16th International Conference on Machine Vision Applications (MVA)
National Olympics Memorial Youth Center, Tokyo, Japan, May 27-31, 2019.

A very concise feature representation for time series classification understanding

Pattreeya Tanisaro
University of Osnabrück, Germany

Gunther Heidemann
University of Osnabrück, Germany

Abstract

One major problem of time series analysis, particularly of a multivariate time series, is to find their feature representations. Especially, with the emerging of deep recurrent neural networks (RNNs), researchers opt to train the networks with raw signals by using an end-to-end framework to achieve the highest classification accuracy. Their works focus on modifying the network models and fine-tuning millions of hyperparameters; however, they lack the required level of understanding of the intrinsic properties of the data. In our work, we adopted a technique for dimensionality reduction of non-time-series to transform the time series data into small sets of feature representations. Our proposed technique allows the analyst to easily visualize the feature representations of the data and detect an instance which has a potential to cause a test failure. We demonstrated the robustness of our technique by subjecting the extracted features to a conventional classification approach such as Random Forest. The datasets used for the evaluation of this task are from the known benchmarking of 15 multivariate time series datasets and two Motion Caption datasets of 27 and 65 actions. The classification results were compared with the outputs from the Echo State Networks (ESNs) and the deep Bidirectional Neural Networks (BRNNs).

1 Introduction

With the emergence of various deep neural network frameworks, the classification of time series has become more efficient than ever. However, a challenge in time series classification is to find a data representation which can be interpreted or explained to an audience when the test fails. For time series analysis this is a key issue which allows an analyst to detect the anomalies instead of obscuring them by permitting the models to overfit the data. A general approach to visualize a time series is to employ a line graph. However, the line graph does not work well for multivariate time series where inter-dependencies between many variables exist. A much more complex situation occurs if the data instances are not of equal length for many of the data features in a large dataset. There is only one known technique that lets a data analyst inspect the feature representation of the multivariate time series, namely the unthresholded recurrence plots (RPs) [4]. The RP or distance plot is heavily used for the visualization of time series because it allows any high dimensional phase space trajectories to be visualized in subspaces through a two-dimensional representation. It exhibits reoccuring phase space trajectories of dynamic systems. This technique has been employed as an action descriptor for view-independent action recognition in combination with the Bag-of-Features obtained from the Histogram of Oriented Gradients (HOG) [5]. Nevertheless, the downside of this approach is that the lengths of all motion recordings in the experiment must be truncated to an equal unit length in order to get the fixed window size. Therefore, it is not suitable for data with unequal lengths. In addition, RPs cannot be viewed together in the same coordinate system and it requires a lot of work to examine each data sequence individually.

In our work, we demonstrate a representation of a time-dependent data to be captured in a lower dimensional space where it can be understood by a traditional classification approach such as Random Forest (RF), and is easily perceived by a data analyst. *Although the applied dimensionality reduction techniques themselves are not new, however to the best of our knowledge, there was no attempt to express the temporal information in a way which allowed the time series to be inspected simultaneously in the normal Cartesian coordinate system.* The classification outputs are tested on two kinds of datasets i) general multivariate time series of 15 datasets and ii) motion capture (MoCap) of two datasets for action recognition used in [11]. Since the evaluation of this tasks should focus on the generalization of action recognition, therefore we exclude the test subjects from the group of training subjects. The other existing works, for instance, [3, 14] did not condition on separation of the subjects in their experiments; hence, we implemented two types of RNNs, a reservoir computing RNN: the ESN, and a gradient-based RNN: the BRNN, for the comparison.

2 Dimensionality Reduction of Time Series

Let p be the total number of data instances in the dataset, and for any given data instance i, the set of individual data sequences is specified by $\{X^i\}$ where $i \in \{1,..,p\}$. For any high-dimensional data sequence X^i with a fixed number of features m and arbitrary length T_i, we can interpret X^i as a real-valued matrix X with a dimension $m \times T_i$ as illustrated in Figure 1a. Pick the number of selected components c_n for

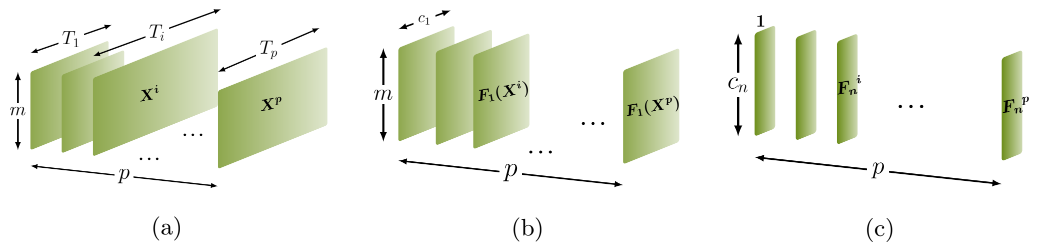

Figure 1: Transformation of time-dependent data into subspaces. (a) p instances of time series of m features with arbitrary sequence lengths T_i. (b) Results after the first transformation of F_1. From this point onward, the arbitrary size of "time dimension" T_i has become all equal with the selected principal components c_1. This new feature representation of X^i can be understood by **the conventional classification approach**. (c) The data after an arbitrary n^{th} transformation giving each signal instance of size c_n which can portray a small feature representation and its first two or three components can be projected onto the XY- or XYZ-plane.

any manifold learning F to matrix X^i where n is the number of manifold learning technique used. For the chosen first principal components c_1 at n=*1*, we obtain $F_1(X^i)$ as illustrated in Figure 1b where $\{T_i\} \geq c_1$. Hence, to apply *n-times* of dimension reduction of F to X^i for c_n components, namely $F_n(F_{n-1}(...F_1(X^i)...))$ as shown in Figure 1c, requires:

$$T_i \geq c_1 \ \forall i \in \{1, \ldots, p\} \text{ and } (m \cdot c_1) \geq c_2 ... \geq c_n \qquad (1)$$

Usually, the sequence length of any signal instance is much larger than the selected number of principal components, that is $T_i \gg c_1 \ \forall i \in \{1, \ldots, p\}$. Before applying the first order transformation, $n = 1$, we may build a feature vector by normalizing each $X^i_{j,k}$ where $j \in \{1, ..., m\}$ and $k \in \{1, ..., T_i\}$ as:

$$X^i_{j,k} \Leftarrow X^i_{j,k} - \bar{X}^i_j \qquad (2)$$

where $\bar{X}^i_j$ is the average over the sequence length T_i of feature j. Likewise, for the case of the trajectories of MoCap dataset, we first normalize the skeleton's joint positions which were computed by the marker positions following [13] by subtracting from each joint position the position of the center of the torso. The normalization by subtracting the mean is optional but is proved to enhance the visualization in many cases. For the case of different scaling of features, the rescaling prior to applying the manifold learning can be beneficial. However, normalizing time series data by dividing it by its standard deviation does not improve our visualization in general. Similar evidence was reported in [12] for human motion classification. After applying the first transformation of F_1 on each normalized X^i, the data sequence X^i can be newly represented as $F_1(X^i) \in \mathbb{R}^{m \times c_1}$ as depicted in Figure 1b. The time axis now has been replaced with the number of principal components of the first transformation. The feature vector for the second order transformation may be arranged using a concatenation of an average vector to $F_1(X^i)$ as $[\bar{X}^i_j; F_1(X^i)]$. After a second order transformation, $F_2(F_1(X^i))$, the new matrix can be shortly written as ${F_2}^i \in \mathbb{R}^{1 \times c_2}$ which is depicted in Figure 1c.

3 Experimental Setup

3.1 Multivariate Time Series Datasets.

For our experiment, we took fifteen datasets of the multivariate time series collected in [1]. These datasets were used to benchmark the classification methods in [1, 9, 7, 6]. The characteristics of each dataset are shown in Table 1 are i) the number of the attributes, ii) the lengths of sequences in the dataset, iii) the number of output classes, iv) the number of training data and v) the number of testing data. We grouped these datasets into four categories according to the levels of difficulty based on the classification results from DTW [1] which is the state-of-the-art approach for time series classification. These four levels are *i) very difficult* to solve datasets which have an error rate greater than 20%. These datasets are indicated with the deep blue squares (■) in front of the dataset names. *ii) difficult datasets* are marked with orange circles (●), which have an error rate in the range of (10-20]%. *iii)normal difficulty* with the error rates in the range of (5-10]%. The datasets are marked with the green triangle (▲), and *iv) easy datasets* indicated with pink diamond (♦). They have the error rates not higher than 5%.

3.2 Motion Capture Datasets.

We extended our experiment by selecting two different MoCap datasets, the UTD-MHAD [2] and the HDM05 [8] to demonstrate the effectiveness of our proposed technique. The test subjects were excluded from the training set to examine the generalization of action recognition.

UTD-MHAD consists of 27 different actions performed by eight subjects. Each subject repeated the same action four times. The dataset contains a total of 861 trials or data instances, where three sequences were corrupted and removed from the dataset on the official website. The training was performed on six subjects, and two subjects were left out for the test. The recognition rate was reported on an average of 28 combinations. This dataset was recorded using 20 markers.

HDM05 was originally made up of 130 classes consisting of five subjects performing actions with and without repeating the same cycles separately. This created a total of 2343 instances. We followed [3, 14, 11] in grouping non-repetitive and repetitive motions together yielding 65 actions. This dataset has been heavily biased on some actions and their lengths. For example, *walk* and *elbowToKnee* contain 94 and 80 trials, respectively; while for about 20 actions have less than 20 trials, i.e., *throwBasketball*, *throwFarR* and *jumpDown* having only 14 trials each. Nonetheless, since we focused on the action recognition of unseen subjects, four subjects were used in the training set and one subject was used for the test. We reduced the original number of markers to 19, where some nearby sensors e.g., on the spine were merged.

3.3 Configurations of the Classifiers

We employed two known linear and non-linear manifold learning such as PCA and Kernel PCA in order to get the features $F_1(X^i)$ in combination with RF with 50 and 100 trees for classification. Our proposed approach is abbreviated as DRe in the results. To display all data simultaneously in two- and three-dimensional projections, we select various manifold learning algorithms F_2 for the best visualization. For the setup of RNNs, we adopted many configurations and took the one with best output performance. The ESN configurations in this experiment followed the guideline in [10]. The number of neurons was varied in the range of 200-600 neurons with 10, 30 and 50% sparsity. We also applied the spectral radii of 1.0 and 5.0, a leaky rate of 0.1 and 0.9 and a fixed regularization coefficient of 0.1. For BRNNs, we created more than one hundred configurations with various depths and widths of BRNNs and picked the best models shown in Table 1. Following the setup of deep BRNN geometries in [11], we varied the size of the networks in the range of $2 \times \{200-600\}$ neurons. The model with one layer of BRNNs with 2×500 neurons is presented as BR^{500}, where 500 indicates the number of neurons in one direction. Therefore, for the two hidden layers of BRNNs with 100 neurons in one direction for each layer is written shortly as $BR^{2L\cdot100}$ and so on. For three hidden layers of BRNNs, we simply took the best outcome from several configurations and referred to it as $BR3L$. Furthermore, we experimented using both GRUs and LSTMs as neurons.

4 Experimental Results and Discussion

4.1 Multivariate Time Series

Table 1 shows the error rates of 15 datasets from ten classifiers. The results from left to right are, DTW and LPS taken from [1] and our implementation as the following: ESN, BRNNs with one hidden layer ($BR1L$), four strategies of BRNNs with two hidden layers ($BR^{2L\cdot100}$, $BR^{2L\cdot150}$, $BR^{2L\cdot250}$, and $BR^{2L\cdot500}$), BRNNs with three hidden layers ($BR3L$), and our proposed method (DRe). Next to the results of DRe appears one of the two symbols, ♥ and ✖, to indicate whether it makes sense (♥) to apply the dimensionality reduction to the dataset. We prefer (♥) the DRe when the following two conditions meet: i) its error rate is lower than two third among DTW, LPS and ESN, and ii) its error rate is less than 10%. The LPS generally performs much better than DTW and has two outstanding results which are difficult to be solved by other classifiers; they are *Libras* and *UwaveMTS*. The ESN also gives satisfactory results for most datasets and becomes the winner for *CharTrajectories* and *JapaneseVowels*; nonetheless, it has a problem to classify three datasets marked with ●. The $BRNN1L$ is the winner for most datasets. Interestingly, however, all BRNNs perform worst on the *NetworkFlow*. This is probably caused by the characteristics of the attributes. The *NetworkFlow* represents a network traffic protocol where a series of network packets defines a sequence. Each packet consists of four attributes which are used to identify the applications that generated the traffic flow. These attributes are the packet size, transfer direction (either 0 or 1), payload and the duration. Whereas the payload and packet size are in the magnitude of a few thousands but the direction can be either 0 or 1. Therefore, this might cause a problem for the gradient computation. Furthermore, it is important to note that the $BR1L$ performance is better than of the deep BRNNs for most datasets here.

For DRe, five datasets are considered unsuited (✖) to be processed by DRe method. This is due to: i) the transformation of the matrix of $m \times T_i$ to $m \times c_1$ which is constrained by T_i and m_i, and ii) the characteristic of all data classes which should be captured by $m \times c_1$. That is $m \times c_1$ should be sparse to be captured by RF or specifically "sufficiently greater" than the number of classes. These two restrictions can be explained by the characteristics of datasets, which are: i) restricted by T_i, i.e., *ArabicDigits* which has the shortest length of 4 yielding a matrix of maximum size 13×4 for classifying 10 classes, or ii) restricted by m if the dataset has a small number of attributes but requires many output classes, for instance, *CharTrajectories* which has 3 attributes for 20 classes, *EGG* which has 2 attributes for 2 classes, *Libras* which has 2 attributes for 15 classes and *UwaveMTS* which has 3 attributes for 8 classes. Hence, we can easily notice that the pleasing results

Dataset	nAttr	Length	nClass	nTrain	nTest	DTW	LPS	ESN	$BR1L$	$BR^{2L\cdot 100}$	$BR^{2L\cdot 150}$	$BR^{2L\cdot 250}$	$BR^{2L\cdot 500}$	BR3L	DRe
▲ArabicDigits	13	4-93	10	6600	2200	0.092	0.029	0.070	0.003	0.010	0.016	0.009	0.010	0.007	0.244 ✖
■AUSLAN	22	45-136	95	1140	1425	0.238	0.246	0.094	0.061	0.178	0.109	0.061	0.060	0.095	0.096 ♥
♦CharTrajectories	3	60-182	20	300	2558	0.033	0.035	0.023	0.033	0.043	0.046	0.045	0.042	0.048	0.186 ✖
▲CMUsubject16	62	127-580	2	29	29	0.069	0.000	0.000	0.000	0.172	0.379	0.034	0.483	0.000	0.000 ♥
▲DigitsShape	2	30-98	4	24	16	0.069	0.000	0.000	0.000	0.172	0.379	0.034	0.483	0.000	0.000 ♥
●ECG	2	39-152	2	100	100	0.150	0.180	0.270	0.160	0.210	0.200	0.200	0.160	0.210	0.250 ✖
■JapaneseVowels	12	7-29	9	270	370	0.351	0.049	0.011	0.016	0.024	0.032	0.041	0.027	0.022	0.043 ♥
♦KickvsPunch	62	274-841	2	16	10	0.100	0.100	0.100	0.200	0.500	0.500	0.500	0.500	0.400	0.000 ♥
●Libras	2	45	15	180	180	0.200	0.097	0.206	0.156	0.328	0.272	0.322	0.250	0.256	0.617 ✖
■NetworkFlow	4	50-997	2	803	534	0.288	0.032	0.034	0.779	0.779	0.779	0.779	0.779	0.779	0.028 ♥
●PEMS	963	144	7	267	173	0.168	0.156	0.278	0.058	0.191	0.185	0.260	0.231	0.202	0.092 ♥
♦Shapes	2	52-98	3	18	12	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000 ♥
▲UwaveMTS	3	315	8	896	3582	0.071	0.020	0.089	0.039	0.155	0.133	0.051	0.041	0.044	0.551 ✖
♦Wafer	6	104-198	2	298	896	0.040	0.038	0.028	0.035	0.077	0.085	0.057	0.106	0.079	0.020 ♥
♦WalkvsRun	62	128-1918	2	28	16	0.000	0.000	0.000	0.000	0.000	0.188	0.000	0.188	0.000	0.000 ♥

Table 1: The characteristics of 15 datasets and their error rates in ten classifiers.

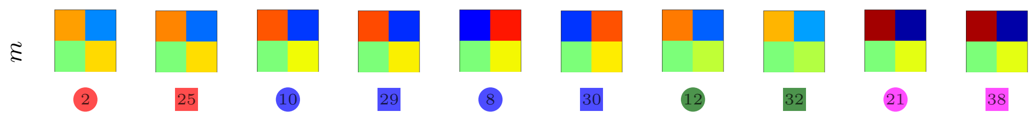

Figure 2: Feature vectors of ten instances of the *DigitsShape* dataset which have two attributes using PCA with two principal components for the classification. Below each feature is the label of the corresponding instance indicated with the number. Each labeled color matches each data class in Figure 3. The training data is presented using circles (●) while the test data is presented using squares (■).

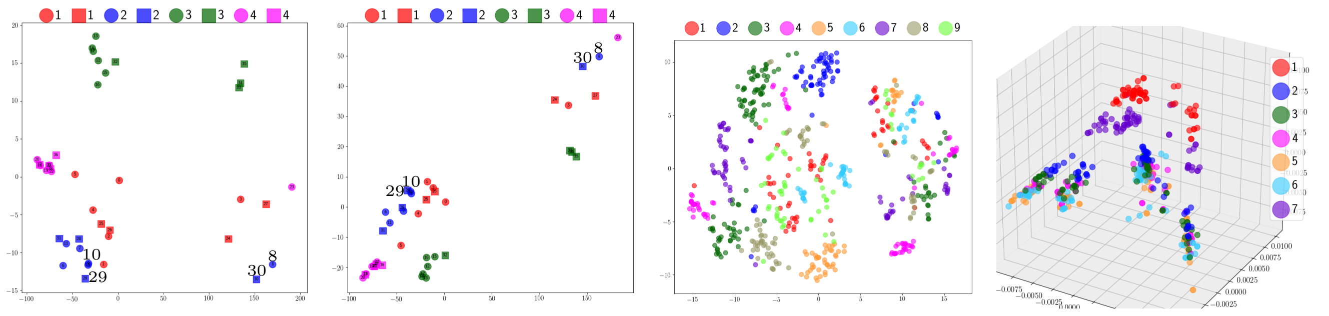

Figure 3: The two left images show the second order transformations of *DigitsShape* using only the first two principal components projected on the two-dimensional plane. The first manifold learning for classification is the PCA followed by two different second transformation approaches "for visualization in 2D projection" using PCA, and MDS, respectively. Four labels, 8, 10, 29, 30 of class "2" (■/●) are enlarged to verify the conformity of their feature representation as shown in Figure 2. The next two images are the projections of 640 instances (of training and test data) of *JapaneseVowels* and 440 instances of *PEMS* (of 963 attributes for each instance).

of *DRe* are obtained for datasets which have many attributes without restriction on the sequence lengths as we can see the outstanding results from *Wafer*, *PEMS* and *NetworkFlow*. Most important, the main benefit of this approach is that we can extract a concise feature which is understandable by the data analyst. Figure 2 shows the feature presentations of ten instances from the *DigitsShape* of 2 attributes to classify 4 classes. Figure 3 shows visualizations on the Cartesian coordinates of *DigitsShape*, *JapanesesVowels* and *PEMS*. Moreover, we further investigate the three-dimensional projection of 1194 instances of *Wafer* as displayed in Figure 4. Ten corresponding feature representations of the *Wafer* dataset are depicted next to its 2D projection on the rightmost. The *Wafer* refers to a silicon wafer in a semiconductor manufacture. Each instance consists of six variables recorded during the etching process and is marked as normal (●) or defective (●).

4.2 Motion Capture

The error rates of two MoCap datasets of 27 actions of the MHAD and 65 actions of the HDM05 using PCA for the transformation together with RF can be found in Table 2. The outputs from BRNNs were ob-

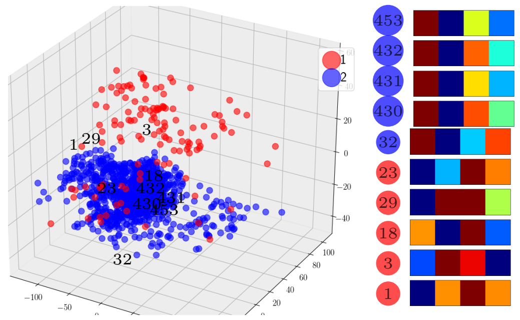

Figure 4: The visualization of projecting the features into 3D space. The image on the left shows 1194 instances of *Wafer*. The feature representations of ten instances of *Wafer* are displayed on the right. The numbers in front of the feature representations are the instance's ids for tracking them in the 3D space.

Dataset	ESN	$BR1L$	$BR^{2L\cdot100}$	$BR^{2L\cdot250}$	$BR^{2L\cdot300}$	$BR^{3L\cdot200}$	DRe
MHAD27	0.167	0.100	0.213	0.086	0.127	0.162	0.196
HDM05	0.410	0.254	0.254	0.185	0.187	0.207	0.308

Table 2: The error rates of 27 actions in the MHAD and 65 actions in the HDM05.

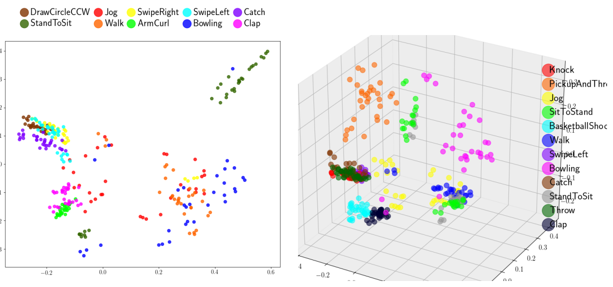

Figure 5: Selected actions of MHAD for visualization. Left image is the result of a 2D projection of ten selected actions and the right image is the result of 3D projection of twelve actions.

tained by fine-tuning hyperparameters for more than one hundred configurations. Although our approach is not better than the BRNNs in general, its performance is comparable. In addition, it took about 10 minutes to complete 28 folds of training and testing the MHAD27, 45 minutes on the ESNs with 500 neurons on Intel i7-3770 CPU 3.40GHz with 16 GB RAM running on one core, while it took three and a half hours on Intel Xeon 3.70GHz 64GB RAM with GeForce GTX TITAN X for one hidden layer of BRNNs of 2×500 neurons. The results of *DRe* are reproducible and there is no need to worry about fine-tuning the parameters.

Figure 5 shows the projections of two small subsets of the MHAD. For the image on the left, the actions involve the movements with just an arm, such as *SwipeRight* (●), *SwipeLeft* (●), *Catch* (●) and *DrawCircleCCW* (●) are plotted close to each other. While the actions which engage the movements of arms and legs, for instance, *Jog* (●) and *Walk* (●) are drawn pretty close, the action *StandToSit* (●) and *Bowling* (●) are drawn afar from the other actions. The image on the right shows twelve actions in the 3D space. Similar to the left image, the actions which engage only the movements of one arm such as *Catch* (●) , *Knock* (●) , *SwipeLeft* (●), and *Throw* (●) are drawn very close to each other. The actions which are sparsely distributed in the plot such as *Bowling* (●), *Jog* (●) and *PickAndThrow* (●) are the actions which have more free movement in space. The *SitToStand* (●) and *StandToSit* (●) are located near each other.

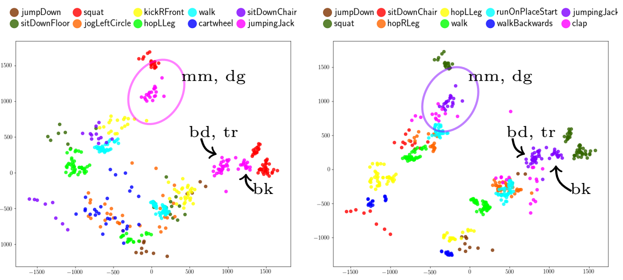

Figure 6: Two-dimensional projections of selected actions in the HDM05. When the actions have changed between the two images on the left to the right, the distributions of the same actions remain the same. These actions are *jumpDown* (●), *squat* (● → ●), *hopLLeg* (● → ●), *walk* (● → ●), *sitDownChair* (● → ●), and *jumpingJack* (● → ●).

Two images in Figure 6 show the distributions of the same actions on two different subsets (the image on the left vs. the right). Notice the actions which are separated into two groups. This is because two of the subjects "mm" and "dg" are positioned in perpendicular direction to the opposing three subjects "bd", "bk" and "tr". Figure 7 can explain why we have such behavior in the plot. From a camera viewpoint, posing actions to 0 and 90 degree creates different trajectories while keeping the correlations of the joints of that movement. The unthresholded RPs of *jumpingJack* and *walking* can be found in Figure 8 on the left and our extracted features are illustrated on the right. The feature representation from our proposed algorithm show that the repetition of the patterns still gives the same fixed concise feature. Moreover, the proposed algorithm is robust against noise as displayed for action indexing 302, while the unthresholded RPs shows its sensitivity to this small noise.

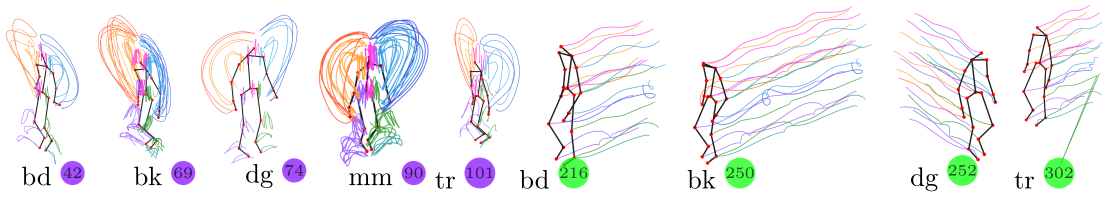

Figure 7: Nine trials of subjects "bd","bk","dg", "mm" and "tr" in the HDM05 performing *jumpingJack* and *walk*

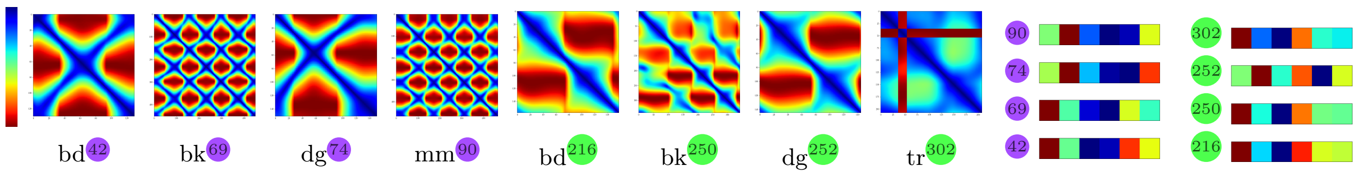

Figure 8: The unthresholded RPs of actions in Figure 7 versus our representations using six components.

5 Conclusion

We have presented an approach to represent a time series as a small set of features. We adopted conventional dimensionality reduction techniques such as PCA and KernelPCA to capture the intrinsic properties of the signals. The robustness of our approach has been proven by employing a traditional classifier with these representations. The results were compared using original signals with two types of RNNs with hundreds of configurations. The main benefit of our approach is that regardless of their lengths and the number of features, the time series can be represented in a very concise manner. Furthermore, we can visualize a large amount of time series data simultaneously in a Cartesian coordinate system. An instance which has a unique property would be laid afar from its group. Although our approach has a few limitations, nevertheless its strength lies in the fact that it is very simple to implement and lightweight because the transformation operation is just a matrix decomposition.

References

[1] Baydogan, M.G., Runger, G.: Time series representation and similarity based on local autopatterns. Data Mining and Knowledge Discovery **30**(2), 476–509 (2016)

[2] Chen, C., Jafari, R., Kehtarnavaz, N.: Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In: Proceedings of IEEE International Conference on Image Processing. pp. 168–172 (2015)

[3] Du, Y., Wang, W., Wang, L.: Hierarchical recurrent neural network for skeleton based action recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015)

[4] Eckmann, J.P., Kamphorst, O.S., Ruelle, D.: Recurrence plots of dynamical systems. Europhysics Letters **4** (Nov 1987)

[5] Junejo, I.N., Dexter, E., Laptev, I., Perez, P.: View-independent action recognition from temporal self-similarities. IEEE Trans. Pattern Anal. Mach. Intell. **33**(1), 172–185 (2011)

[6] Karim, F., Majumdar, S., Darabi, H., Harford, S.: Multivariate lstm-fcns for time series classification (2018)

[7] Luczak, M.: Combining raw and normalized data in multivariate time series classification with dynamic time warping. Journal of Intelligent and Fuzzy Systems **34**(1), 373–380 (2018). https://doi.org/10.3233/JIFS-171393

[8] Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B., Weber, A.: Documentation mocap database hdm05. Tech. Rep. CG-2007-2, Universität Bonn (2007)

[9] Schäfer, P., Leser, U.: Multivariate time series classification with WEASEL+MUSE. CoRR **abs/1711.11343** (2017), http://arxiv.org/abs/1711.11343

[10] Tanisaro, P., Heidemann, G.: Time series classification using time warping invariant echo state networks. In: 15th IEEE International Conference on Machine Learning and Applications, (ICMLA) (2016)

[11] Tanisaro, P., Heidemann, G.: An empirical study on bidirectional recurrent neural networks for human motion recognition. In: 25th International Symposium on Temporal Representation and Reasoning (2018)

[12] Tanisaro, P., Lehman, C., Sütfeld, L., Pripa, G., Heidemann, G.: Classifying bio-inspired model in point-light human motion using echo state network. In: The 26th International Conference on Artificial Neural Networks (ICANN), 2017 (2017)

[13] Tanisaro, P., Mahner, F., Heidemann, G.: Quasi view-independent human motion recognition in subspaces. In: Proceedings of 9th International Conference on Machine Learning and Computing (ICMLC) (2017)

[14] Zhu, W., Lan, C., Xing, J., Zeng, W., Li, Y., Shen, L., Xie, X.: Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. pp. 3697–3703 (2016)