with just one ML method offers several advantages over more advanced and complex methods: it is easier to use and less likely to perform in some unexpected manner on unseen data, whereas more complex methods can more easily overfit.

In regards to smaller improvements, one of them is related to detecting changes of activities within a one-minute interval. Currently, the whole one-minute interval is classified with only one activity due to several reasons, such as simplicity and overall better accuracy. To detect the change of activity inside a one-minute interval, we would have to design a special procedure that detects when the activity transition happens within an interval, and then split it into smaller windows. Our estimation is that with this and some additional modifications, the accuracy could be improved by another 1-2%.

References

[1] Happiness and Satisfaction with Work Commute. https://link.springer.com/article/10.1007/s11205-012-0003-2.

[2] Mutual info score. http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html.

[3] Pearson correlation coefficient. https://en.wikipedia.org/wiki/Pearson_correlation_coefficient.

[4] Sussex-Huawei Locomotion Challenge. http://www.shl-dataset.org/activity-recognition-challenge.

[5] tsfresh. http://tsfresh.readthedocs.io/en/latest/.

[6] XGBoost. https://xgboost.readthedocs.io/en/latest/.

[7] Cvetković, B., Janko, V., and Luštrek, M. Demo abstract: Activity recognition and human energy expenditure estimation with a smartphone. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*, IEEE (2015), 193–195.

[8] Cvetković, B., Szeklicki, R., Janko, V., Lutomski, P., and Luštrek, M. Real-time activity monitoring with a wristband and a smartphone. *Information Fusion 43* (2018), 77–93.

[9] Gjoreski, H., Ciliberto, M., Wang, L., Ordonez Morales, F. J., Mekki, S., Valentin, S., and Roggen, D. The university of sussex-huawei locomotion and transportation dataset for multimodal analytics with mobile devices. *IEEE Access* (2018).

[10] Kozina, S., Gjoreski, H., Gams, M., and Luštrek, M. Efficient activity recognition and fall detection using accelerometers. In *International Competition on Evaluating AAL Systems through Competitive Benchmarking*, Springer (2013), 13–23.

[11] Pogorelc, B., Bosnic, Z., and Gams, M. Automatic recognition of gait-related health problems in the elderly using machine learning. *Multimedia tools and applications 58*, 2 (July 2012), 333–354.

[12] Su, X., Tong, H., and Ji, P. Activity recognition with smartphone sensors. *Tsinghua Science and Technology 19*, 3 (June 2014), 235–249.

[13] Wang, L., Gjoreski, H., Muraom, K., Okita, T., and Roggen, D. Summary of the sussex-huawei locomotion-transportation recognition challenge. In *Proceedings of the 6th International Workshop on Human Activity Sensing Corpus and Applications (HASCA2018)*, Springer (2018).