Time Series Classification and its Applications

Krisztian Buza
TNG Technology Consulting
München, Germany
buza@biointelligence.hu

ABSTRACT

Time-series classification is one of the most important machine learning tasks related to time series. It is the common denominator in various recognition tasks, such as signature verification, person identification based on keystroke dynamics, detection of cardiovascular diseases and brain disorders (e.g. early stage of Alzheimer disease or dementia). This tutorial aims to give an introduction to the most prominent challenges, methods, evaluation protocols and biomedical applications related to time series classification. Besides the "conventional" time series classification task, early classification and semi-supervised classification will be considered. Both preprocessing techniques – FFT, SAX, etc. – and wide-spread classifiers – such as similarity-based, feature-based, motif/shaplet-based classifiers and convolutional neural networks – will be covered. As dynamic time warping (DTW) is the one of the key components of many time series classifiers, including recent ones based on deep learning, we will describe this technique in detail.
Slides: http://www.biointelligence.hu/pdf/timeseriestutorial.pdf

CCS CONCEPTS

• **Mathematics of computing** → *Time series analysis*; • **Computing methodologies** → *Machine learning*;

KEYWORDS

time series classification, tutorial

ACM Reference Format:
Krisztian Buza. 2018. Time Series Classification and its Applications. In *WIMS '18: 8th International Conference on Web Intelligence, Mining and Semantics, June 25–27, 2018, Novi Sad, Serbia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3227609.3227690

1 INTRODUCTION

Most real-world phenomena change over time: the activity of brain regions, the number of heart beats per minute, exchange ratios and salaries. With the words of Heraclitus, the ancient Greek philosopher: "No man ever steps in the same river twice, for it's not the same river and he's not the same man."

In the simplest case, we observe a single numeric value periodically (e.g. the temperature every hour). The resulting *univariate time series* T is a sequence of these numeric values: $T = (x_1, \ldots, x_n)$ where $x_i \in \mathbb{R}$ for $1 \leq i \leq n$. With *multivariate time series* we refer to the case, when we observe a vector in subsequent moments of time. For example, while the user is painting with her finger on the touch screen of a smartphone or tablet, horizontal and vertical coordinates together with the pressure and touched area may be observed:

$$T = \big((x_1, y_1, p_1, a_1) \ldots, (x_n, y_n, p_n, a_n)\big)$$

where $x_i, y_i, p_i, a_i \in \mathbb{R}$ for $1 \leq i \leq n$. We use the phrase time series in a broad sense and allow observations to be complex instances such as functional magnetic resonance imaging (fMRI) recordings (roughly speaking: 3-dimensional images).

W.r.t. sampling frequency, we may distinguish *evenly* sampled and *unevenly* sampled time series. In the later case, observations are usually associated with a timestamp, e.g., the unix timestamp "is the number of seconds that have elapsed since January 1, 1970."[1]

Popular data mining tasks related to temporal data [16] include time series forecasting [7], clustering [18, 19] and detection of outliers or anomalies [21, 28]. In contrast to the aforementioned problems, this tutorial focuses on time series classification.

2 TIME SERIES CLASSIFICATION PROBLEMS

With time series classification, we refer to the common denominator of recognition tasks in various domains ranging from medicine [17] over science to industry. These tasks include signature verification [24], person identification based on keystroke dynamics [25], detection of cardiovascular diseases [5, 14] and brain disorders, e.g. early stage of Alzheimer disease or dementia [29].

Formally, the *time series classification problem* can be defined as follows: given a set of classes Y, a training data $\mathcal{T}$ of time series T_i associated with their class labels $y(T_i) \in Y$, i.e., $\mathcal{T} = \big\{\big(T_1, y(T_1)\big) \ldots, \big(T_m, y(T_m)\big)\big\}$, the goal is to find a function f so that $f(T) = y(T)$ also for time series $T \notin \mathcal{T}$.

The semantics of classes varies from application to application: in case of diagnosing a disease, a positive and negative class may be given: time series recorded for patient affected by the disease belong to the positive class, whereas the time series of person who have not been diagnosed with the disease belong to the negative class. In case of signature verification or person identification, classes may correspond to different persons.

The function f is called *classifier* or *model* and it may be realized in various ways, e.g. by neural networks, support vector machines or ensembles. Regardless of which model is used, the process of finding an appropriate function f is called *training*.

In the ideal case, we expect the model to classify *new* time series correctly, in other words: the classifier is expected to predict the class of unseen time series correctly. In reality, this goal is usually not achieved completely, therefore, one aims to minimize the classification error. In order to assess the classification error in an

[1] https://www.epochconverter.com/

WIMS '18, June 25–27, 2018, Novi Sad, Serbia

ACM ISBN 978-1-4503-5489-9/18/06.
https://doi.org/10.1145/3227609.3227690

unbiased way, the error is usually measured on a set of "new" time series, called *test set*, which is disjoint from the training set. When evaluating a model, depending on the requirements of the application, various quality measures may be used, such as *accuracy* (the ratio of correctly classified instances), *F-measure* (the harmonic mean of precision and recall), *area under receiver-operator curve* (AUC) or *the area under the precision-recall curve* (AUPR).

We point out that carefully designed evaluation protocols are required in order to simulate application scenarios and to assess the quality of the models fairly. For example, in case of class-imbalanced data, a trivial classifier (which predicts the majority class for all instances) may reach high accuracy, and therefore, rather F-measure, AUC or AUPR should be used. Furthermore, one has to make realistic assumptions about the availability of training data (in some cases, such as the diagnosis of rare diseases, a large training set that contains many instances from all the classes may not be available). Additionally, one has to pay attention to split the data carefully: for example, if we do not expect to retrain and adapt the model for each patient, *patient-based* splits should be used, i.e., we have to make sure that *all* different time series of the same patient appear either in the training or in the test set, in this case, it is not allowed that some of the time series of a patient are in the training set, while other time series of the same patient are in the test set. If we want to try out the model with different hyperparameter values (such as the number of nearest neighbors in case of kNN classifiers), the proper values of the hyperparameters should be determined solely based on the training data (i.e., the training data should be split into two subsets, the one is used to train the model at different settings of the hyperparameters, the second is used to measure the quality for each setting of the hyperparameters). Cross-validation [30] may be required in order to measure standard deviation of the quality measures and assess whether a model performs statistically significantly better than another model.

Besides the above time series classification problem, there are various different problem formulations in accordance with the requirements of specific applications. When it is difficult to obtain class labels (e.g. in case of rare diseases), and the labeled data is not representative, however, large amount of unlabeled time series is available, we might learn the model using both labeled and unlabeled data. This is known as *semi-supervised classification* [11, 22]. In a slightly different setting, called *active learning*, the model is allowed to "ask" an expert for the label of a few time series, particularly, for those labels from which the model can learn the most [12]. In case of *early classification*, the model aims to determine the class label from a prefix of the time series as soon as possible, i.e., without observing the entire time series [32]. Usually, there is a trade-off between the accuracy and earliness of classification.

3 PREPROCESSING

In most applications, raw time series data is preprocessed first. Generally, the goal of preprocessing is to transform the data to a format which allows to solve the classification task accurately. The required preprocessing techniques depend on the particular application. For example, in case of unevenly sampled time series, interpolation techniques may be necessary. Next, we point out three of the most common preprocessing techniques.

Transforming time series into the frequency domain by Fourier transformation may highlight relevant properties. For example, in case of speech recognition, the signals corresponding to "*yes*" and "*no*" may be distinguished relatively simply based on their Fourier transform due to the presence or absence of high and low frequency components corresponding to the voices "*s*" and "*o*".

Averaging consecutive observations and assigning a discrete ("symbolic") value to them reduces both noise and the length of time series, thus the (absolute) computational costs of subsequent analysis steps are reduced as well. This technique is often called symbolic aggregate approximation [20].

In numerous cases, the change of a value may be more descriptive than the value itself. Let us consider, for example, handwriting recognition based on touch screen data describing the position of the tip of the pen in subsequent moments of time. In this case, instead of the raw position coordinates (x_i, y_i), it is worth to calculate how much the tip of the pen moved in horizontal and vertical directions in subsequent moments of time, i.e., instead of the time series $T = \big((x_1, y_1), \ldots, (x_n, y_n)\big)$, we consider

$$T' = \big((x_2 - x_1, y_2 - y_1), \ldots, (x_n - x_{n-1}, y_n - y_{n-1})\big).$$

At the conceptual level, this corresponds to deriving the time series according to time.

4 TIME SERIES CLASSIFIERS

Classification of time series data is challenging due to various factors. First of all, subsequent observations are naturally correlated, which is not modeled by most of the conventional classifiers, such as logistic classifier, support vector machines or naive Bayes. Furthermore, the length of time series is usually not uniform (e.g. writing a character on a touch screen does not take the *exactly* same time, the length of a heartbeats in milliseconds vary). In contrast, the aforementioned classifiers assume that the number of input features is fixed, therefore, it is not straight forward to use the observations directly.

One of the most popular ways to overcome the aforementioned difficulties is to extract a fixed number of features (such as minimum, maximum, average, standard deviation) from each time series and use these features in a conventional classifier. Useful feature may be based on the presence or absence of a local pattern (often referred to as *motif* or *shapelet*) [6, 13], or the similarity to some selected time series [1, 25].

Recent works on time series classification are based on deep learning [23, 33]. While convolutional neural networks may perform well in terms of classification accuracy [34], large amount of training data may be required and it may be difficult to interpret the resulting model and explain decisions of the classifier.

5 DYNAMIC TIME WARPING

Xi et al. claimed that 1-nearest neighbor with dynamic time warping (DTW) "is an exceptionally competitive classifier" [31]. Their empirical observations are in accordance with the theoretical results about nearest neighbor models [8, 9]. Furthermore, DTW is one of the key components of some recent time series classifiers based on deep learning, see e.g. [23]. Therefore, we review DTW and the k-nearest neigbor (kNN) classifier.

While the kNN is intuitive in vector spaces, in principle, it can be applied to any kind of data, i.e., not only in case if the instances correspond to points of a vector space. The only requirement is that an appropriate distance measure is present that can be used to determine the most similar train instances. In case of time-series classification, the instances are time-series and one of the most widely used distance measures is DTW. We proceed by describing DTW for the case of *evenly sampled, univariate time series*.

In the most simple case, while calculating the distance of two time series T_1 and T_2, one would compare the i-th element of T_1 to the i-th element of T_2 and aggregate the results of such comparisons. In reality, however, when observing the same phenomenon several times, we cannot expect it to happen (or any characteristic pattern to appear) always at the same time position, and the event's duration may also vary slightly. Therefore, DTW captures the similarity of two time series' shapes in a way that it allows for elongations: the i-th position of time series T_1 is compared to the i'-th position of T_2, and i' may or may not be equal to i.

DTW is an edit distance [27]. This means that we can conceptually consider the calculation of the DTW distance of two time series T_1 and T_2 of length L_1 and L_2 as transforming T_1 into T_2. Suppose we have already transformed a prefix (possibly having length zero or L_1 in the extreme cases) of T_1 into a prefix of T_2. Consider the next elements, the elements that directly follow the already-transformed prefixes, of T_1 and T_2. The following editing steps are possible, both of which being associated with a cost:

(1) *replacement* of the next element of T_1 to the next element of T_2, in this case, the next element of T_1 is matched to the next element of T_2, and
(2) *elongation* of an element: the next element of T_1 is matched to the last element of the already-matched prefix of T_2 or vice versa.

As result of the replacement step, both prefixes of the already-matched elements grow by one element (by the next elements of T_1 and T_2 respectively). In contrast, in an elongation step, one of these prefixes grows by one element, while the other prefix remains the same as before the elongation step.

The cost of transforming the entire time series T_1 into T_2 is the sum of the costs of all the necessary editing steps. In general, there are many possibilities to transform T_1 into T_2. DTW calculates the one with minimal cost. This minimal cost serves as the distance between the two time series. The details of the calculation of DTW are described next.

DTW utilizes the dynamic programming approach. The calculation of the minimal transformation cost is done by filling the entries of an $L_1 \times L_2$ matrix. Each entry of the matrix corresponds to the distance between a subsequence of T_1 and a subsequence of T_2. In particular, the number in the i-th row and j-th column, denoted as $d_{i,j}$ corresponds to the distance between the subsequences $T_1' = (x_1^{(1)}, \ldots, x_i^{(1)})$ and $T_2' = (x_1^{(2)}, \ldots, x_j^{(2)})$.

Consider the i-th position of T_1 and the j-th position of T_2. There are three possible cases: (i) elongation in T_1, (ii) elongation in T_2, and (iii) no elongation. If there is no elongation, the prefix of T_1 up to the $(i-1)$-th position is matched to the prefix of T_2 up to the $(j-1)$-th position, and the i-th position of T_1 is matched to the j-th position of T_2.

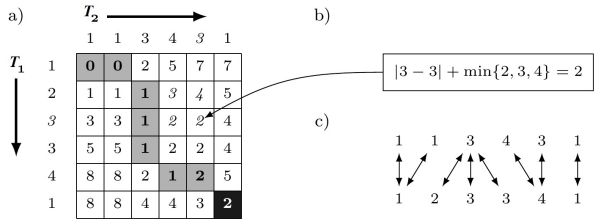

Figure 1: Example for the calculation of the DTW-matrix. a) The DTW-matrix ($c_{el} = 0$). The time series T_1 and T_2 are shown on the left and top of the matrix. The marked entries of the matrix correspond to the mapping between the both time series. b) The calculation of the value of a cell. c) The implicitly constructed mapping between the values of the both time series.

Elongation in T_1 at the i-th position means that the i-th position of T_1 has already been matched to at least one position of T_2, i.e., the prefix of T_1 up to the i-th position is matched to the prefix of T_2 up to the $(j-1)$-th position, and the i-th position of T_1 is matched again, this time to the j-th position of T_2. This way the i-th position of T_1 is elongated, in the sense that it is matched to several positions of T_2. The elongation in T_2 can be described in an analogous way.

Out of these three possible cases, DTW selects the one that transforms the prefix $T_1' = (x_1^{(1)}, \ldots, x_i^{(1)})$ into the prefix $T_2' = (x_1^{(2)}, \ldots, x_j^{(2)})$ with minimal overall costs:

$$d_{i,j} = c_{tr}(x_i^{(1)}, x_j^{(2)}) + \min\left\{d_{i,j-1} + c_{el},\ d_{i-1,j} + c_{el},\ d_{i-1,j-1}\right\}.$$

In this formula, the first, second, and third terms of the minimum correspond to the above cases of elongation in x_1, elongation in x_2 and no elongation, respectively. The cost of matching $x_i^{(1)}$ to $x_j^{(2)}$ is $c_{tr}(x_i^{(1)}, x_j^{(2)})$. This cost is present in all the three above cases. In the cases, when elongation happens, there is an additional elongation cost denoted as c_{el}.

According to the principles of dynamic programming, $d_{i,j}$ can be calculated for all i, j in a column-wise fashion. First, set $d(1,1) = c_{tr}(x_1^{(1)}, x_1^{(2)})$. Then we begin calculating the first column of the matrix ($j = 1$), followed by the next column corresponding to $j = 2$, etc. The cells of each column are calculated in order of their row-indexes: within one column, the cell in the row corresponding $i = 1$ is calculated first, followed by the cells corresponding to $i = 2$, $i = 3$, etc. In some cases (in the first column and in the first cell of each row), in the *min* function of the above formula, some of the terms are undefined (when $i - 1$ or $j - 1$ equals 0). In these cases, the minimum of the other (defined) terms are taken.

The DTW distance of T_1 and T_2, i.e., the cost of transforming the entire time series T_1 into T_2 is $d = d_{L_1,L_2}$.

The cost of transformation c_{tr} for individual observations of the time series, depend on the application. In case of univariate time series with numerical values $x_i^{(1)}$ and $x_j^{(2)}$, it can be calculated as follows:

$$c_{tr}(x_i^{(1)}, x_j^{(2)}) = |x_i^{(1)} - x_j^{(2)}|.$$

For multivariate time series, the Euclidean distance may be used:

$$c_{tr}\left((x_i^{(1)}, y_i^{(1)}), (x_j^{(2)}, y_j^{(2)})\right) = \sqrt{\left(x_i^{(1)} - x_j^{(2)}\right)^2 + \left(y_i^{(1)} - y_j^{(2)}\right)^2}.$$

An example for the calculation of DTW is shown in Fig. 1.

DTW implicitly constructs a mapping between the positions of the time series T_1 and T_2: by back-tracking which of the possible cases leads to the minimal transformation cost in each step, we can reconstruct the mapping of positions between T_1 and T_2.

For the final result of the distance calculation, usually, the values close to the diagonal of the matrix are the most important ones. Therefore, a simple, but effective way of speeding-up dynamic time warping is to restrict the calculations to the cells around the diagonal of the matrix, i.e., $d_{i,j}$ is calculated $\Leftrightarrow |i - j| \leq w$.

Further techniques to speed up DTW-based classification include lower bounding [15] and instance selection [4, 31]. The accuracy of classification can be increased by weighting schemes [10], the incorporation of additional distance measures [3], "individualized" selection of the number of nearest neighbors [2] and the usage of "advanced" variants of nearest neighbor classifiers [26].

6 CONCLUSION

Recent developments in sensor technology lead to increasing interest for the time series data mining. In this tutorial, we aimed to give an overview of time series classification techniques.

ACKNOWLEDGMENTS

The author would like to thank to Noémi Gaskó (Babes-Bolyai University, Cluj Napoca, Romania) for proofreading the manuscript and her insightful comments.

REFERENCES

[1] Krisztian Buza, Júlia Koller, and Kristóf Marussy. 2015. PROCESS: projection-based classification of electroencephalograph signals. In *International Conference on Artificial Intelligence and Soft Computing*. Springer, 91–100.

[2] Krisztian Buza, Alexandros Nanopoulos, and Lars Schmidt-Thieme. 2010. Time-series classification based on individualised error prediction. In *13th International Conference on Computational Science and Engineering*. IEEE, 48–54.

[3] Krisztian Buza, Alexandros Nanopoulos, and Lars Schmidt-Thieme. 2011. Fusion of similarity measures for time series classification. In *International Conference on Hybrid Artificial Intelligence Systems*. Springer, 253–261.

[4] Krisztian Buza, Alexandros Nanopoulos, and Lars Schmidt-Thieme. 2011. Insight: efficient and effective instance selection for time-series classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 149–160.

[5] Krisztian Buza, Alexandros Nanopoulos, Lars Schmidt-Thieme, and Julia Koller. 2011. Fast classification of electrocardiograph signals via instance selection. In *Healthcare Informatics, Imaging and Systems Biology (HISB), 2011 First IEEE International Conference on*. IEEE, 9–16.

[6] Krisztian Buza and Lars Schmidt-Thieme. 2009. Motif-based classification of time series with bayesian networks and svms. In *Advances in Data Analysis, Data Handling and Business Intelligence*. Springer, 105–114.

[7] Chris Chatfield. 2000. *Time-series forecasting*. CRC Press.

[8] George H Chen, Stanislav Nikolov, and Devavrat Shah. 2013. A latent source model for nonparametric time series classification. In *Advances in Neural Information Processing Systems*. 1088–1096.

[9] Luc Devroye, Laszlo Gyorfi, Adam Krzyzak, and Gabor Lugosi. 1994. On the strong universal consistency of nearest neighbor regression function estimates. *The Annals of Statistics* (1994), 1371–1385.

[10] Zoltan Geler, Vladimir Kurbalija, Miloš Radovanović, and Mirjana Ivanović. 2016. Comparison of different weighting schemes for the kNN classifier on time-series data. *Knowledge and Information Systems* 48, 2 (2016), 331–378.

[11] Mabel González, Christoph Bergmeir, Isaac Triguero, Yanet Rodríguez, and José M Benítez. 2018. Self-labeling techniques for semi-supervised time series classification: an empirical study. *Knowledge and Information Systems* 55, 2 (2018), 493–528.

[12] Guoliang He, Yong Duan, Yifei Li, Tieyun Qian, Jinrong He, and Xiangyang Jia. 2015. Active learning for multivariate time series classification with positive unlabeled data. In *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*. IEEE, 178–185.

[13] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. 2014. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery* 28, 4 (2014), 851–881.

[14] Argyro Kampouraki, George Manis, and Christophoros Nikou. 2009. Heartbeat time series classification with support vector machines. *IEEE Transactions on Information Technology in Biomedicine* 13, 4 (2009), 512–518.

[15] Eamonn Keogh, Li Wei, Xiaopeng Xi, Sang-Hee Lee, and Michail Vlachos. 2006. LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In *Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment, 882–893.

[16] Vladimir Kurbalija, Miloš Radovanović, Zoltan Geler, and Mirjana Ivanović. 2010. A framework for time-series analysis. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*. Springer, 42–51.

[17] Vladimir Kurbalija, Miloš Radovanović, Mirjana Ivanović, Danilo Schmidt, Gabriela Lindemann von Trzebiatowski, Hans-Dieter Burkhard, and Carl Hinrichs. 2014. Time-series analysis in the medical domain: A study of Tacrolimus administration and influence on kidney graft function. *Computers in biology and medicine* 50 (2014), 19–31.

[18] Vladimir Kurbalija, Charlotte von Bernstorff, Hans-Dieter Burkhard, Jens Nachtwei, Mirjana Ivanović, and Lidija Fodor. 2012. Time-series mining in a psychological domain. In *Proceedings of the Fifth Balkan Conference in Informatics*. ACM, 58–63.

[19] T Warren Liao. 2005. Clustering of time series data – a survey. *Pattern recognition* 38, 11 (2005), 1857–1874.

[20] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. ACM, 2–11.

[21] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. 2015. Long short term memory networks for anomaly detection in time series. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Presses universitaires de Louvain, 89–94.

[22] Kristóf Marussy and Krisztian Buza. 2013. SUCCESS: a new approach for semi-supervised classification of time-series. In *International Conference on Artificial Intelligence and Soft Computing*. Springer, 437–447.

[23] Regina J Meszlényi, Krisztian Buza, and Zoltán Vidnyánszky. 2017. Resting state fMRI functional connectivity-based classification using a convolutional neural network architecture. *Frontiers in neuroinformatics* 11 (2017), 61.

[24] Mario E Munich and Pietro Perona. 1999. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, Vol. 1. IEEE, 108–115.

[25] Dora Neubrandt and Krisztian Buza. 2017. Projection-based person identification. In *International Conference on Computer Recognition Systems*. Springer, 221–228.

[26] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. Time-series classification in many intrinsic dimensions. In *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 677–688.

[27] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing* 26, 1 (1978), 43–49.

[28] Haemwaan Sivaraks and Chotirat Ann Ratanamahatana. 2015. Robust and accurate anomaly detection in ECG artifacts using time series motif discovery. *Computational and mathematical methods in medicine* 2015 (2015).

[29] Annamária Szenkovits, Regina Meszlényi, Krisztian Buza, Noémi Gaskó, Rodica Ioana Lung, and Mihai Suciu. 2018. Feature Selection with a Genetic Algorithm for Classification of Brain Imaging Data. In *Advances in Feature Selection for Data and Pattern Recognition*. Springer, 185–202.

[30] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. 2005. *Introduction to data mining*. Boston: Pearson Addison Wesley.

[31] Xiaopeng Xi, Eamonn Keogh, Christian Shelton, Li Wei, and Chotirat Ann Ratanamahatana. 2006. Fast time series classification using numerosity reduction. In *Proceedings of the 23rd international conference on Machine learning*. ACM, 1033–1040.

[32] Zhengzheng Xing, Jian Pei, and S Yu Philip. 2012. Early classification on time series. *Knowledge and information systems* 31, 1 (2012), 105–127.

[33] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. 2015. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition.. In *IJCAI*. 3995–4001.

[34] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. 2014. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*. Springer, 298–310.