Dense Bag-of-Temporal-SIFT-Words for Time Series Classification

Adeline Bailly, Simon Malinowski, Romain Tavenard, Laetitia Chapel, Thomas Guyet

Dense Bag-of-Temporal-SIFT-Words for Time Series Classification

Adeline Bailly[1,3], Simon Malinowski[2], Romain Tavenard[1], Laetitia Chapel[3], and Thomas Guyet[4]

[1] Université de Rennes 2, IRISA, LETG-Rennes COSTEL, Rennes, France
[2] Université de Rennes 1, IRISA, Rennes, France
[3] Université de Bretagne-Sud, IRISA, Vannes, France
[4] Agrocampus Ouest, IRISA, Rennes, France

Abstract. The SIFT framework has shown to be effective in the image classification context. In [4], we designed a Bag-of-Words approach based on an adaptation of this framework to time series classification. It relies on two steps: SIFT-based features are first extracted and quantized into words; histograms of occurrences of each word are then fed into a classifier. In this paper, we investigate techniques to improve the performance of Bag-of-Temporal-SIFT-Words: dense extraction of keypoints and different normalizations of Bag-of-Words histograms. Extensive experiments show that our method significantly outperforms nearly all tested standalone baseline classifiers on publicly available UCR datasets.

Keywords: time series classification, Bag-of-Words, SIFT, dense features, BoTSW, D-BoTSW.

1 Introduction

Classification of time series has received an important amount of interest over the past years due to many real-life applications, such as medicine [29], environmental modeling [11], speech recognition [16]. A wide range of algorithms have been proposed to solve this problem. One simple classifier is the k-nearest-neighbor (kNN), which is usually combined with Euclidean Distance (ED) or Dynamic Time Warping (DTW) similarity measure. The combination of the kNN classifier with DTW is one of the most popular methods since it achieves high classification accuracy [24]. However, this method has a high computation cost which makes its use difficult for large-scale real-life applications.

Above-mentioned techniques compute similarity between time series based on point-to-point comparisons. Classification techniques based on higher level structures (*e.g.* feature vectors) are most of the time faster, while being at least as accurate as DTW-based classifiers. Hence, various works have investigated the extraction of local and global features in time series. Among these works, the Bag-of-Words (BoW) approach (also called Bag-of-Features) consists in representing documents using a histogram of word occurrences. It is a very common technique in text mining, information retrieval and content-based image retrieval

because of its simplicity and performance. For these reasons, it has been adapted to time series data in some recent works [5, 6, 18, 26, 29]. Different kinds of features based on simple statistics, computed at a local scale, are used to create the words.

In the context of image retrieval and classification, scale-invariant descriptors have proved their accuracy. Particularly, the Scale-Invariant Feature Transform (SIFT) framework has led to widely used descriptors [21]. These descriptors are scale and rotation invariant while being robust to noise. In [4], we build upon this framework to design a BoW approach for time series classification where words correspond to quantized versions of local features. Features are built using the SIFT framework for both detection and description of the keypoints. This approach can be seen as an adaptation of [27], which uses SIFT features associated with visual words, to time series. In this paper, we improve on our previous work by applying enhancement techniques for BoW approaches, such as dense extraction and BoW normalization. To validate this approach, we conduct extensive experiments on a wide range of publicly available datasets.

This paper is organized as follows. Section 2 summarizes related work, Section 3 describes the proposed Bag-of-Temporal-SIFT-Words (BoTSW) method and its improved version (D-BoTSW) and Section 4 reports experimental results. Finally, Section 5 concludes and discusses future work.

2 Related work

Our approach for time series classification builds on two well-known methods in computer vision: local features are extracted from time series using a SIFT-based approach and a global representation of time series is produced using Bag-of-Words. This section first introduces state-of-the-art distance-based methods in Time Series Classification (TSC), then presents previous works that make use of Bag-of-Words approaches for TSC and finally introduces some ensemble classifiers adapted to TSC.

2.1 Distance-based time series classification

Data mining community has, for long, investigated the field of time series classification. Early works focus on the use of dedicated similarity measures to assess similarity between time series. In [24], Ratanamahatana and Keogh compare Dynamic Time Warping to Euclidean Distance when used with a simple kNN classifier. While the former benefits from its robustness to temporal distortions to achieve high accuracy, ED is known to have much lower computational cost. Cuturi [9] shows that, although DTW is well-suited to retrieval tasks since it focuses on the best possible alignment between time series, it fails at precisely quantifying dissimilarity between non-matching sequences (which is backed by the fact that DTW-derived kernel is not positive definite). Hence, he introduces the Global Alignment Kernel that takes into account all possible alignments in order to produce a reliable similarity measure to be used at the core of standard

kernel methods such as Support Vector Machines (SVM). Instead of building classification decision on similarities between time series, Ye and Keogh [31] use a decision tree in which the partitioning of time series is performed with respect to the presence (or absence) of discriminant sub-sequences (named shapelets) in the series. Though accurate, the method is very computational demanding as building the decision tree requires one to check for all candidate shapelets. Douzal and Amblard [10] define a dedicated similarity measure for time series which is then used in a classification tree.

2.2 Bag-of-Words for time series classification

Inspired by text mining, information retrieval and computer vision communities, recent works have investigated the use of Bag-of-Words (BoW) for time series classification [5, 6, 18, 25, 26, 29]. These works are based on two main operations: converting time series into BoW and building a classifier upon this BoW representation. Usually, standard techniques such as random forests, SVM or kNN are used for the classification step. Yet, many different ways of converting time series into BoW have been introduced. Among them, Baydogan *et al.* [6] propose a framework to classify time series denoted TSBF where local features such as mean, variance and extrema are computed on sliding windows. These features are then quantized into words using a codebook learned by a class probability estimate distribution. In [29], discrete wavelet coefficients are extracted on sliding windows and then quantized into words using k-means. Quantized Fourier coefficients are used as words in the BOSS representation introduced in [25]. In [18, 26], words are constructed using the Symbolic Aggregate approXimation (SAX) representation [17] of time series. SAX symbols are extracted from time series and histograms of n-grams of these symbols are computed to form a Bag-of-Patterns (BoP). In [26], Senin and Malinchik combine SAX with Vector Space Model to form the SAX-VSM method. In [5], Baydogan and Runger design a symbolic representation of multivariate time series (MTS), called SMTS, where MTS are transformed into a feature matrix, whose rows are feature vectors containing a time index, the values and the gradient of time series at this time index (on all dimensions). Random samples of this matrix are given to decision trees whose leaves are seen as words. A histogram of words is output when the different trees are learned.

Local feature extraction has been investigated for long in the computer vision community. One of the most powerful local feature for image is SIFT [21]. It consists in detecting keypoints as extremum values of the Difference-of-Gaussians (DoG) function and describing their neighborhoods using histograms of gradients. Xie and Beigi [30] use similar keypoint detection for time series. Keypoints are then described by scale-invariant features that characterize the shapes surrounding the extremum. In [8], extraction and description of time series keypoints in a SIFT-like framework is used to reduce the complexity of DTW: features are used to match anchor points from two different time series and prune the search space when searching for the optimal path for DTW.

2.3 Ensemble classifiers for time series

Recently, ensemble classifiers have been designed for time series classification. They typically rely on standalone classifiers (such as the ones described above) and fuse information from their outputs to build a more reliable classification decision. Lines and Bagnall [19] propose an ensemble classifier based on a family of elastic distance measures, named Proportional Elastic Ensemble (PROP). Bagnall *et al.* [3] propose a meta ensemble classifier that is a collection of 35 classifiers for time series (COTE).

In this paper, we build upon BoW of SIFT-based descriptors. We propose an adaptation of SIFT to mono-dimensional signals that preserves their robustness to noise and their scale invariance. We then use BoW to gather information from many local features into a single global one.

3 Bag-of-Temporal-SIFT-Words (BoTSW)

The proposed method is based on three main steps: (i) extraction of keypoints in time series, (ii) description of these keypoints through gradient magnitude at a specific scale and (iii) representation of time series by a BoW, where words correspond to quantized version of the description of keypoints. These steps are depicted in Fig. 1 and detailed below.

3.1 Keypoint extraction in time series

The first step of our method consists in extracting keypoints in time series. Two approaches are described here: the first one is based on scale-space extrema detection (as in [4]) and the second one proposes a dense extraction scheme.

Scale-space extrema detection. Following the SIFT framework, keypoints in time series can be detected as local extrema in terms of both scale and (temporal) location. These scale-space extrema are identified using a DoG function, and form a list of scale-invariant keypoints. Let $L(t, \sigma)$ be the convolution ($*$) of a time series $S(t)$ with a Gaussian function $G(t, \sigma)$ of width σ:

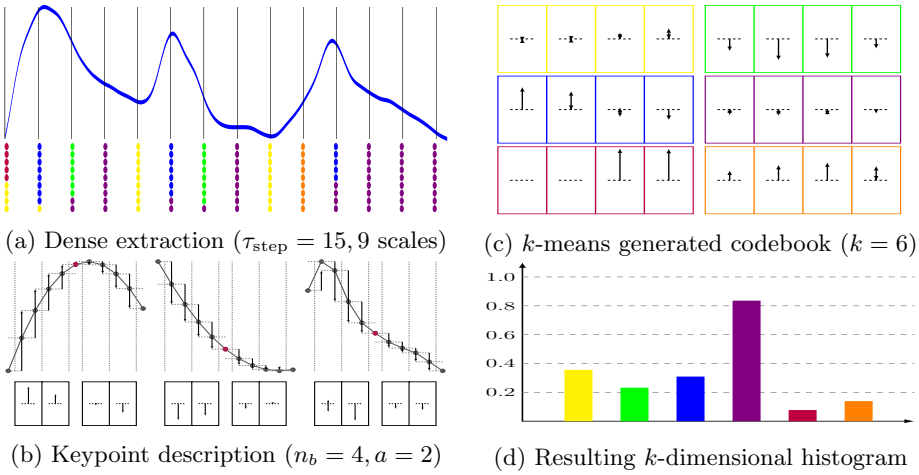
$$L(t, \sigma) = S(t) * G(t, \sigma) \tag{1}$$

where $G(t, \sigma)$ is defined as

$$G(t, \sigma) = \frac{1}{\sqrt{2\pi}\ \sigma}\ e^{-t^2/2\sigma^2}. \tag{2}$$

Lowe [20] proposes the Difference-of-Gaussians (DoG) function to detect scale-space extrema in images. Adapted to time series, a DoG function is obtained by subtracting two time series filtered at consecutive scales:

$$D(t, \sigma) = L(t, k_{\text{sc}}\sigma) - L(t, \sigma), \tag{3}$$



(a) Dense extraction ($\tau_{\text{step}} = 15, 9$ scales)

(c) k-means generated codebook ($k = 6$)

(b) Keypoint description ($n_b = 4, a = 2$)

(d) Resulting k-dimensional histogram

Fig. 1: Approach overview: (a) A time series and its dense-extracted keypoints. (b) Keypoint description is based on the time series filtered at the scale at which the keypoint is extracted. Descriptors are quantized into words. (c) Codewords obtained *via* k-means, the color is associated with the dots under each keypoint in (a). (d) Histograms of word occurrences are given to a classifier (linear SVM) that learns boundaries between classes. Best viewed in color.

where k_{sc} is a parameter of the method that controls the scale ratio between two consecutive scales.

Keypoints are then detected at time index t in scale j if they correspond to extrema of $D(t, {k_{\text{sc}}}^j \sigma_0)$ in both time and scale, where σ_0 is the width of the Gaussian corresponding to the reference scale. At a given scale, each point has two neighbors: one at the previous and one at the following time instant. Points also have neighbors one scale up and one scale down at the previous, same and next time instants, leading to a total of eight neighbors. If a point is higher (or lower) than all of its neighbors, it is considered as an extremum in the scale-space domain and hence a keypoint.

Dense extraction. Previous works have shown that accurate classification could be achieved by using densely extracted local features [14, 28]. In this section, we present the adaptation of this setup to our BoTSW scheme. Keypoints selected with dense extraction no longer correspond to extrema but are rather systematically extracted at all scales every τ_{step} time steps on Gaussian-filtered time series $L(\cdot, {k_{\text{sc}}}^j \sigma_0)$.

Unlike scale-space extrema detection, regular sampling guarantees a minimal amount of keypoints per time series. This is especially crucial for smooth time series from which very few keypoints are detected when using scale-space extrema detection. In all cases, description of these keypoints (*cf.* Section 3.2) covers the

description of scale-space extrema if τ_{step} is small enough, which leads to a more robust representation. A dense extraction scheme is represented in Fig. 1. In the following, when dense extraction is performed, we will refer to our method as D-BoTSW (for dense BoTSW).

3.2 Description of the extracted keypoints

Next step in our process is the description of keypoints. A keypoint at time index t and scale j is described by gradient magnitudes of $L(\cdot, {k_{\text{sc}}}^j \sigma_0)$ around t. To do so, n_b blocks of size a are selected around the keypoint. Gradients are computed at each point of each block and weighted using a Gaussian window of standard deviation $\frac{a \times n_b}{2}$ so that points that are farther in time from the detected keypoint have lower influence. Then, each block is described by two values: the sum of positive gradients and the sum of negative gradients. Resulting feature vector is hence of dimension $2 \times n_b$.

3.3 Bag-of-Temporal-SIFT-Words for time series classification

Training features are used to learn a codebook of k words using k-means clustering. Words represent different local behaviors in time series. Then, for a given time series, each feature vector is assigned the closest word in the codebook. The number of occurrences of each word in a time series is computed. (D-)BoTSW representation of a time series is the normalized histogram of word occurrences.

Bag-of-Words normalization. Dense sampling on multiple Gaussian-filtered time series provides considerable information to process. It also tends to generate words with little informative power, as stop words do in text mining applications. In order to reduce the impact of those words, we compare two normalization schemes for BoW: Signed Square Root normalization (SSR) and Inverse Document Frequency normalization (IDF). These normalizations are commonly used in image retrieval and classification based on histograms [12, 13, 23, 27].

Jégou *et al.* [13] and Perronin *et al.* [23] show that reducing the influence of frequent codewords before ℓ_2 normalization could be profitable. They apply a power $\alpha \in [0, 1]$ on their global representation. SSR normalization corresponds to the case where $\alpha = 0.5$, which leads to near-optimal results [13, 23]. IDF normalization also tends to lower the influence of frequent codewords. To do so, document frequency of words is computed as the number of training time series in which the word occurs. BoW are then updated by diving each component by its associated document frequency. SSR and IDF are both applied before ℓ_2 normalization. Note that normalizing histograms using SSR followed by ℓ_2 can be seen as an explicit feature map for the Hellinger kernel between ℓ_1-normalized histograms [1]. We show in the experimental part of this paper that using BoW normalization improves the accuracy of our method.

Normalized histograms finally feed a classifier that learns how to discriminate classes from this BoW representation.

Time complexity The process of computing (D-)BoTSW representation for a time series has linear time complexity in the number of features extracted. When using dense extraction, this number depends on the length of the time series. For a time series of length T, features will be computed at $\lfloor T/\tau_{\text{step}} \rfloor$ different time instants. At each time instant, features will be computed at all scales and the number of scales is $\left\lfloor \frac{\log(T/8\sigma_0)}{\log k_{\text{sc}}} \right\rfloor$. Finally, time complexity of computing D-BoTSW for a time series of length T is in $O(T \log T)$.

4 Experiments and results

In this section, we investigate the impact of both dense extraction of keypoints and normalizations of the Bag-of-Words on classification performance. We then compare our results to the ones obtained by 9 relevant baselines.

Experiments are conducted on the 86 currently available datasets from the UCR repository [15], the largest online database for time series classification. It includes a wide variety of problems, such as sensor reading (*ECG*), image outline (*ArrowHead*), human motion (*GunPoint*), as well as simulated problems (*TwoPatterns*). All datasets are split into a training and a test set, whose size varies between less than 20 and more than 8000 time series. For a given dataset, all time series have the same length, ranging from 24 to more than 2500 time instants. Following results should be interpreted in the light of the potential bias that is implied by using such a setup, especially when few training data are available [2].

For the sake of reproducibility, C++ source code used for (D-)BoTSW in these experiments and all raw numbers are made available for download[1]. To provide illustrative timings for our methods, we ran it on a personal computer, for a given set of parameters, using dataset *Cricket_X* [15] that is made of 390 training time series and 390 test ones. Each time series in the dataset is of length 300. Extraction and description of dense keypoints take around 2 seconds for all time series in the dataset. Then, 75 seconds are necessary to learn a k-means and fit a linear SVM classifier using training data only. Finally, classification of all D-BoTSW corresponding to test time series takes less than 1 second.

4.1 Experimental setup

Parameters a, n_b, k and C_{SVM} of (D-)BoTSW are learned, whereas we set $\sigma_0 = 1.6$ and $k_{\text{sc}} = 2^{1/3}$, as these values have shown to produce stable results [21]. Parameters a, n_b, k and C_{SVM} vary inside the following sets: $\{4, 8\}$, $\{4, 8, 12, 16, 20\}$, $\left\{2^i, \forall i \in \{5..10\}\right\}$ and $\{1, 10, 100\}$ respectively. A linear SVM is used to classify time series represented as (D-)BoTSW. For our approach, the best sets (in terms of accuracy) of $(a, n_b, k, C_{\text{SVM}})$ parameters are selected by performing cross-validation on the training set. Due to the heterogeneity of the datasets, leave-one-out cross-validation is performed on datasets where the

[1] http://github.com/a-bailly/dbotsw

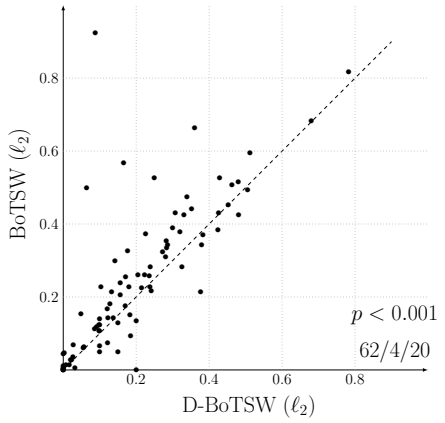


Fig. 2: Error rates of BoTSW compared to D-BoTSW.

training set contains less than 300 time series, and 10-fold cross-validation is used otherwise. These best sets of parameters are then used to build the classifier on the training set and evaluate it on the test set. For datasets with little training data, it is likely that several sets of parameters yield best performance during the cross-validation process. For example, when using *DiatomSizeReduction* dataset, BoTSW has 150 out of 180 parameter sets yielding best performance, while there are 42 such sets for D-BoTSW with SSR normalization. In both cases, the number of *best* parameter sets is too high to allow a fair parameter selection. When this happens, we keep all parameter sets with best performance at training and perform a majority voting between their outputs at test time.

Parameters a and n_b both influence the descriptions of the keypoints; their optimal values vary between sets so that the description of keypoints can fit the shape of the data. If the data contains sharp peaks, the size of the neighborhood on which features are computed (equal to $a \times n_b$) should be small. On the contrary, if it contains smooth peaks, descriptions should take more points into account. The number k of codewords needs to be large enough to precisely represent the different features. However, it needs to be small enough in order to avoid overfitting. We consequently allow a large range of values for k.

In the following, BoTSW denotes the approach where keypoints are selected as scale-space extrema and BoW histograms are ℓ_2-normalized. For all experiments with dense extraction, we set $\tau_{\text{step}} = 1$, and we extract keypoints at all scales. Using such a value for τ_{step} enables one to have a sufficient number of keypoints even for small time series, and guarantees that keypoint neighborhoods overlap so that all subparts of the time series are described.

4.2 Dense extraction *vs.* scale-space extrema detection

Fig. 2 shows a pairwise comparison of error rates between BoTSW and its dense counterpart D-BoTSW for all datasets in the UCR repository. A point on the

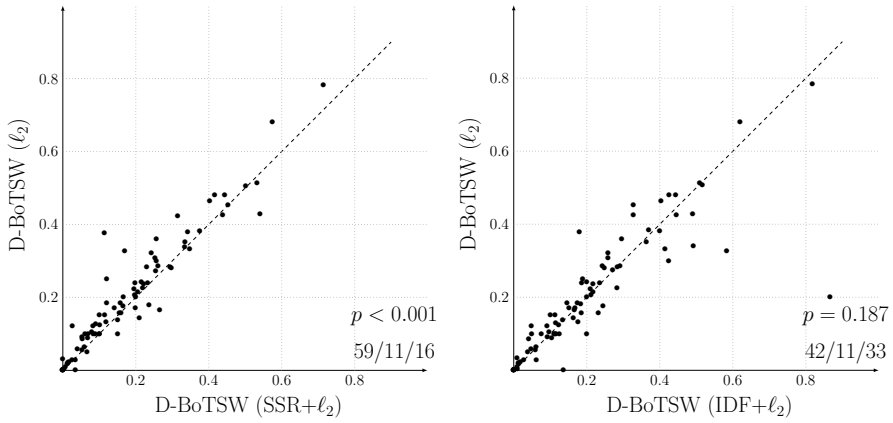


Fig. 3: Error rates of D-BoTSW with and without normalization.

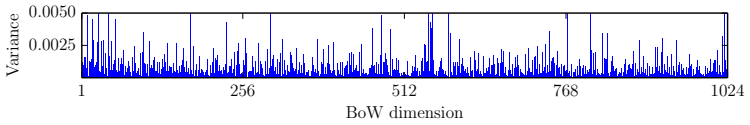
diagonal means that methods obtain equal error rates. A point above the diagonal illustrates a dataset for which D-BoTSW outperforms BoTSW. One-sided Wilcoxon signed rank test's p-value and Win/Tie/Lose scores are given in the bottom-right corner of the figure. D-BoTSW reaches better performance than BoTSW on 62 datasets, equivalent performance on 4 datasets and worse on 20 datasets. If the p-value is less than the 5% significance level, the method on the x-axis is considered significantly better than the one on the y-axis, *e.g.* Fig. 2 shows that D-BOTSW (ℓ_2) is significantly better than BoTSW (ℓ_2). D-BoTSW improves classification on a large majority of datasets. In the following, we show how to further improve these results thanks to BoW normalization.

4.3 Impact of the BoW normalization

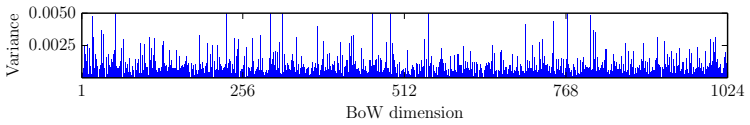
As can be seen in Fig. 3, both SSR and IDF normalizations improve classification performance (though the improvement of using IDF is not statistically significant). Lowering the influence of largely-represented codewords hence leads to more accurate classification with D-BoTSW. IDF normalization only leads to a small improvement in classification accuracy, whereas SSR normalization significantly improves classification accuracy. This is backed by Fig. 4, which shows that SSR normalization tends to spread energy across BoW dimensions, leading to a more balanced representation than other two normalization schemes.

4.4 Comparison with state-of-the-art methods

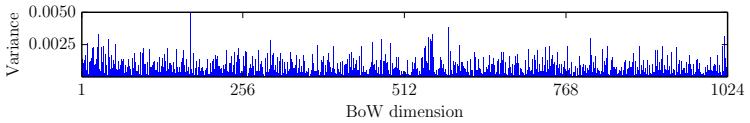
In the following, we will refer to dense SSR-normalized BoTSW as D-BoTSW, since this setup is the one providing the best classification performance. We now compare D-BoTSW to the most popular state-of-the-art methods for time series classification (a detailed comparison of time series classification algorithms can be found in [2]). The UCR repository provides error rates for the



(a) ℓ_2 normalized D-BoTSW



(b) IDF+ℓ_2 normalized D-BoTSW



(c) SSR+ℓ_2 normalized D-BoTSW

Fig. 4: Per-dimension energy of D-BoTSW vectors extracted from dataset *ShapesAll*. The same codebook is used for all normalization schemes so that dimensions are comparable across all three sub-figures.

86 datasets with Euclidean distance 1NN (EDNN) and Dynamic Time Warping 1NN (DTWNN) [24]. We use published error rates for TSBF (45 datasets) [6], SAX-VSM (51 datasets) [26], SMTS (45 datasets) [5], BoP (20 datasets) [18], BOSS (83 datasets) [25], PROP (46 datasets) [19] and COTE (45 datasets) [3].

As BoP only provides classification performance for 20 datasets, we decided not to plot pairwise comparison of error rates between D-BoTSW and BoP. Note however that the Win/Tie/Lose score is 15/2/3 in favor of D-BoTSW and this difference is statistically significant ($p < 0.001$).

Fig. 5 shows that D-BoTSW performs better than 1NN combined with ED (EDNN) or DTW (DTWNN), TSBF, SAX-VSM and SMTS. This difference is statistically significant. We can also notice from Fig. 5 that BOSS and D-BoTSW have comparable Win/Tie/Lose performance. Note that, if D-BoTSW is not significantly better than BOSS ($p = 0.791$), the reverse is also true ($p = 0.209$). It is striking to realize that D-BoTSW not only improves classification accuracy, but might improve it considerably. Error rate on *Shapelet Sim* dataset drops from 0.461 (EDNN) and 0.35 (DTWNN) to 0 (D-BoTSW), for example. Pairwise comparisons of methods show that D-BoTSW is significantly better than almost all state-of-the-art standalone classifiers, excepted BOSS that exhibits equivalent performance compared to D-BoTSW.

In Fig. 6, we compare our standalone classifier D-BoTSW to ensemble classifiers. Wilcoxon tests show that D-BoTSW is not statistically better than neither PROP nor COTE. Testing the reverse hypothesis that D-BoTSW is outperformed by these methods gives significance for COTE ($p = 0.046$) but not for PROP ($p = 0.725$). In all cases, ensemble classifiers can benefit from the design of new standalone classifiers as D-BoTSW.

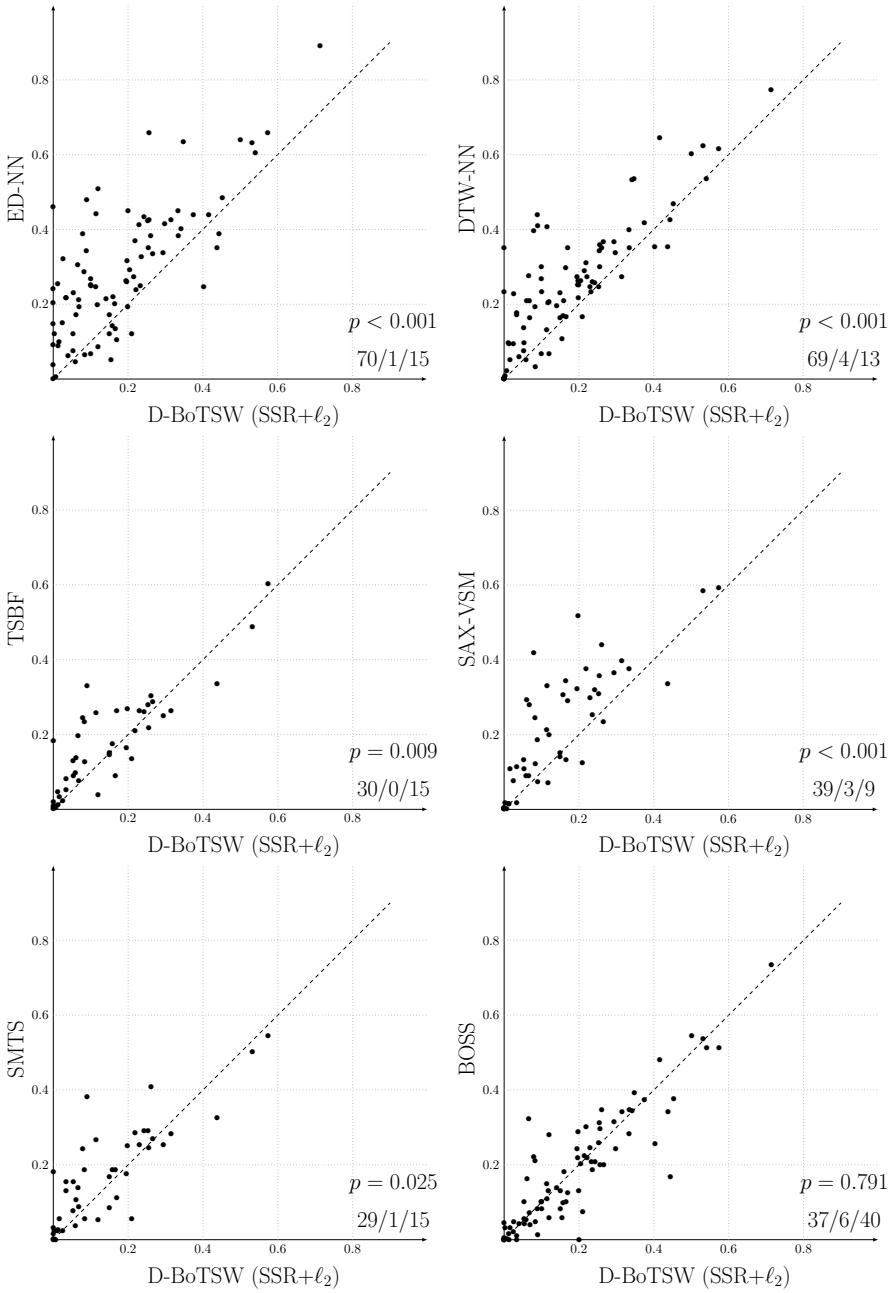


Fig. 5: Error rates for D-BoTSW with SSR normalization versus standalone baseline classifiers (ED-NN, DTW-NN, TSBF, SAX-VSM, SMTS and BOSS).

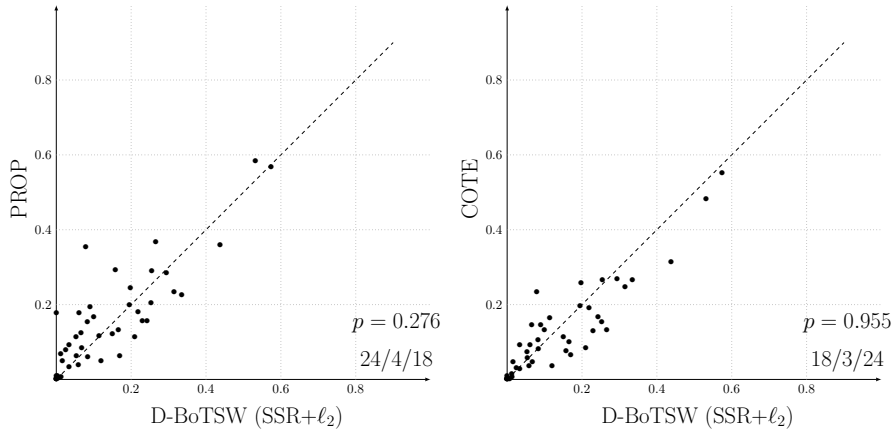


Fig. 6: Error rates for D-BoTSW with SSR normalization versus baseline ensemble classifiers (PROP and COTE).

We noticed that D-BoTSW performs especially well in the presence of large amounts of training data. On the contrary, when faced with smaller training sets, it is more likely (though still a minority) that non-parametric methods such as DTWNN or EDNN are competitive against D-BoTSW.

These experiments, conducted on a wide variety of time series datasets, show that D-BoTSW significantly outperforms most considered standalone classifiers.

5 Conclusion

In this paper, we presented the D-BoTSW technique, which transforms time series into histograms of quantized local features. The association of SIFT keypoints and Bag-of-Words has been widely used and is considered as a standard technique in image domain; however it has never been investigated for time series classification. We carried out extensive experiments and showed that dense keypoint extraction and SSR normalization of Bag-of-Words lead to the best performance for our method. We compared the results with standard techniques for time series classification and show that D-BoTSW significantly outperforms most standalone state-of-the-art time series classification algorithms.

We believe that classification performance could be further improved by taking more time information into account. Indeed, only local temporal information is embedded in our model and the global structure of time series is ignored. Moreover, more detailed global representations for feature sets than BoW have been proposed in the computer vision community [13, 22], and such global features could be used in our framework. Future work also includes the evaluation of our method on multidimensional signals: a straightforward extension would be to consider one dimension at a time and describe it, as done for HSV-SIFT [7].

Acknowledgments

This work has been partly funded by ANR project ASTERIX (ANR-13-JS02-0005-01), Région Bretagne and CNES-TOSCA project VEGIDAR. Authors also thank anonymous reviewers for their fruitful comments as well as data donators.

References

1. R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2911–2918, 2012.
2. A. Bagnall, A. Bostrom, J. Large, and J. Lines. The Great Time Series Classification Bake Off: An Experimental Evaluation of Recently Proposed Algorithms. Extended Version. *CoRR*, abs/1602.01711, 2016.
3. A. Bagnall, J. Lines, J. Hills, and A. Bostrom. Time-series classification with COTE: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535, 2015.
4. A. Bailly, S. Malinowski, R. Tavenard, T. Guyet, and L. Chapel. Bag-of-Temporal-SIFT-Words for Time Series Classification. In *Proceedings of the ECML-PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, 2015.
5. M. G. Baydogan and G. Runger. Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery*, 29(2):400–422, 2015.
6. M. G. Baydogan, G. Runger, and E. Tuv. A Bag-of-Features Framework to Classify Time Series. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2796–2802, 2013.
7. A. Bosch, A. Zisserman, and X. Muoz. Scene classification using a hybrid generative/discriminative approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(4):712–727, 2008.
8. K. S. Candan, R. Rossini, and M. L. Sapino. sDTW: Computing DTW Distances using Locally Relevant Constraints based on Salient Feature Alignments. In *Proceedings of the International Conference on Very Large DataBases*, volume 5, pages 1519–1530, 2012.
9. M. Cuturi. Fast global alignment kernels. In *Proceedings of the International Conference on Machine Learning*, pages 929–936, 2011.
10. A. Douzal-Chouakria and C. Amblard. Classification trees for time series. *Elsevier Pattern Recognition*, 45(3):1076–1091, 2012.
11. P. Dusseux, T. Corpetti, and L. Hubert-Moy. Temporal kernels for the identification of grassland management using time series of high spatial resolution satellite images. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, pages 3258–3260, 2013.
12. H. Jégou and O. Chum. Negative evidences and co-occurrences in image retrieval: the benefit of PCA and whitening. In *Proceedings of the European Conference on Computer Vision*, pages 774–787, 2012.
13. H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3304–3311, 2010.
14. F. Jurie and B. Triggs. Creating Efficient Codebooks for Visual Recognition. In *Proceedings of the International Conference on Computer Vision*, pages 604–610, 2005.

15. E. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR Time Series Classification/Clustering Homepage, 2011. `www.cs.ucr.edu/~eamonn/time_series_data/`.
16. Y. Le Cun and Y. Bengio. Convolutional networks for images, speech, and time series. In *The Handbook of Brain Theory and Neural Networks*, pages 255–258, 1995.
17. J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 2–11, 2003.
18. J. Lin, R. Khade, and Y. Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *International Journal of Information Systems*, 39:287–315, 2012.
19. J. Lines and A. Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29(3):565–592, 2014.
20. D. G. Lowe. Object Recognition from Local Scale-Invariant Features. In *Proceedings of the International Conference on Computer Vision*, pages 1150–1157, 1999.
21. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
22. F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
23. F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier. Large-Scale Image Retrieval with Compressed Fisher Vectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3384–3391, 2010.
24. C. A. Ratanamahatana and E. Keogh. Everything you know about dynamic time warping is wrong. In *Proceedings of the Workshop on Mining Temporal and Sequential Data*, pages 22–25, 2004.
25. P. Schäfer. The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6):1505–1530, 2014.
26. P. Senin and S. Malinchik. SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model. *Proceedings of the IEEE International Conference on Data Mining*, pages 1175–1180, 2013.
27. J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, pages 1470–1477, 2003.
28. H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *Proceedings of the British Machine Vision Conference*, pages 124.1–124.11, 2009.
29. J. Wang, P. Liu, M. F.H. She, S. Nahavandi, and A. Kouzani. Bag-of-Words Representation for Biomedical Time Series Classification. *Biomedical Signal Processing and Control*, 8(6):634–644, 2013.
30. J. Xie and M. Beigi. A Scale-Invariant Local Descriptor for Event Recognition in 1D Sensor Signals. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1226–1229, 2009.
31. L. Ye and E. Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 947–956, 2009.