Distributed Data Compression for Edge Devices

Kevin Van Vaerenbergh, Tom Tourwé

Distributed data compression for edge devices.*

Kevin Van Vaerenbergh✉ and Tom Tourwé

EluciDATALab Sirris, Boulevard A. Reyerslaan 80, 1030 Brussels, Belgium
kevin.vanvaerenbergh@sirris.be

Abstract. In this paper, we elaborate on the issue of reliable storage and efficient communication of large quantities of data in the absence of continuous connectivity. We illustrate how advanced machine learning techniques can run locally at the edge, in the context of data compression related to special-purpose vehicles. Two different data compression techniques are compared by calculating general compression metrics, e.g., compression rate and root mean-squared error, while also validating the results using an event detection algorithm. These techniques exploit real-world usage data captured in the field using the I-HUMS platform provided by our industrial partner ILIAS solutions Inc.

Keywords: IoT · Distributed data analysis · Time series compression

1 Introduction

Ever more industrial assets are being instrumented and connected thanks to significant evolutions in Internet-of-Things (IoT) technology, e.g., smaller sensors and reliable connectivity. Detailed data on how/when/where an asset is used can be captured continuously, transferred to a central platform, where it is analysed via advanced data analytics technologies to extract useful insights. This enables advanced health and usage monitoring applications that help to ensure availability, reliability and safety of the equipment.

At present, such usage monitoring is mostly done at the level of the individual asset. Many companies however are operating and managing large groups of assets, and would like to apply advanced data-driven analysis techniques to extract insights across their entire fleet. Examples are fleets of vehicles operated at globally-distributed sites, wind turbines arranged within parks, compressors and pumps in industrial surroundings, etc. In that context, a reliable storage and efficient communication of large quantities of data is challenging due to the absence of continuous connectivity [11], [14].

Industrial assets can continuously gather data but are not necessarily continuously connected. Some are highly-mobile (e.g vehicles or aircraft) and sites where they are deployed can be extremely remote, hence continuous and reliable communication means are not guaranteed. In addition, different connection means are possible (e.g. fast Wi-Fi, sat-com or slower 3/4G), but each technology influences how much data can be transferred, at which speed, at what cost,

* This work is supported by the Brussels-capital region - Innoviris.

etc. Finally, data offloading opportunities can be scarce and short and offloading should not interfere with normal operations.

Consequently, data needs to be stored on-asset until it can be off-loaded to a collection point. However, storage comes at a cost and explicitly managing this cost requires carefully considering the amount of data that is retained at asset-level and eventually transferred. To ensure the most relevant data is always retained, data reduction techniques need to be considered. In Sections 2 and 3, we will detail how we consider two different data compression techniques for special-purpose vehicles in such a distributed context.

1.1 Special-purpose vehicle monitoring

We use vehicle usage data provided by our industrial partner ILIAS Solutions[1]. A fleet of special-purpose vehicles is instrumented by ILIAS' I-HUMS system which captures vehicle usage measurements. It consists of a smart sensor device, a collector antenna and a decentralized data processing server. The device contains several sensors sampling at 200Hz, e.g. 3-axis accelerometer and 3-axis gyroscope, and can connect to a vehicle's CANbus for extra data collection, sampled at 1Hz. It offloads data via the collector antenna when in the vicinity and transfers it to the server.

Fig. 1: ILIAS' I-HUMS sensor.

The dataset represents 5 months of driving data from one vehicle where three accelerometer-based aggregations are gathered, i.e., minXAcc, minYAcc and minZAcc. These signals represent the minimum value of each accelerometer axis, aggregated from 200Hz to 1Hz.

1.2 Data compression

Vehicle usage data is gathered at high frequency and to deal with this fine-grained time series data, we employ data compression. Data compression can be lossy or lossless. In contrast to most lossless techniques [13] such as [8] and [17], a much higher compression rate can be achieved using lossy techniques [15]. Many lossy compression techniques for time series have been researched [5], [10] and [12] with some having trouble on compressing all data types or having a large run-time. Therefore, we investigate intelligent data retention techniques that can be applied on edge devices. The intuition behind data retention methodologies is that all data points are not equally relevant. Many assets have "stable" periods where they are barely active, and periods with higher activity. For instance, a vehicle can be waiting at a traffic light, generating a stable period of mostly irrelevant data. Later on, the vehicle can be driving off-road in mountainous

[1] https://www.ilias-solutions.com

terrain, generating a period with relevant data to better understand/monitor it's behaviour. These methodologies intend to detect the irrelevant periods and retain that data with a lower resolution, e.g. 1 minute granularity, while keeping the periods with relevant data at a higher resolution, e.g. 1 second granularity. In this paper, we consider two data retention techniques, data compression using Recurrent Auto-encoders (RAE, Section 2) and data compression using Swinging Door Trending (SDT, Section 3) and compare them using signal comparison metrics as well as in the context of an event detection algorithm (Section 4).

2 Data compression using recurrent auto-encoders

The first algorithm we implemented is proposed by [7] and uses a recurrent auto-encoder (RAE) [9] based on an LSTM network[6] for compressing time series signals. The methodology follows three steps, (i) train a model to characterize the data input, i.e. model the stable and active periods, (ii) use the model to detect the stable and active periods in the signal to compress and (iii) compress the stable periods or retain the active periods.

The model used is a recurrent auto-encoder (RAE) (Figure 2). RAEs are a specific kind of recurrent neural nets (RNN) [16] dedicated to reconstruct an input signal using time-dependent features. It is composed of two long short-term memory nets (LSTM), one encoder and one decoder. The encoder compresses the input signal, which the decoder receives and from which it tries to reconstruct the input signal as closely as possible. Some limitations are given to the decoder in order to avoid a perfect reconstruction of the signal. These limitations ensure that the decoder focuses on the main characteristics of the input signal and discards unnecessary information.

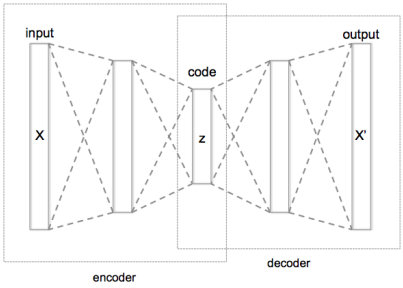


Fig. 2: Basic structure of an auto-encoder where, for this structure, the vertical bars represent the LSTM layers.

The methodology relies on two parameters; standard deviation (STD) threshold, used to segment the signal for training of the RAE and error margin, used to

identify the amount of error that the reconstructed signal can have with respect to the original signal.

2.1 Strategy

To train the RAE, we segment the signal given a STD threshold, calculated in an automatic way by bootstrapping a set of segments of distinct sizes from the original signal and averaging their STD. A STD-based threshold is preferable as it represents the evolution of the signal better than e.g., the average. The original segmentation method scans the signal sequentially and defines the periods where the STD threshold is not surpassed. Unfortunately, this approach does not take into account the change in variability of the signal. In some situations, this can lead to a segment that is a combination of a period with high variability, followed by a period of low variability. We added an extra segmentation step to avoid this by first scanning the signal to define the indexes where there is a sudden change in STD over a rolling window. The indexes identified by the first scan, together with the fixed STD thresholds are used to segment the signal properly.

We trained the RAE using the segments defined above offline, using an RAE consisting of two LSTM layers for encoding and decoding, with 32 and 16 as input size for encoding, and the reverse for decoding. The network was trained using 300 epoch, a batch size of 60 and an input size of 32. When the RAE is trained, we can use the model on the edge to compress the signals.

Two different strategies are applied in this paper; (i) we train a separate RAE for each axis, i.e., a x-axis RAE, a y-axis RAE and a z-axis RAE, and (ii) we train a global RAE using the data from all the axes. Since accelerometer data captures the movement of the asset, all three axes capture similar data and therefore can be used combined as a larger training set. We want to validate this assumption by comparing the compression results using both the separate and the combined RAE on all three axis data.

2.2 Separate RAE

For the separate compression, 6 different RAE are trained, 2 RAE's for each signal with varying input sizes, i.e., 16 and 32 RAE for x-axis, y-axis and z-axis acceleration. Each RAE is trained using only the historical data from the appropriate axis. The different signals are then compressed using the RAE that was trained for that specific axis.

Figure 3 shows a segment of the compressed z-axis signal for a certain drive, compare the compressed signals of both RAE trained using only z-axis data. We notice that the 16 size RAE does a better job of tracking the higher peaks of the signal, which in most cases represent the more informative segments.

2.3 Combined RAE

The combined approach compresses the accelerometer signals using the two RAE's that are trained using the historical data from all the axes, one with

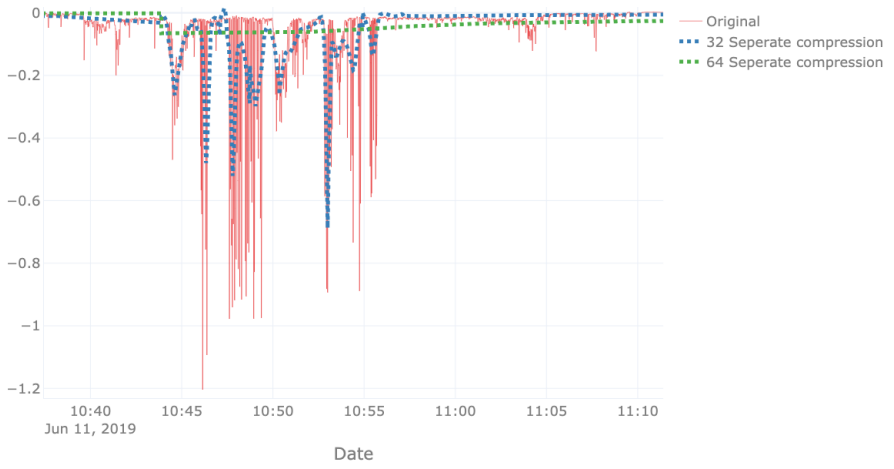


Fig. 3: Comparison of compression results for all the different RAE.

length of 16, the other with length of 32. Since the RAE is trained using all the data, the same RAE is used to compress all the different axes.

Figure 4 presents the comparison between the original z-axis signal for a drive and both compressed signals. The large overshoot just before 10:30 is most notable. We clearly see that the 32 RAE is not reconstructing that part very well. The rest of the signal is properly tracked as with the separate RAE versions.

3 Data compression using swinging door trending.

The second algorithm we implemented is proposed by [4] and uses Swinging Door Trending (SDT) [1] for intelligent data retention. This algorithm has been successfully used in the wind power ramp event detection for wind turbines [2] and solar ramp detection for solar panels [3].

The swinging door trending technique compresses a sequence of data points by using a simplified linear representation. It is computationally simple, so it can run on low-resource devices without significant overhead. The technique considers an error deviation parameter δy, which determines the error between the original signal and the compressed signal that one is willing to tolerate. The technique passes through the time series data sequentially, starting with the first two points. The first point is retained, a line is drawn between this point and the second point, and a so-called tolerance band is defined by computing the upper and lower bound, based on the deviation parameter. An iterative process is started, where the technique considers the next point in the time series and verifies whether that point falls within the tolerance band. If this is the case, the point does not need to be retained and the next data point is considered.

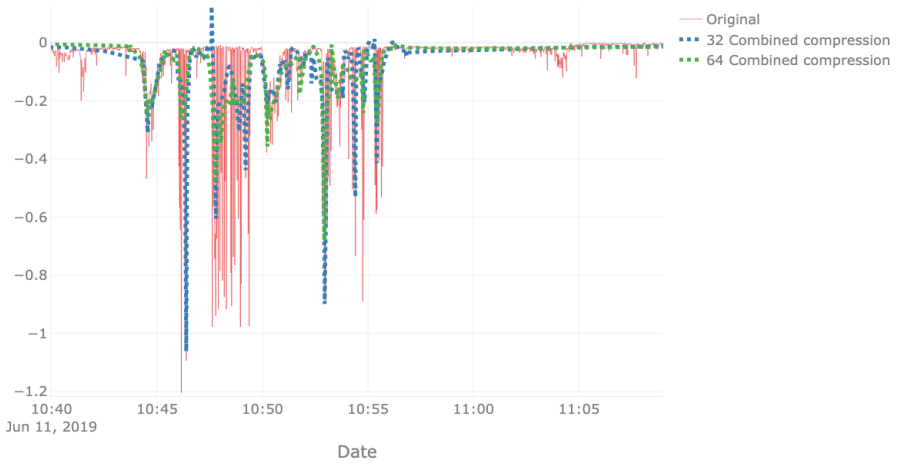
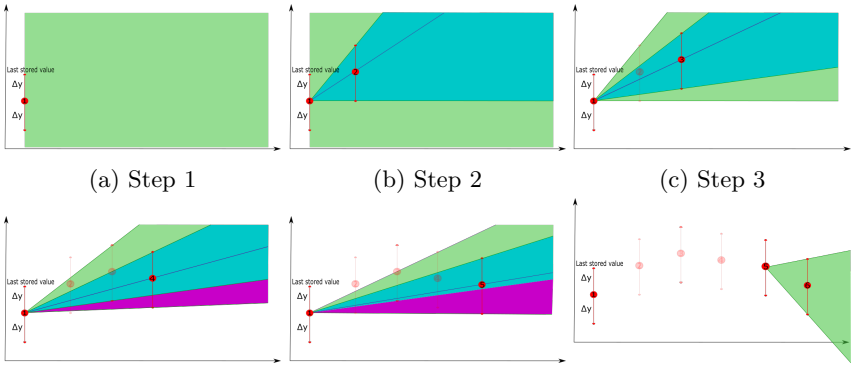


Fig. 4: Comparison of compression results for all the different RAE.

This continues until a point is reached that does not fall within the tolerance band, meaning this point can not be represented by a linear representation between the point last retained and the current point. This last point is then also retained, and a new iterative process starts. At each step, the tolerance band is also updated. When a next point is considered, its upper and lower bounds are reconsidered. The upper or lower bounds will be updated if the bounds of the next point fall below or above the previous ones, respectively, i.e. if the upper bound of the new point is lower than the previous upper bound, the upper bound is updated, and vice versa for the lower bound. This ensures that the tolerance band will always become smaller when new points are added, since the more points, the less precise the linear representation will become.

The process is illustrated by the following figures (Fig. 5). In the first step, the technique starts with the first point in the time series, the tolerance band is the whole area after the point. When the second point is considered, a linear interpolation is constructed between these two points and the tolerance band is updated using the deviation parameter δy. When a next point is selected, the new linear representation is constructed between the last point that was retained and the next point, and the new tolerance band is defined. If the next point, i.e. point 4, has the new lower bound lower than the previous lower bound, the tolerance band is not updated (the purple area will not be used as a tolerance band update), similar for the upper bound. As long as there are new points that fall in the tolerance band, they can be represented by the linear representation. Once a new point is considered that falls outside the tolerance band, e.g. point 6, the last point of the linear representation, e.g. point 5, will be retained as the compressed version of points 1 to 5. Starting from point 5 a new linear



(a) Step 1 (b) Step 2 (c) Step 3

(d) Step 4 (e) Step 5 (f) Step 6

Fig. 5: Process of SDT algorithm.

representation will be constructed with point 6 as the first point to consider. As an end result, the first 5 points will be compressed by a linear representation that is defined by point 1 and point 5, i.e. removing points 2, 3 and 4. A new linear representation will start from point 5.

3.1 Strategy

In this section, we apply the same strategies used in the previous method, (i) considering each of the three accelerometer signals separately and (ii) considering all of the signals combined. For each strategy, we compare the results using three different deviation parameter methods; (i) an automatic definition based on a mean variance bootstrapping method (**ZMEAN**), (ii) an automatic definition based on the value ranges and (**RANGE**) (iii) and fixed deviation parameter for all drives defined by a domain expert (**DEF**).

Separate SDT The first application of SDT compresses the three signals separately. For each signal and every drive, a deviation parameter is identified using the above mentioned deviation parmeter methods. In this paper, 3 different deviation parameter methods are investigated; (i) an automatic definition based on a mean variance bootstrapping method (**ZMEAN**), (ii) an automatic definition based on the value ranges and (**RANGE**) (iii) and fixed deviation parameter for all drives defined by a domain expert (**DEF**).

Figure 6 compares the resulting compressed signals with respect to the original one for a certain drive. There is clearly a big difference between the different deviation parameter selection methods. The **RANGE** compression follows the original signal best, resulting in a lower compression, while **ZMEAN** and **DEF** compression achieve a higher compression as they discard more datapoints.

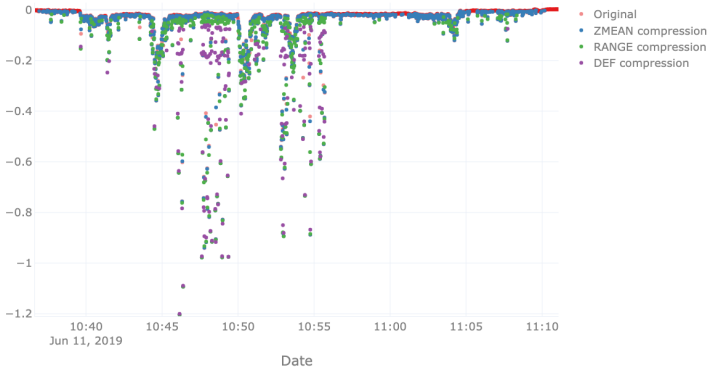


Fig. 6: Comparison on the resulting compression for a subset of the data.

Combined SDT The combined SDT compresses the signals in a combined fashion where for each drive, all three signals are compressed simultaneously as such that at each timestep, all three signals are compressed or none are compressed. The same methods that were previously mentioned are also investigated using the combined method, namely mean variance bootstrapping, value ranges and fixed deviation parameter definition.

Figure 7 shows the results of the compression of one signal using the three different deviation calculation methods. We noticed that the different methods all compress the signals differently, with the **DEF** method, compression the signal the most and the **RANGE** method compression it the least. The difference between the **RANGE** and the **ZMEAN** method are quite small but still significant is some areas, e.g., between 11:00 and 11:30.

After comparing the SDT and RAE methods internally, we will now compute several compression metrics and do a validation of the compression results of both methods against each other.

4 Validation

To define the performance of both techniques, RAE and SDT, we compare the resulting signals using data compression ratio.

Next to evaluating the compression performance, we should also validate whether the informativeness of the data is retained at a sufficiently high level.The level of informativeness of a dataset is typically linked to the application for which that dataset is used. In what follows, we will consider an event detection approach and validate whether the results of the application are influenced by applying the compression techniques. We apply the approach on the original dataset and on the compressed datasets and compare the results.

First we calculate the performance metrics.

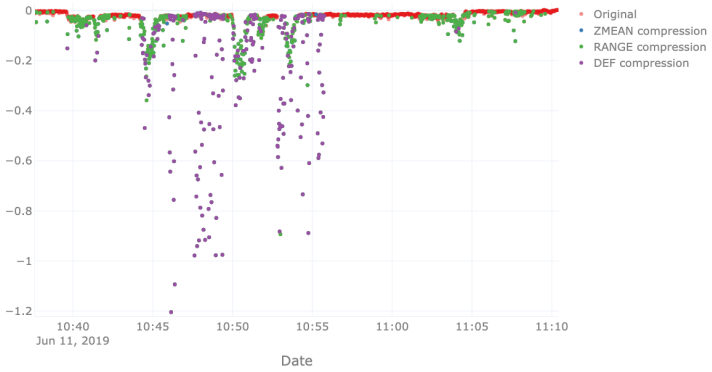


Fig. 7: Comparison on the resulting compression for a subset of the data using the combined SDT approach.

4.1 Compression ratio

In this section we will compare the achieved compression ratio per technique. The data compression ratio measures the amount of size reduction that is achieved when comparing with the size of the original signal. In Figure 8, all the compression rates are plotted, with a clear difference between the RAE compression ratios, with on average outperform the SDT ratios, except for the SDT compression using domain knowledge defined deviation parameter. This is due to the fact that this fixed deviation parameter allows a high number of compression error. Another interesting result is the compression ratio of the short RAE models using 32 input size on the x-axis accelerometer data, and the long RAE models using 64 input size. This can be due to the fact that the x-axis accelerometer data is more easy to learn using smaller input segments. For the other axes, all RAE models perform equally indicating that these axes are harder to learn. Since these axes are harder to learn, adding them as extra training data to the combined RAE models, decreases the compression performance substantially on the x-axis. A larger compression rate does not mean the performance is better as the goal of these retention techniques is to compress as much as possible without losing to much informativeness with respect of the original signal.

4.2 Event detection validation

To validate the informativeness of the resulting compressed datasets, we apply an event detection algorithm to the original and compressed datasets.

We use a simple event detection approach that defines a threshold for very high/low accelerometer values and flags data points above/below that threshold as events. We also assign a severity to the event, based on the percentage of how much the accelerometer value is exceeding the threshold. Each accelerometer

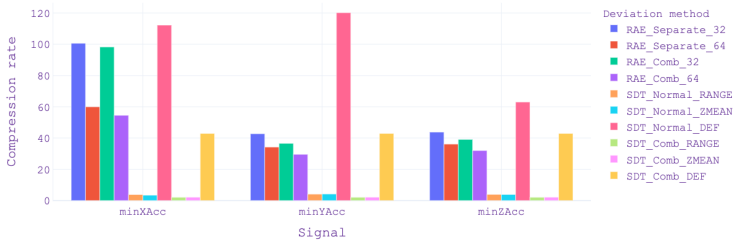


Fig. 8: Comparison of the compression rate of all different methods and parameterisations.

has his own threshold values, X-axis acceleration < -2.5, Y-axis acceleration < -1.8 and Z-axis acceleration < -2.5, and we flag the event based on severity, i.e., *WARNING* when value is between 100% and 150%, *INCIDENT* when value is between 150% and 200%, and *HARD INCIDENT* when values is above 200%.

Figure 9 represents the results of the event detection applied on the three compressed signals computed using the separate SDT algorithm (upper figure) and the compressed SDT algorithm (lower figure). We can see that the **ZMEAN** deviation parameter selection produces the best result but it still loses some information as a few *Warning* events are not found and a few *No event* events are predicted as *Warning* events. The other two approached miss-classify more events and therefore "lose" more information after compression. The combined SDT algorithm keeps 100% of the informativeness of the signal after compression for all three different approaches.

Figure 10 represents the results for the RAE approach using separately trained auto-encoders (upper figures) and combined trained auto-encoders (lower figure). The results are quite poor as most of the signals after compression "lose" a big part of the informativeness. All approaches "loose" the informativeness of the signal as they all classified every event as *No event*.

5 Conclusion

Data retention techniques aim to compress data while trying to retain a high level of informativeness. In this paper, we presented two of such techniques, Swinging Door Trending (SDT)and Recurrent Auto-Encoder (RAE).

We used both techniques to compress accelerometer data captured by a vehicle in two ways, a separate and a combined way.

A technical evaluation showed that the techniques perform differently with respect to the compression rate, with the RAE techniques able to compress up to 100 times the signal. This unfortunately comes with a price as these techniques perform very bad at the validation since they "lose" most of the informativeness of the signals after compression. In that aspect, the SDT compressed signals

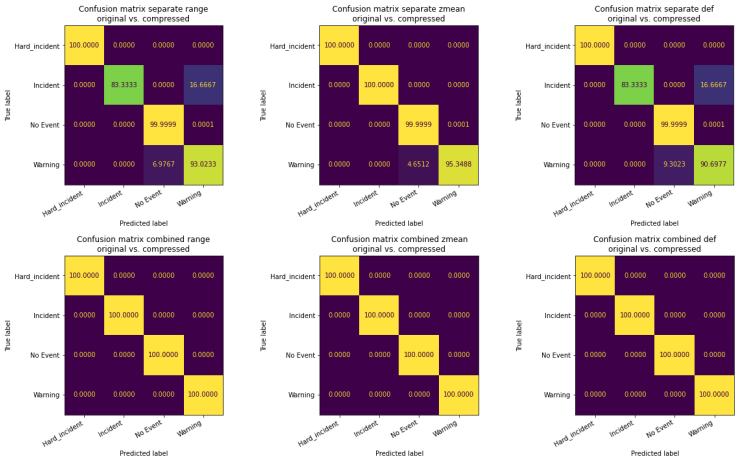


Fig. 9: Results of the event detection for the separate SDT algorithm (upper) and the combined SDT algorithm (lower).

retain a high level of informativeness after compression with the combined approach still retaining 100% of the informativeness after compression.

References

1. Bristol, E.: Swing door trending: Adaptive trend recording. In: ISA National Conference Proceedings. pp. 749–753 (1990)
2. Cui, M., Zhang, J., Florita, A.R., Hodge, B., Ke, D., Sun, Y.: An optimized swinging door algorithm for wind power ramp event detection. In: 2015 IEEE Power Energy Society General Meeting. pp. 1–5 (2015)
3. Cui, M., Zhang, J., Florita, A.R., Hodge, B., Ke, D., Sun, Y.: Solar power ramp events detection using an optimized swinging door algorithm. In: Proceedings of the ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (2015)
4. David Arias Correa, J., Sandro Roschildt Pinto, A., Montez, C., Leão, E.: Swinging Door Trending Compression Algorithm for IoT Environments. In: Anais Estendidos do Simpósio Brasileiro de Engenharia de Sistemas Computacionais (SBESC). pp. 143–148. Sociedade Brasileira de Computação - SBC (Nov 2019)
5. Elmeleegy, H., Elmagarmid, A., Cecchet, E., Aref, W., Zwaenepoel, W.: Online piece-wise linear approximation of numerical streams with precision guarantees. Proceedings of the VLDB Endowment **2**, 145–156 (2009)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**, 1735–80 (12 1997)
7. Hsu, D.: Time Series Compression Based on Adaptive Piecewise Recurrent Autoencoder (Aug 2017)
8. Huffman, D.A.: A method for the construction of minimum-redundancy codes. Proceedings of the IRE **40**(9), 1098–1101 (1952)
9. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2013)

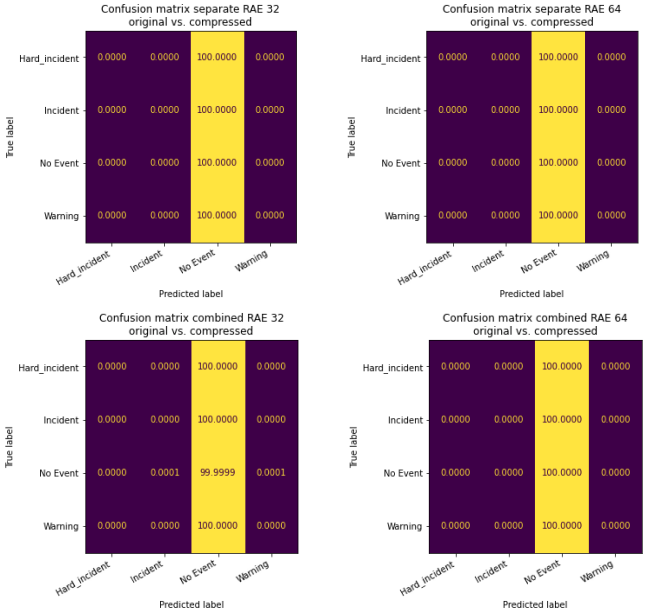


Fig. 10: Results of the event detection for the separate RAE algorithm (upper) and the combined RAE algorithm (lower).

10. Lazaridis, I., Mehrotra, S.: Capturing sensor-generated time series with quality guarantees. In: Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405). pp. 429–440 (2003)
11. Ma, T., Hempel, M., Peng, D., Sharif, H.: A survey of energy-efficient compression and communication techniques for multimedia in resource constrained systems. IEEE Communications Surveys Tutorials **15**(3), 963–972 (2013)
12. Papaioannou, T.G., Riahi, M., Aberer, K.: Towards online multi-model approximation of time series. In: 2011 IEEE 12th International Conference on Mobile Data Management. vol. 1, pp. 33–38 (2011)
13. Ringwelski, M., Renner, C., Reinhardt, A., Weigel, A., Turau, V.: The Hitchhiker's Guide to Choosing the Compression Algorithm for Your Smart Meter Data. In: Proceedings of the 2nd IEEE ENERGYCON Conference and Exhibition / ICT for Energy Symposium (ENERGYCON). pp. 998–1003 (2012)
14. Sadler, C.M., Martonosi, M.: Data compression algorithms for energy-constrained devices in delay tolerant networks. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems. p. 265–278 (2006)
15. Salomon, D.: A Concise Introduction to Data Compression. Undergraduate topics in computer science (2008)
16. Sherstinsky, A.: Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. Physica D: Nonlinear Phenomena **404**, 132306 (Mar 2020)
17. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Transactions on Information Theory **23**(3), 337–343 (1977)