Classifying Time Series Using Local Descriptors with Hybrid Sampling

Jiaping Zhao, *Student Member, IEEE* and Laurent Itti, *Member, IEEE*

Abstract—Time series classification (TSC) arises in many fields and has a wide range of applications. Here, we adopt the bag-of-words (BoW) framework to classify time series. Our algorithm first samples local subsequences from time series at feature-point locations when available. It then builds local descriptors, and models their distribution by Gaussian mixture models (GMM), and at last it computes a Fisher Vector (FV) to encode each time series. The encoded FV representations of time series are readily used by existing classifiers, e.g., SVM, for training and prediction. In our work, we focus on detecting better feature points and crafting better local representations, while using existing techniques to learn codebook and encode time series. Specifically, we develop an efficient and effective peak and valley detection algorithm from real-case time series data. Subsequences are sampled from these peaks and valleys, instead of sampled randomly or uniformly as was done previously. Then, two local descriptors, Histogram of Oriented Gradients (HOG-1D) and Dynamic time warping-Multidimensional scaling (DTW-MDS), are designed to represent sampled subsequences. Both descriptors complement each other, and their fused representation is shown to be more descriptive than individual ones. We test our approach extensively on 43 UCR time series datasets, and obtain significantly improved classification accuracies over existing approaches, including NNDTW and shapelet transform.

Index Terms—Time series classification, local descriptor, fisher vector, bag of words

1 Introduction

Time series classification (TSC) has numerous applications in many fields, including data mining, machine learning, signal processing, computational biology, etc. Typical classification approaches can be categorized as instance-based (e.g., one nearest neighbor classifier with euclidean distance (NN-Euclidean), or dynamic time warping distance (NNDTW)), shapelet [1], [2], [3], [4], feature-based [5], [6], and local pattern-frequency histogram based methods [6], [7], [8], [9], [10]. Instance-based methods, like NNDTW, have been successfully used for TSC and shown to be very hard to beat [11], [12], [13], but they are usually less interpretable. Shapelet is another promising method for TSC, and it discovers subsequences that are discriminative of class membership and provides more interpretable results, but searching for shapelets on large datasets becomes time-consuming or even intractable [14]. Feature-based methods do show promising classification results, but their capabilities are largely attributed to strong classifiers like SVM, adaboost and random forest, instead of being due to better global/local features and representations. Our work belongs to local pattern-frequency histogram methods, and we exploit the general Bag-of-Words (BoW) framework.

A typical BoW framework consists of three major steps: (1) local feature points detection and description, (2) codebook generation and (3) signal encoding. Afterwards, any classifier can be trained on signal encodings to do the final classification. The performance of a BoW framework implementation depends on all steps. In the computer vision community, many efforts have been made to improve each step. Regarding local feature detection and description, successful feature extractors (e.g., SIFT [15], Space Time Interest Points (STIPs) [16]) have been developed to detect local feature points, and manually-crafted descriptors (e.g., Histogram of Gradients [17], Motion Boundary Histogram (MBH) [18]) have been invented to represent local 2D image patches and 3D visual cuboid patterns around feature points. However, as reviewed below, fewer developments have been made with 1D time series descriptors. The next step, codebook generation, attempts to model the local descriptor space and to provide a partition in that space. Two typical ways are K-means and Gaussian Mixture Models (GMM). For the last step, encoding, there is a large family of research studies; several representative encoding methods are vector quantization (hard voting) [19], sparse coding [20] and Fisher Vector encoding [21]. In this work, we adopt the BoW pipeline: we focus on improving the first step: designing better local feature extractors and descriptors, while using existing techniques for the second and third steps; specifically, GMM is used to produce the codebook and Fisher Vector [21] is employed to encode the time series.

While local feature extractors are well studied in the computer vision community, in the time series community, no widely used extractors exist yet, such that most methods sample feature points either uniformly or randomly. In this paper, we introduce an efficient and effective feature point extractor, which detects all peaks and valleys, termed as landmarks, from time series. Afterwards, subsequences centered on landmarks are sampled. Landmark-based sampling gives deterministic and phase-invariant subsequences, while uniform or random sampling are affected by the phase of the time series [9]. Due to the observation that dense sampling

• *The authors are with the Department of Computer Science, University of Southern California, Los Angeles, CA 90089. E-mail: {jiapingz, itti}@usc.edu.*

Manuscript received 16 Apr. 2015; revised 14 Oct. 2015; accepted 15 Oct. 2015. Date of publication 19 Oct. 2015; date of current version 2 Feb. 2016. Recommended for acceptance by E. Keogh.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2015.2492558

outperforms sparse interest-points sampling in image classification [22] and activity recognition [23], in experiments, we adopt a hybrid sampling strategy: first sample subsequences from landmarks, then sample uniformly in "flat" featureless regions of the signal. In this way, information from both feature-rich and feature-less intervals is incorporated in the sampled subsequences. We show experimentally that this new hybrid sampling strategy outperforms both uniform and random sampling significantly.

To the best of our knowledge, little recent literature on time series classification is focused on developing better local descriptors for local time series subsequences. Commonly used local features are often simple, including mean, variance and slope [6], [8], [9]. However, statistical features like mean and variance cannot characterize local segment shapes well. Although slope incorporates shape information, it will underfit the shape of local subsequences if the interval (here, a subsequence is divided into equal-length non-overlapping temporal intervals and represented as a sequence of slopes of intervals) is too long, and becomes sensitive to noise if the interval is too short. Symbolic Aggregate approXimation (SAX) [24] is shown be a good representation for time series, however, its usage in a BoW framework [25] creates a large codebook, resulting in high-dimensional encoding vectors for time series, which inevitably enburdens downstream classifier training and prediction. Other widely used and somewhat older time series representations include Discrete Fourier Transform (DFT) coefficients, Discrete Wavelet Transform (DWT), piecewise linear approximation (PLA), etc. It is important to clarify that SAX, DFT, DWT and PLA have been used in general to represent the whole time series, instead of local subsequences.

In our work, we propose two new local descriptors, namely Histogram of Oriented Gradients (HOG) of 1D time series (HOG-1D) and Dynamic time warping-multidimensional scaling (DTW-MDS), which are shown experimentally to be quite descriptive of local subsequence shapes. These two descriptors have individual advantages: HOG-1D consists of statistical histograms, therefore is robust to noise. Moreover, HOG-1D is invariant to y-axis magnitude shift. While DTW-MDS is sensitive to noise and magnitude shift, it is more invariant to stretching, contraction and temporal shifting. Two descriptors thus complement each other. By fusing them into a single descriptor, the fused one, HOG-1D+DTW-MDS, combines the benefits of both descriptors, becomes more descriptive of subsequences, and thus is more discriminative for classification tasks. Experimental results show that our fused descriptor outperforms existing descriptors, such as DFT, DWT and Zhang's [5], [26], significantly on 43 UCR datasets for time series classification. Here DFT, DWT and Zhang's are used to represent local subsequences, instead of the whole time series. All local descriptors, including our fused one, work under the same classification pipeline: (1) feature points extraction, (2) local subsequence representation, (3) time series encoding by Fisher Vector, (4) linear kernel SVM classification. In addition, we compare TSC performance of our fused descriptor with two state-of-the-art algorithms, NNDTW and shapelet transform [2], on 41 UCR datasets, and ours achieves the best performance on 22 of them (including ties). Wilcoxon signed rank test on relative accuracy boost (see Section 4.3 for its definition) shows our fused descriptor improves relative classification accuracies significantly compared to NNDTW ($p < 0.0017$) and shapelet transform ($p < 0.0452$). Our algorithm performs well on UCR datasets, which have fixed length time series instances, however, our algorithm is also applicable to datasets with variable length time series instances, since Fisher Vector is essentially a normalized encoder, making encodings largely invariant to time series length.

Our contributions are several fold: (1) we introduce a simple but effective feature point extractor, which detects a set of landmarks from time series; (2) we explicitly design two local subsequence descriptors, namely HOG-1D and DTW-MDS, which are descriptive of local shapes and complement each other; (3) we obtain significantly improved classification accuracies using our fused descriptor when compared with two competing state-of-the-art TSC algorithms, NNDTW and shapelet transform, and three existing descriptors, DFT, DWT and Zhang's, on 43 UCR datasets. Our algorithm pipeline[1] is shown in Fig. 1.

2 Previous Work

Time series classification methods can be categorized into instance-based, shapelets, feature-based and pattern frequency histogram methods.

Instance-based methods predict labels of test time series based on their similarity to the training instances. The most popular similarity metrics include euclidean distance and elastic distances, e.g., the dynamic time warping (DTW) distance. Using a single nearest neighbor, with euclidean distance (NNEuclidean) or DTW distance (NNDTW), has demonstrated successful time series label prediction. DTW allows time series to be locally shifted, contracted and stretched, and lengths of time series hence need not be the same. Therefore, DTW usually gives a better similarity measurement than Euclidean distance, and NNDTW has been shown to be very hard to beat on many datasets [11]. A number of more complex elastic distance measures have been proposed, including longest common subsequences (LCSS) [27], Edit distance with Real Penalty (ERP) [28] and edit distance on Real Sequence (EDR) [29]. However, in [30], the authors claimed that no other elastic distance measure outperforms DTW by a statistically significant amount, and DTW is the best measure. Instance-based approaches, like NN-euclidean and NNDTW, are accurate, but they are less interpretable, since they are based on global matching and provide limited insights into the temporal characteristics.

Shapelet is a localized time series subsequence, which is discriminative of class membership, and it was first proposed and used by Ye and Keogh [1] for time series classification. The original shapelet algorithm [1] searches for shapelets recursively, and builds a decision tree using different shapelets as splitting criteria. However, the expressiveness of shapelets is limited to binary decision questions. In [4], the authors proposed logical-shapelets, specifically conjunctions or disjunctions of shapelets, which are shown to be more expressive than a single shapelet, and to

1. Code available at: https://github.com/jiapingz/TSClassification

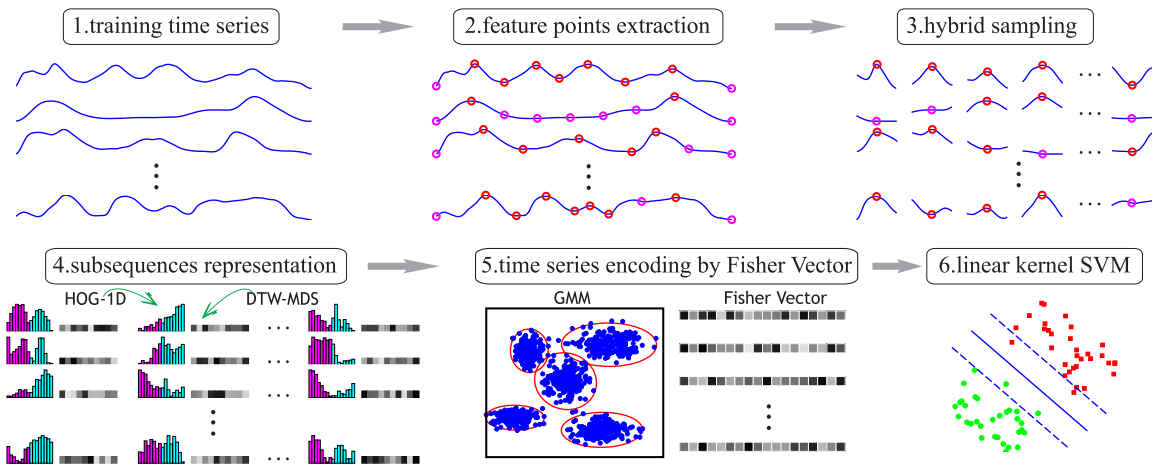

Fig. 1. Algorithm pipeline: we adopt a typical BoW framework: (1) local feature extraction and representation; (2) codebook generation; and (3) time series encoding. The encoded feature vectors are used by any classifier to do the classification. In our work, we focus on detecting better feature points and obtaining better local representations, while using existing techniques to learn codebook and encode time series. Specifically, we develop an effective algorithm to detect peaks and valleys from time series, and subsequences are sampled from these feature points (steps 2 and 3). Then, we introduce two descriptors, Histogram of Oriented Gradients (HOG-1D) and Dynamic Time Warping-Multidimensional Scaling (DTW-MDS), to represent local subsequences (step 4). Afterwards, we fit the distribution of subsequences descriptors by a K component Gaussian Mixture Model and encode each time series by a Fisher Vector (FV) (step 5). At last, linear kernel SVM is employed to classify time series based on FV encodings. The pipeline shows the process for training time series, and test time series T_t go through the same process, and the only difference is in step 5: T_t is directly encoded as a FV by learned GMM model parameters from training (without GMM model fitting any more).

experimentally outperform the original shapelet algorithm. The above two algorithms embed shapelet discovery in a decision tree, while in [2], the authors separate shapelet discovery from classifier by finding the best k shapelets in a single scan of all time series. The shapelets are used to transform the data, where each attribute in the new dataset represents the distance of a time series to one of the k shapelets. Hills et al. demonstrate that the transformed data, in conjunction with more complex classifiers, produces better accuracies than the embedded shapelet tree. Since shapelets are localized class-discriminative subsequences, shapelets-based methods have increased interpretability than global instance-based matching. The main drawback is the time complexity of searching for shapelets, and subsequent research, e.g., [14], focuses on developing efficient shapelet-searching algorithms.

Feature-based methods generally consist of two sequential steps: extract features and train a classifier based on features. Typical global features include statistical features, like variance and mean, PCA coefficients, DFT coefficients, zero-crossing rate [5]. These features are extracted either from time domain or from transformed domains, like frequency domain and principal component space. Afterwards, the extracted features either go through feature selection procedures to prune less significant ones [5], or are fed directly into complex classifiers, like multi-layer neural network [31]. Global features lose temporal information, although it is potentially informative for classification. In [8], the authors extracted features from intervals of time series, constructed and then boosted binary stumps on these interval features, and trained an SVM on outputs of the boosted binary stumps. In [6], the authors extracted simple interval features as well, including mean, variance and slope, trained a random forest ensemble classifier, and showed better performance than NNDTW. Although feature-based methods have shown promising classification results, their capabilities are largely attributed to strong classifiers such as SVM, adaboost and random forest, instead of being due to better global/local features and representations.

Another popular method is based on pattern frequency histograms, widely known as bag of words. The BoW approach incorporates word frequencies but ignores their locations. In time series applications, several recent papers adopted BoW ideas. Lin et al. [25] first symbolize time series by SAX, then slide a fixed-sized window to extract a contiguous set of SAX words, and at last use the frequency distribution of SAX words as a representation for the time series. Baydogan et al. [9] propose a similar bag-of-features framework. They sample subsequences with varying lengths randomly, use mean, variance, slope and temporal location t to represent each subsequence, afterwards utilize random forest classification to estimate class probabilities of each subsequence, and finally represent the raw time series by summarizing the subsequence class-probability distribution information. They showed superior or comparable results to competing methods like NNDTW on UCR datasets [4]. Wang et al. [10] adopted a typical bag of words framework to classify biomedical time series data, and they sample subsequences uniformly and represent them by DWT. Grabocka and Schmidt-Thieme [32] introduce a similar BoW pipeline to classify time series: they sample subsequences from time series instances uniformly, learn latent patterns and membership assignments of each subsequence to those patterns, and sum up membership assignments of subsequences on a time series as the representation of that time series. Time series representations are then classified by polynomial kernel SVM. Our work belongs to this category, but emphasizes detecting better local feature points and developing better local subsequence representations.

There are two recent papers using local descriptors as well [33], [34]. In [33], the authors attempt to improve efficiency of traditional DTW computation, to be concrete, they extract local feature points, match them by their descriptors and compute the local band constraints (based on matched

pairs) applicable during the execution of the DTW algorithm. In this way, they only have to compute the accumulative distances within the band, and the DTW computation efficiency is improved. Our work is different from [33] in that: our use local descriptors to improve classification accuracies, while [33] use local descriptors to improve DTW computation efficiency. In [34], the authors extract local features from multi-variate time series by leveraging metadata, and their method for local feature extraction is only applicable for multi-variate time series data with known correlations and dependencies among different dimensions. UCR datasets are univariate time series datasets, and their method cannot be used here.

3 Methodology

3.1 Algorithm Overview

We follow the classic bag-of-words pipeline closely to do the classification. As mentioned, we focus on developing better local feature extractors and better local subsequence representations, while using existing algorithms for the following steps, specifically, GMM is used to generate the codebook and Fisher Vector [21] is employed to encode the time series. We propose a simple but effective feature point extractor, which robustly detects peaks and valleys from real case time series, and local subsequences are sampled from these feature points instead of sampled randomly or uniformly as previously done. Afterwards, we introduce two descriptors, namely HOG-1D and DTW-MDS, to represent the local subsequences. Two descriptors have individual advantages and complement each other, and both capture local shapes well. Since each subsequence has two descriptors, we can either use them separately, or fuse them as a single descriptor. In case of fusion, two descriptors are first ℓ_2-normalized and then concatenated to form a new descriptor, i.e., $d^i = [d^i_{HOG-1D} / \parallel d^i_{HOG-1D} \parallel_2 \, d^i_{DTW-MDS} / \parallel d^i_{DTW-MDS} \parallel_2]$, where d^i_{HOG-1D} and $d^i_{DTW-MDS}$ are HOG-1D and DTW-MDS descriptors of the i^{th} subsequence, and d^i is its concatenated new descriptor: termed as HOG-1D+DTW-MDS.

To do TSC, we first extract subsequences and represent them by either (1) HOG-1D, (2) DTW-MDS, or (3) HOG-1D + DTW-MDS descriptors, then learn a generative K component GMM to model the distribution of local descriptors based on training data, and at last encode each time series by a Fisher Vector [21]. Subsequently, an SVM classifier with linear kernel is used for training and testing based on the Fisher Vector representations of time series. The details of local feature extraction and representation, codebook generation and global time series encoding are given in the following sections.

3.2 Feature Points Detector

To the best of our knowledge, existing literature uses regular constant-step sliding window sampling (a.k.a., uniform sampling) or random sampling strategies to extract subsequences from long time series [9], [10], [25]. In the case of uniform sampling, a fixed-size window is slid along the temporal axis with a constant stride and subsequence within each sliding window is sampled. In case of random sampling, subsequences are extracted from random locations on time series. There is inevitable randomness in both sampling strategies, specifically different start sampling points make sampled subsequences different under the uniform sampling case, while randomness under random sampling is inherent. This partially motivates us to design a procedure to make sampled subsequences deterministic. Concretely we propose to extract temporal feature points first, and then sample subsequences from there, if feature points are deterministic, sampled subsequences are fixed each time.

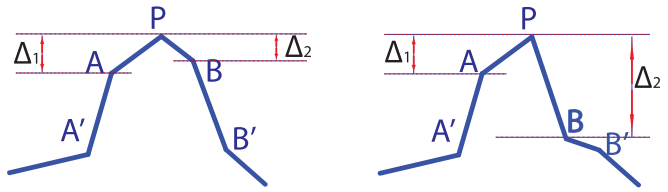

Fig. 2. Local maxima: **Algo1** will miss some true local optima. p on two segments are both local peaks, suppose Δ_1 in both cases are the same, but Δ_2 on the left segment is much smaller than Δ_2 on the right segment, then p on the left segment is harder to satisfy the peak condition in **Algo1**, indicating p on the left segment may be missed.

Compared with non-feature points, local feature points are more descriptive of local shapes, and more robust to noise. Successful local feature point detectors include the 2D image feature point detector SIFT [15] and 3D spatio-temporal video feature point detector STIPs [16]. SIFT and STIPs are widely used for object recognition in 2D images and activity recognition in videos. Inspired by their great performance, we introduce a 1D temporal feature point detector, aiming at reaching downstream higher TSC accuracies. The feature point detection makes following steps invariant, to some extent, to time series phases.

We define temporal feature points to be peaks and valleys on time series. Given a time series: $T = (t_1, t_2, \ldots, t_n)$, a peak or valley t_i on a noise-free time series satisfies: $(t_i - t_{i-1}) \cdot (t_{i+1} - t_i) < 0$, i.e., its left and right derivatives change signs. However, this simple criterion for feature point detection fails to work for real case signals, since there are many small bumps on signals, and many false positives are detected as a result. Since valley detection in raw time series T can be transformed to peak detection in $-T$, we will focus on peak detection in raw time series T.

If our algorithm returns t_i as a peak when it satisfies $t_i > t_{i-1}$ and $t_i > t_{i+1}$, many points found in this way lie on ascending or descending slopes, which are false peaks. The challenge is to find 'better' local maxima while discarding 'fake' ones. To be more selective of peaks, a straightforward modification is: only if $t_i > t_{i-1} + \Delta$ and $t_i > t_{i+1} + \Delta$, where $\Delta > 0$, then t_i is returned as a local peak. The larger Δ is, the more selective the algorithm is and the fewer peaks are returned. In this way, on the contrary, lots of true positives will be missed when setting Δ to be large. We can relax the constraints: t_i is returned as a peak if It is larger than one neighbor by a gap Δ, we name this algorithm as **Algo1**:

Algo1: t_i is returned as a peak when it satisfies either (1) $t_i > t_{i-1} + \Delta$ and $t_i > t_{i+1}$ or (2) $t_i > t_{i-1}$ and $t_i > t_{i+1} + \Delta$.

However, **Algo1** will inevitably miss some true positives. For example, in Fig. 2, p on two segments are both local peaks, suppose Δ_1 in two cases are the same, but Δ_2 on the left segment is much smaller than Δ_2 on the right segment, then p on the left segment is harder to satisfy the peak condition in **Algo1**, indicating p on the left segment may not be

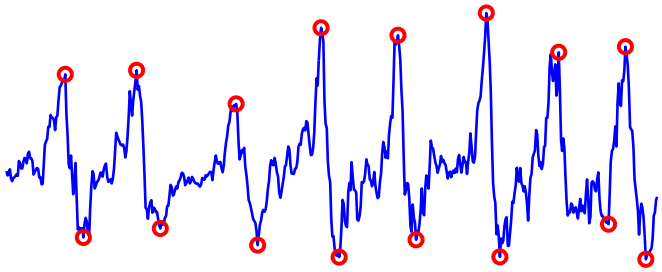

Fig. 3. Feature Points Extraction by **Algo3**: a real-case time series and peak and valley points (red circles) returned by running **Algo3**. Visually, all the true positives are extracted while the false positives are suppressed.

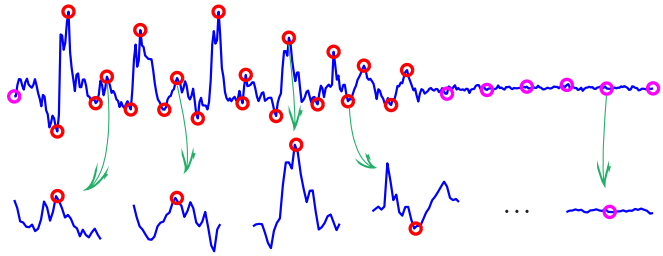

Fig. 4. Hybrid Sampling: in feature-rich regions, sample from feature points; in flat regions, sample uniformly. Red circles are detected feature points, and magenta circles are evenly-spaced points in flat regions. Subsequences centered around both point types are sampled. The second row shows examples of sampled subsequences.

returned as peak. We introduce **Algo2**, which is guaranteed to find both peaks found by **Algo1** and some true positives missed by **Algo1**.

Algo2: Given a real case time series data $T = (t_1, t_2, \ldots, t_n)$, only keep temporal points whose left and right derivatives change sign, i.e., $(t_i - t_{i-1}) \cdot (t_{i+1} - t_i) < 0$ and organize them in the same temporal order as they are in the raw time series to form a trimmed time series as $T' = (t'_1, t'_2, \ldots, t'_m)$. Use T'as input, return t'_i as a peak when t'_i satisfies either (1) $t'_i > t'_{i-1} + \Delta$ and $t'_i > t'_{i+1}$ or (2) $t'_i > t'_{i-1}$ and $t'_i > t'_{i+1} + \Delta$. Compared with **Algo1**, there is a trimming step in **Algo2**, and the following peak extraction process remains the same.

Algo2 is guaranteed to detect peak points detected by Algo1. Proof: let $(\ldots, A', A, p, B, B', \ldots)$ be part of raw time series T as shown in Fig. 2 (the right segment), and assume p is returned by **Algo1** as a peak, then p must satisfy either (1) $p > A + \Delta$ and $p > B$; or (2) $p > A$ and $p > B + \Delta$. In either case, $(p - A) \cdot (B - p) < 0$, i.e., p is a left/right derivative sign changing point and will be kept in the trimmed time series T'. We will show p will be returned by **Algo2** as a peak as well. Case one: if both A and B are sign changing points, then segment $-A - p - B-$ is also a segment in the trimmed time series T'. Then p satisfies either (1) $p > A + \Delta$ and $p > B$; or (2) $p > A$ and $p > B + \Delta$ as well, and returned by **Algo2** as a peak. Case two: if only one of A and B is a sign changing point, without loss of generality, assume A be the sign changing point and kept in T' to be left neighbor of p. Assume the new right neighbor of p is B'', then B'' is a sign changing point on the right side of B in T. If $B'' \geq B$, then there must be some point $\widehat{B}$ in the raw time times T between B and B'', satisfying $\widehat{B} < B$ and $\widehat{B} < B''$, otherwise B will be a sign changing point. In this case $\widehat{B}$ is a sign changing point and would be the right neighbor of p, contradicting the assumption. Therefore $B'' \leq B$. In this case, p in the trimmed time series T' will as well satisfy the criterion defined in **Algo1** and be returned. Case three: both A and B are non-sign changing points, since this can be reduced to case two, and we omit its proof.

Algo2 will return more peak points than Algo1. From the above analysis, the peak condition is easier to be satisfied in the trimmed time series, resulting in more returned peak points. A concrete example is: under some magnitude constraint of $\Delta, \Delta_1, \Delta_2$, p on the left segment of Fig. 2 will be missed by **Algo1**, but will be detected by **Algo2** easily.

In practice, we use a slightly modified version of **Algo2** to detect peaks, and name the new algorithm as **Algo3**: after obtaining a trimmed time series $T' = (t'_1, t'_2, \ldots, t'_m)$ as in **Algo2**, return t'_i as a peak when t'_i satisfies either (1) $t'_i > min(t'_{<i}) + \Delta$ and $t'_i > t'_{i+1}$ or (2) $t'_i > t'_{i-1}$ and $t'_i > min(t'_{>i}) + \Delta$. Note in **Algo2**, t'_i is compared with its immediate left and right neighbors t'_{i-1} and t'_{i+1}, while in **Algo3**, t'_i is compared to its immediate right (left) neighbor t'_{i+1} (t'_{i-1}) and some neighbor $min(t'_{<i})$ ($min(t'_{>i})$) from its left (right) side. In the constraint $min(t'_{<i})$ ($min(t'_{>i})$) denotes some point residing between t'_i and its left (right) closest peak, and having the minimum value. Compared with **Algo2**, **Algo3** is more robust to noise, and easily removes many 'false peaks' on ascending or descending slopes (see feature points detection results in Fig. 3). The only parameter in **Algo3** is Δ, which is set to be some ratio of the value range of the time series instance, i.e., $\Delta = \lambda \cdot (\max(T) - \min(T))$, where $0 < \lambda < 1$ (see supplementary materials, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2015.2492558, for details and run our demo code).

Fig. 3 shows a typical example of feature points detection by **Algo3** from a noisy time series. As we see, **Algo3** achieves both high precision and recall of feature points, i.e., detects most visually true peaks/valleys while suppressing false positives. Afterwards, subsequences of fixed length centered on these landmarks will be extracted. However, in practice, we exploit a hybrid sampling strategy: in feature-rich regions, we sample subsequences from landmarks while in "flat" regions, we sample uniformly. In this way, information from both feature-rich and feature-less intervals is preserved, and time series are well characterized by the hybrid-sampled subsequences. Hybrid sampling is illustrated in Fig. 4. In the following two sections, we introduce two descriptors, HOG-1D and DTW-MDS, to represent sampled subsequences.

3.3 HOG-1D Descriptor

Histogram of oriented gradients was first introduced by Dalal and Triggs [17] for object detection. It is shown that local object appearances and shapes are well captured by the distribution of local intensity gradients. Its excellent performance for pedestrian detection was empirically demonstrated. Later on, Klaser and Marszalek [35] generalized the key HOG concept to 3D spatio-temporal video domain, and developed histograms of oriented 3D spatio-temporal gradients (HOG-3D) descriptor. They applied HOG-3D descriptor to several action datasets, and obtained the state-of-the-art activity recognition results at that time.

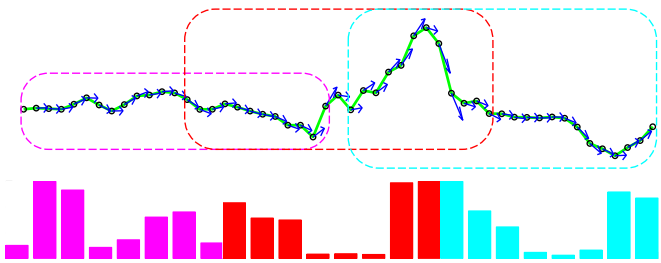

Fig. 5. HOG-1D descriptor: a subsequence s is shown as green line. At each temporal point p_i on s, centered gradient is estimated, with the blue arrow indicating its direction and magnitude. The subsequence is divided into three overlapping intervals, boxed by magenta, red and cyan rectangles. In each interval, a histogram of oriented gradients (HOG) is accumulated over all temporal points in that interval, and shown under that interval. Concatenation of three HOGs gives the HOG-1D descriptor for the subsequence s. Gradient orientations lie within $(-90, 90 \text{ degree})$, and in this figure eight evenly spaced orientation bins are used, resulting in a 24D HOG-1D descriptor.

Based on the success of HOG and HOG-3D, we here introduce a new HOG-1D descriptor for 1D time series data. We inherit key concepts from HOG, and adapt them to 1D temporal data. Assume a subsequence $s = (p_1,\ p_2, \ldots, p_l)$ of length l, divided into n constant-length overlapping or non-overlapping intervals $I = \{I_1, I_2, \ldots, I_n\}$, where the cardinality of each interval is $|\, I_i \,| = c$. Within each interval I_i, a 1D histogram of oriented gradients is accumulated over all temporal points in I_i. Concatenating n interval histograms forms the descriptor of the subsequence s, and we term it the HOG-1D descriptor. Fig. 5 shows the generation process of HOG-1D descriptor of a sample subsequence. The statistical nature of histograms makes HOG-1D less sensitive to observation noise, while the concatenation of sequential histograms captures temporal information well. When the number of intervals decreases to 1 (i.e., $n = 1$), the descriptiveness of HOG-1D is weakened since temporal information is lost, while when n increases, the cardinality of each interval will decrease (suppose n intervals do not overlap), and the HOG of a short interval will become more sensitive to noise. In practice, setting $n = 2, 3$ works well (the influence of the number of intervals on classification performances is analyzed in supplementary materials, available online). In the following, we give implementation details of HOG computation within each interval.

Given an interval $I = (p_{t_1}, p_{t_1+1}, \ldots, p_{t_2})$ with time span (t_1, t_2), the magnitude of gradient at temporal point t $(t_1 \leq t \leq t_2)$ is calculated as $|g_t| = |\sigma \cdot (p_{t+1} - p_{t-1})/((t+1) - (t-1))| = |\sigma \cdot \frac{1}{2}(p_{t+1} - p_{t-1})|$, and its orientation is $\arctan(g_t)$, which lies within $(-90°, 90°)$. σ is a global scaling factor, accounting for different time series sampling frequencies. Specifically, for time series sampled at a high frequency, adjacent observations p_{t-1} and p_t will almost be the same, making $|g_t| \approx 0$ and $\arctan(g_t) \approx 0°$ everywhere. Under this scenario, HOG over interval I will be a spiked distribution, making HOG-1D unable to distinguish different local shapes. In practice, σ is set such that gradient orientations distribute within $(-90, 90 \text{ degree})$ approximately evenly (see the algorithm in the supplementary material, available online, to search for α). After gradient computation at each temporal point, the next step is to accumulate gradient votes within orientation bins, and obtain HOG over bins. The orientation bins are evenly spaced within $(-90, 90 \text{ degree})$. Typically, a gradient votes for its two neighboring bins, and votes are determined bi-linearly in terms of angular distances from the gradient orientation to the bin centers. In our experiments, we exploit a kernel smoothed voting strategy, i.e., a gradient votes for all orientation bins, and the voting magnitude for the ith bin b_i is determined as $|g_t| \cdot \exp\{-\frac{1}{2}(\arctan(g_t) - \angle(b_i))^2/\hat{\sigma}^2\}$, where $\angle(b_i)$ is the orientation of bin b_i, $\hat{\sigma}$ is a scale factor indicating the decaying rate of Gaussian smoothing kernel. HOG-1D is insensitive to noise and and invariant to y-axis magnitude shift; however, since magnitude information sometimes benefits TSC, we introduced another subsequence descriptor, DTW-MDS, which complements HOG-1D to account for magnitude shift, as well contraction and stretching distortion.

3.4 DTW-MDS Descriptor

There are dozens of similarity measures for time series, and the most straightforward measure is the euclidean distance; however it does not handle distortion and misalignment in time. DTW is another ubiquitous measure, which accounts for nonlinear distortions in the temporal dimension. Dozens of alternative elastic distance measures have been invented, but experimental tests on forty datasets suggested none of them consistently beats DTW[30]. In this paper, we choose DTW as distance measure between time series subsequences.

Let $s_p = (p_1, p_2, \ldots, p_m)$ and $s_q = (q_1, q_2, \ldots, q_n)$ denote two subsequences of length m and n. Sometimes, a warping window size $\triangle$ is further enforced to make temporal indices of matched points p_i and q_j satisfy $|i - j| \leq \triangle$. However, in our experiments, we use DTW without the warping widow size constraint. Assume there are N subsequences $\{s_1, s_2, \ldots, s_N\}$, by computing pairwise subsequence DTW distances, we get a $N \times N$ symmetric DTW distance matrix d_{DTW}. In order to use d_{DTW} in the kernel based classifier like SVM, several attempts have been made to derive kernels from d_{DTW}, examples include Gaussian dynamic time warping (GDTW) kernel function $\mathcal{K}(s_i, s_j) = \exp\{-\gamma d_{DTW}(s_i, s_j)\}$ or negative dynamic time warping (NDTW) kernel function $\mathcal{K}(s_i, s_j) = -d_{DTW}(s_i, s_j)$. However, kernel matrices constructed by these functions are not positive semi-definite (PSD), thus, efficacy of SVM cannot be enjoyed by these kernels. Empirical results in [36] showed that SVM with either GDTW or NDTW kernel has inferior performance in TSC compared with NNDTW. Some attempts have been made to construct a PSD kernel matrix from a distance measure between two time series [37]. The authors considered the similarity scores spanned by all possible alignments, instead of just the score of the best alignment, and derived a kernel out of this formulation. This kernel is shown to be PSD and can be used in the kernel-based classifiers. However, construction of a valid PSD kernel from some dissimilarity matrix is unfortunately a non-trivial task. To work around the difficulty of fixing DTW kernel matrix to be PSD and at the same time to enjoy the superiority of the DTW distance measure and the strong performance of SVM, we introduce the DTW-MDS descriptor for subsequences. Briefly, we use the multidimensional scaling (MDS) algorithm to find a layout of N subsequences in space Ω ($\Omega \in R^h$) such that pairwise subsequence DTW distances d_{DTW} are preserved as well as possible in Ω. Then each subsequence is

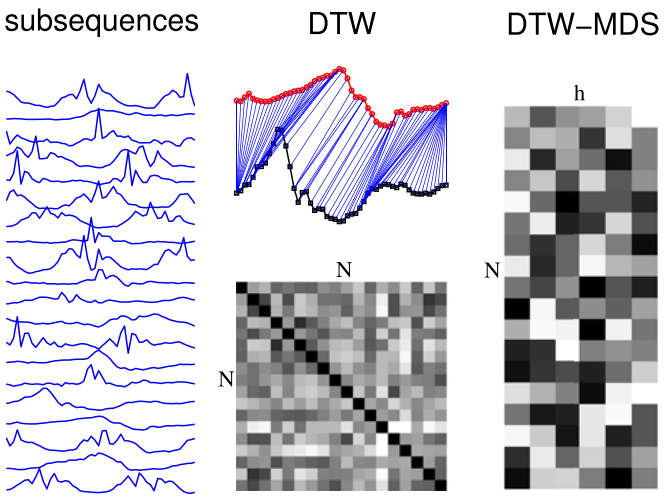

Fig. 6. DTW-MDS descriptor: the first column shows N sampled subsequences, the second column shows a $N \times N$ symmetric dynamic time warping distance matrix d_{DTW}, with each entry d_{ij} indicating the DTW distance between subsequence i and j, and the third column shows: by applying MDS on d_{DTW}, we obtain N vectors in h dimensional space, with each being the DTW-MDS descriptor of one subsequence.

represented by a vector x ($x \in R^h$) in Ω, and x is termed as the DTW-MDS descriptor.

Given an $N \times N$ DTW distance matrix d_{DTW} computed between N subsequences $\{s_1, s_2, \ldots, s_N\}$, MDS aims to find a h dimensional descriptor $x_i (x_i \in R^h, i = 1, 2, \ldots, N)$ for each subsequence, such that for each pair of subsequences s_i and s_j, their euclidean distance $\| x_i - x_j \|_2$ in the h dimensional space satisfies their DTW distance $d_{DTW}(s_i, s_j)$ as well as possible. Mathematically, we define representation error for each pairwise DTW distance as $e_{ij} = \| x_i - x_j \|_2 - d_{DTW}(s_i, s_j)$, then MDS minimizes the following normalized stress:

$$\mathbf{X} = \arg \min_{\{x_1, x_2, \ldots, x_N\}} \frac{1}{\mathcal{Z}} \sum_{1 \le i < j \le N} e_{ij}^2, \tag{1}$$

where $\mathcal{Z}$ is a normalization factor. In practice, we set $\mathcal{Z}$ to the sum of squares of pairwise DTW distances, i.e., $\mathcal{Z} = \sum_{1 \le i < j \le N} d_{DTW}^2(s_i, s_j)$.

The solution $\mathbf{X}$ of problem (1) is an $N \times h$ matrix, with ith row x_i being the descriptor for subsequence s_i. Problem (1) can be solved in a coordinate descent way: first, determine an updating order of rows, typically either in random order, or sequentially (e.g., $1 \to 2 \to \cdots \to N \to 1 \to 2 \to \cdots$). Then update each row x_i by keeping all other rows $x_j (j \neq i)$ fixed; in this case, problem (1) becomes a quadratic programming of x_i, which is convex and x_i can be solved by the standard Levenberg-Marquardt algorithm. After N iterations, all rows x_i $(i = 1, 2, \ldots, N)$ are updated once. We repeat this process and terminate either when reaching the maximum number of iterations, or when $\triangle x_i \preceq \varepsilon$. Fig. 6 shows the process of computing DTW-MDS descriptors.

In the above way, each subsequence s_i from training time series is coded by an h dimensional vector x_i. However, we do not see subsequences of test time series during training. Here, we can use the same framework to encode each test subsequence. Given a test subsequence $\widehat{s}$, we compute its DTW distance $d_{DTW}(\widehat{s}, s_i)$ to each training subsequence s_i ($s_i \in S_{train}$), and DTW-MDS descriptor $\widehat{x}$ of $\widehat{s}$ can be obtained by solving the following minimization program:

$$\arg \min_{\widehat{x}} \sum_{s_i \in S_{train}} \{ \| \widehat{x} - x_i \|_2 - d_{DTW}(\widehat{s}, s_i) \}^2. \tag{2}$$

This least-square problem can be solved by Levenberg-Marquardt algorithm as well. In experiments, we set the maximum number of iterations to 50, and use this as the termination criterion for both Program 1 and 2.

Solving Program 1 and 2 becomes time and space consuming when the number of training subsequences N goes too large, while in practice, we develop an approximate algorithm to compute DTW-MDS descriptors, in which both space consumption and time cost to compute the DTW-MDS descriptor of a subsequence are independent of N. The key of our approximation algorithm is to choose R representative subsequences from N training subsequences, and first encode R representatives by solving Program 1, then encode each of the left $(N - R)$ training subsequences and test subsequences by solving Program 2 (see supplementary materials, available online, for algorithm details).

After transforming pairwise subsequence DTW distances to descriptors, existing valid kernel functions like RBF and polynomials are ready to be used in the kernel machine classifiers (e.g., SVM). Another advantage of DTW-MDS descriptor is that it can be fused with other features extracted from the same subsequence. When these features and DTW-MDS descriptors are complementary to each other, the merged feature vector has the potential to further boost the classification accuracy. In our experiments, we show by fusing DTW-MDS with HOG-1D, better classification accuracies on most datasets are obtained.

3.5 Time Series Encoding

After feature point detection, local subsequence extraction and representation, each time series $\hat{T}$ is represented by a set of sampled subsequences $\{\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_{\hat{N}}\}$, and each subsequence is described by a descriptor vector (either HOG-1D, DTW-MDS, or HOG-1D+DTW-MDS) $\hat{y}_i$ ($\hat{y}_i \in \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{\hat{N}}\}$). Time series encoding aims at figuring out a vector representation for $\hat{T}$, based on its local subsequence representations $\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{\hat{N}}\}$. Typical encoding methods include hard voting-based (e.g., BoW), reconstruction-based (e.g., LLC [38]) and super vector-based (e.g., Fisher Vector [21]) approaches. As experimentally demonstrated in [39], super vector based encoding is more effective than the other two encodings for action recognition. In our experiments, we choose to use Fisher Vector encoding [21] for time series.

The Fisher Vector encoding [21] is derived from Fisher Kernel, which constructs kernels from probabilistic generative models, in this way, we apply generative models in a discriminative setting and take benefits of both models. The construction of the Fisher vector starts by learning a Gaussian mixture model model from a set of local descriptors $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$. The probability density function of a GMM with K components is given by:

$$p(x; \theta) = \sum_{k=1}^{K} \pi_k \cdot N(x; \mu_k, \Sigma_k), \tag{3}$$

where $\theta = (\pi_1, \mu_1, \Sigma_1, \ldots, \pi_K, \mu_K, \Sigma_K)$ is the vector of parameters in this model, including the component prior probability π_k, the mean $\mu_k (\in R^D)$ and the positive definite covariance matrix $\Sigma_k (\in R^{D \times D})$ of each Gaussian components. For simplicity, covariance matrices are assumed to be diagonal, therefore the GMM is fully specified by $(2D+1) \times K$ scalar parameters. Given a set of $\mathbf{N}$ local descriptors $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$, the parameters in GMM are learned by maximal likelihood estimation using Expectation Maximization algorithm.

After fitting a GMM model, given a set of descriptors $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{\hat{N}}\}$ sampled from some time series $\hat{T}$, let γ_k^n be the soft assignment of descriptor $\hat{y}_n$ to Gaussian component k ($k \in \{1, 2, \ldots, K\}$), define vectors $G_{\mu,k}^{\hat{Y}}$ and $G_{\sigma,k}^{\hat{Y}}$ as:

$$\begin{aligned} G_{\mu,k}^{\hat{Y}} &= \frac{1}{\hat{N} \cdot \sqrt{\pi_k}} \sum_{n=1}^{\hat{N}} \gamma_k^n \left(\frac{\hat{y}_n - \mu_k}{\sigma_k}\right), \\ G_{\sigma,k}^{\hat{Y}} &= \frac{1}{\hat{N} \cdot \sqrt{2\pi_k}} \sum_{n=1}^{\hat{N}} \gamma_k^n \left[\left(\frac{\hat{y}_n - \mu_k}{\sigma_k}\right)^2 - 1\right]. \end{aligned} \tag{4}$$

Each of which is D dimensional. The fisher vector of the set of local descriptors $\hat{Y}$ is then given by the concatenation of $G_{\mu,k}^{\hat{Y}}$ and $G_{\sigma,k}^{\hat{Y}}$ for all K Gaussian components, resulting in a $2DK$ vector:

$$FV^{\hat{Y}} = \left[G_{\mu,1}^{\hat{Y}}, G_{\sigma,1}^{\hat{Y}}, \ldots, G_{\mu,K}^{\hat{Y}}, G_{\sigma,K}^{\hat{Y}}\right], \tag{5}$$

which is the Fisher Vector of time series $\hat{T}$. In our experiments, the local descriptors are either (1) HOG-1D; (2) DTW-MDS; or (3) HOG-1D + DTW-MDS. GMM is first learned to fit the distribution of local descriptors from training time series, then each training time series can be encoded by Fisher Vector as in Eq. (5). Given a test time series, after subsequence sampling and representation, it is represented by a set of descriptors $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{\hat{N}}\}$, and following Eq. (4) and Eq. (5), the test time series can be encoded by a Fisher Vector as well.

Power normalization. As shown in [21], as the number of Gaussian components K increases, the Fisher Vector becomes sparser. Dot products are poor measures of similarity on sparse vectors. Therefore, they proposed to power-normalize each dimension of the raw Fisher Vector by the same power factor α:

$$FV_i^N = \text{sign}(FV_i) \cdot |FV_i|^\alpha, \quad i \in \{1, 2, \ldots, 2DK\}. \tag{6}$$

In experiments, the power factor α is determined by cross-validation on training data. Until now, each time series T has a normalized encoding FV_T^N ($FV_T^N = [FV_1^N, FV_2^N, \ldots, FV_{2DK}^N]$), together with its label, we train a linear kernel SVM to do the classification.

3.6 Computational Complexity Analysis

Our classification pipeline is composed of sequential steps, and its time complexity is a sum of time costs at individual steps. Define notations as follows: let L be the length of a time series, N_{train} be the number of training time series instances, l and n_{train} be the length and number of sampled training subsequences, and r_{train} be the number of representative training subsequences (see its definition in Section 3.4). Our pipeline consists of feature point extraction, HOG-1D computation, DTW-MDS computation, FV encoding and linear SVM classification, whose time complexities during training are: $\mathcal{O}(N_{train}L)$, $\mathcal{O}(n_{train}l)$, $\mathcal{O}(r_{train}^2 l^2) + \mathcal{O}(n_{train}h^3)$, $\mathcal{O}(|d|n_{train} + Kn_{train}^2)$ and $\mathcal{O}(N_{train})$, where $|d|$ is the dimensionality of the subsequence descriptor. Therefore, the total time cost is: $\mathcal{O}(N_{train}L) + \mathcal{O}(n_{train}l) + \mathcal{O}(r_{train}^2 l^2) + \mathcal{O}(n_{train}h^3) + \mathcal{O}(|d|n_{train} + Kn_{train}^2)$. Since in general $n_{train} \gg N_{train}\, n_{train} \gg r_{train}$ and $l, h, |d|$ are usually small positive values (e.g., $l = 40, h = 20, |d| \leq 36$ in our experiments), the training time cost is quadratic in n_{train}, i.e., $\mathcal{O}(n_{train}^2)$. In practice n_{train} is usually large, training is done offline.

At test, to classify a time series, let n_{test} be the number of extracted subsequences from that time series, then subsequence extraction, HOG-1D, DTW-MDS computation, FV encoding and SVM classification have complexities $\mathcal{O}(L)$, $\mathcal{O}(n_{test}l)$, $\mathcal{O}(n_{test}r_{train}l^2) + \mathcal{O}(n_{test}h^3)$, $\mathcal{O}(n_{test})$ and $\mathcal{O}(1)$ respectively, and the overall test complexity is $\mathcal{O}(n_{test}r_{train}l^2) + \mathcal{O}(n_{test}h^3) + \mathcal{O}(L)$, where $n_{test} \ll L$ and l, h, r_{train} are small positive values as well, therefore, the test can be done online.

NNDTW has a time cost $\mathcal{O}(N_{train}L^2)$ in general; $ShapeletTransform$ [2] contains time-consuming shapelet searching, and in general it takes time $\mathcal{O}(N_{train}^2 L^3)$. $ShapeletTransform$ is usually more time-consuming than our algorithm and NNDTW. Although in general, our algorithm has higher time cost than NNDTW, our time cost at test is usually much cheaper than that of NNDTW. As the number of training time series increases, efficiency gain of our algorithm at test is further enlarged, since our test time cost is independent of N_{train}.

4 Experimental Validation

We extensively test our feature point extractor and subsequence descriptors on 43 UCR time series datasets [40]. All results reported throughout the experiments are obtained under the following fixed settings: (1) subsequence length (l) is set to be 40 ($l = 40$) on all datasets; (2) stride of sliding window in the case of uniform sampling is set to be 5 ($s = 5$); (2) in computing HOG-1D descriptors, each subsequence is divided into 2 non-overlapping segments, and the gradient orientation bins are evenly spaced over $(-90\ 90\ \text{degree})$ with the bin number set as 8. Therefore, the dimensionality of HOG-1D is 16; (3) in computing DTW-MDS descriptors, DTW distance is calculated without the warping window size constraint, and the dimensionality h of DTW-MDS is set to be 20 ($h = 20$); (4) we use the Bag-of-Words framework to encode time series, specifically using GMM clustering to generate codebook and Fisher Vector to encode time series; (5) we use a linear-kernel SVM classifier for classification. Two tunable parameters are the number of components (K) in GMM and the power normalization factor α, and different datasets have different optimal parameters. In experiments, K and α on each dataset are set by cross-validation on training data.

Baseline descriptors. We compare with two widely known time series representations, Discrete Fourier Transform and Discrete Wavelet Transform, and another representation

introduced in [5], [26] (Zhang's). Although all three representations are proposed to represent the whole time series, here we use them to represent time series subsequences. (1) **DFT**: we keep the first half of coefficients, and use them as the representation for the subsequence. (2) **DWT**: we use the Haar wavelet basis and decompose each subsequence into three levels. The detail wavelet coefficients of three levels and the approximation wavelet coefficients of the third level are concatenated to form the final representation. (3) **Zhang's**: Zhang and Sawchuk [5] augments conventional statistical features with a set of physical features, the combination of statistical and physical features forms the representation, for details of the feature sets, please refer to their paper.

Baseline TSC methods. We compare with two TSC methods, **NNDTW** and **shapelet transform** [2]. NNDTW is shown to be very hard to beat [11], and, in experiments, we use one nearest neighbor under DTW distance with the warping window size constraint. Classifiers built on shapelet transform [2] are shown to be more accurate than the original tree-based shapelet classifier [1] on a wide range of datasets. Both NNDTW and shapelet transform are top ranked TSC methods, and therefore, they are used as baseline for comparison.

4.1 Hybrid Sampling versus Uniform and Random Sampling

We compare three different sampling strategies: uniform, random and the proposed hybrid sampling. We keep the classification pipeline (local subsequence extraction and representation + FV time series encoding + linear SVM classification) fixed, but only change the subsequence sampling methods. To test whether the superiority of hybrid sampling is independent of local descriptors, we test six descriptors, including HOG-1D, DTW-MDS, HOG-1D+DTW-MDS, DWT, DFT and Zhang's.

Uniform sampling. The stride s of the sliding window is set be 5 ($s = 5$), making contiguous windows overlap by half. For each time series, we randomly choose a start sampling point from $(l/2l)$ (l is the length of subsequence, $l = 40$ in experiments) and then slide the window at stride s to get the following subsequences. Since sampled subsequences of uniform sampling depend on the start sampling point, we repeat experiments for 10 times, and the mean classification accuracy is reported.

Random sampling. To make the number of sampled subsequences from some time series the same as in uniform sampling, $\lfloor \frac{L}{s} \rfloor$ (where L is the length of the time series and s denotes the stride of sliding window) subsequences are randomly sampled from each time series. Similarly we repeat experiments 10 times, and the mean classification accuracy is reported.

Hybrid sampling. In this case, subsequences are obtained by (1) sampling from landmark points and (2) uniformly sampling from flat regions with stride 5 ($s = 5$). Sampling from landmark points results in deterministic subsequences, while uniform sampling introduces some randomness because of the start sampling point. Since in all 43 UCR datasets, flat regions take only minimal proportions of a time series, we do the uniform sampling at flat regions only once and ignore the impact of randomness. The reported accuracy is based on one experiment.

The performance of different sampling strategies under the same descriptor on 43 datasets is shown in Fig. 7. The hybrid sampling almost consistently outperforms both uniform and random sampling across different descriptors and different datasets, and in most cases, the performance gap is quite huge. Quantitatively, we perform a Wilcoxon signed rank test between performance of hybrid and uniform (random) sampling, and p-values are listed in Table 1. As seen, hybrid sampling works significantly better than both uniform and random sampling ($p \ll 0.01$). This is partially attributed to phase-invariance of landmark-points based sampling, i.e., sampling from landmark points makes the sampled subsequences independent of phases of time series. This is also due to the fact that subsequences at feature points represent the time series better than those from non-feature points. Sampling from landmarking points results in deterministic subsequences, decreasing randomness to 0, while uniform and random sampling contain much randomness, which results from time series phase shift and sampling initializations.

Since hybrid sampling is shown to be superior to both uniform and random sampling, all experiments in the following sections use hybrid sampling to get subsequences.

4.2 Complementarity of HOG-1D and DTW-MDS

We qualitatively show the descriptiveness of local shapes of HOG-1D and DTW-MDS and quantitatively report the classification accuracies of HOG-1D, DTW-MDS and HOG-1D +DTW-MDS on 43 UCR datasets.

First we visualize HOG-1D (16D) and DTW-MDS (20D) descriptors by t-SNE [41]. t-SNE displays high-dimensional data by giving each data point a location in a 2D or 3D map and is shown to reveal the manifold structures of high-dimensional datasets. 2D (or 3D) visualization of t-SNE preserves neighborhood relationship among high-dimensional points, therefore, the mapped 2D (or 3D) points of similar descriptors have similar coordinates and are grouped and displayed proximately, while descriptors from different manifolds have dissimilar dimension-reduced 2D (or 3D) coordinates and are displayed in different groups. The 1first row in Fig. 8 shows 2D t-SNE visualization of HOG-1D and DTW-MDS descriptors. The plot is generated in four steps: (1) sample subsequences from time series by hybrid sampling; (2) calculate their descriptors (HOG-1D or DTW-MDS); (3) use t-SNE to map descriptors to 2D locations; (4) plot corresponding raw subsequences on their 2D locations. As we see, subsequences of visually similar shapes are displayed proximately, while dissimilar subsequences are spatially separated. This indicates that visually similarly-shaped subsequences have similar descriptors (HOG-1D and DTW-MDS), while differently-shaped subsequences possess different descriptors. t-SNE visualization shows qualitatively that both HOG-1D and DTW-MDS capture local subsequence shapes quite well.

Then we quantitatively show classification performances of three descriptors. Two plots in the second row in Fig. 8 show classification performances between HOG-1D+DTW-MDS and HOG-1D (DTW-MDS). Most points lie above the diagonal, showing better performance of the fused descriptor than either HOG-1D or DTW-MDS. By running a Wilcoxon signed-rank test between the classification accuracies of HOG-1D+DTW-MDS and HOG-1D (DTW-MDS), we

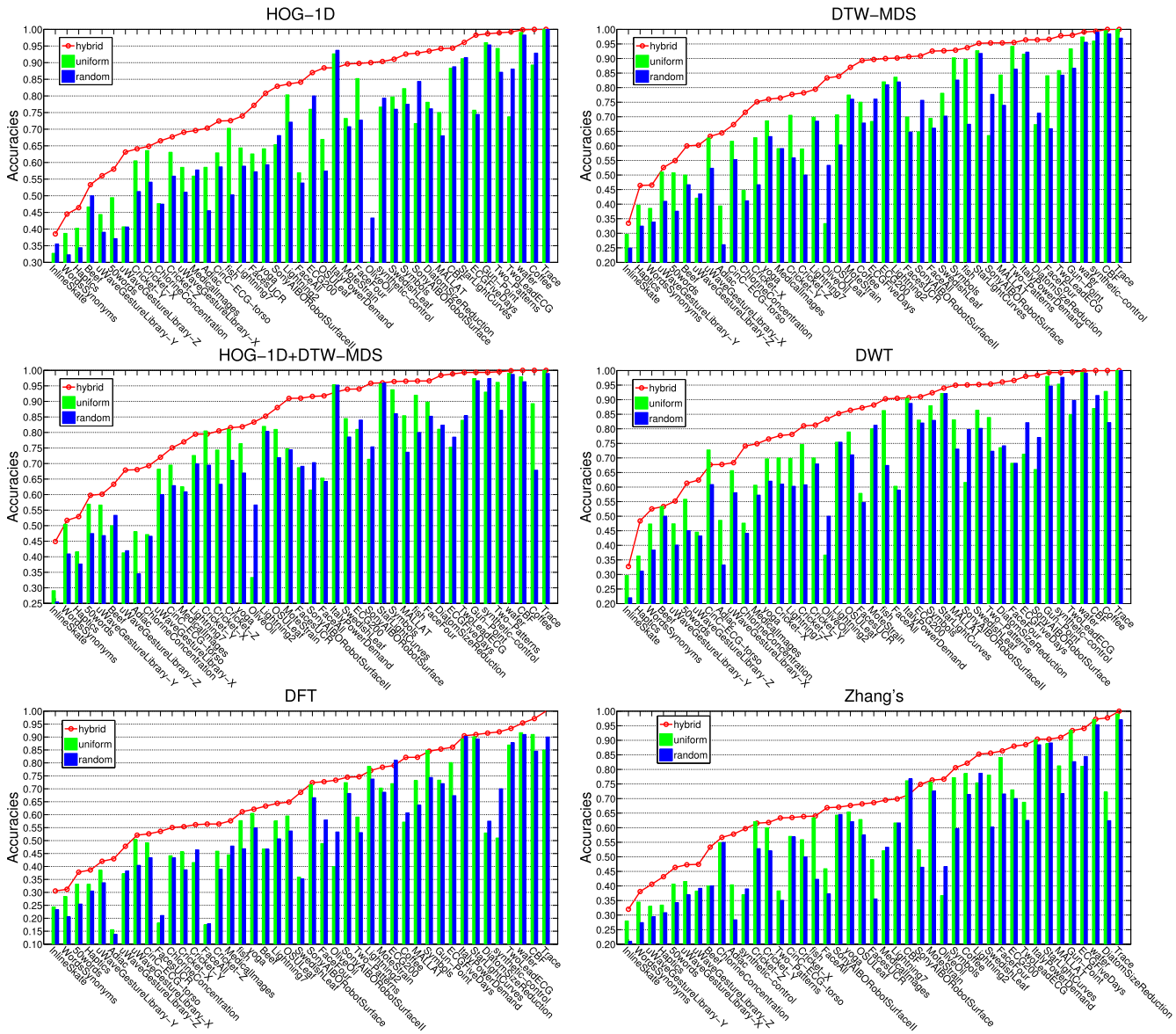

Fig. 7. Performance comparison of hybrid, uniform and random sampling strategies, across different descriptors and different datasets. We test six different descriptors, including HOG-1D, DTW-MDS, HOG-1D+DTW-MDS, DWT, DFT, and Zhang's, on 43 UCR time series datasets under three different sampling methods. Performance accuracies of different descriptors are obtained under BoW encoding and linear kernel SVM pipelines. As visually seen in the plot, hybrid sampling works almost consistently better than uniform and random sampling by a large gap, both across different descriptors and across different datasets. Quantitatively, we run Wilcoxon signed rank test between performance of hybrid and uniform (random) sampling, p-values are listed in Table1, indicating hybrid sampling outperforms both uniform and random sampling significantly ($p < 0.01$).

get p-values $6.7 \cdot 10^{-7}(1.5 \cdot 10^{-5})$, which shows the fused descriptor outperformed both HOG-1D and DTW-MDS significantly ($p \ll 0.01$). The classification error rates of three descriptors are listed in Table 3. The better performance demonstrates that the fused descriptor takes advantages of individual descriptors, becomes more descriptive of

TABLE 1
Wilcoxon Signed Rank Test between Performances of Hybrid and Uniform (Random) Sampling

Hybrid versus Uniform						
descriptors	HOG-1D	DTW-MDS	HOG-1D+DTW-MDS	Zhang's	DFT	DWT
p-values(×1.0e-7)	.2933	.1648	.3632	.6840	.1834	.3631
Hybrid versus Random						
descriptors	HOG-1D	DTW-MDS	HOG-1D+DTW-MDS	Zhang's	DFT	DWT
p-values(×1.0e-7)	.2365	.1120	.1834	.1967	.1385	.1648

Under the Wilcoxon signed rank test, the null hypothesis is that performance differences between hybrid and uniform (random) sampling come from a distribution whose median is 0 at 1 percent significance level. Since all p-values are much smaller than 0.01, showing that the null hypothesis is rejected strongly. This indicates hybrid sampling outperforms both uniform and random sampling in a statistically significant way.

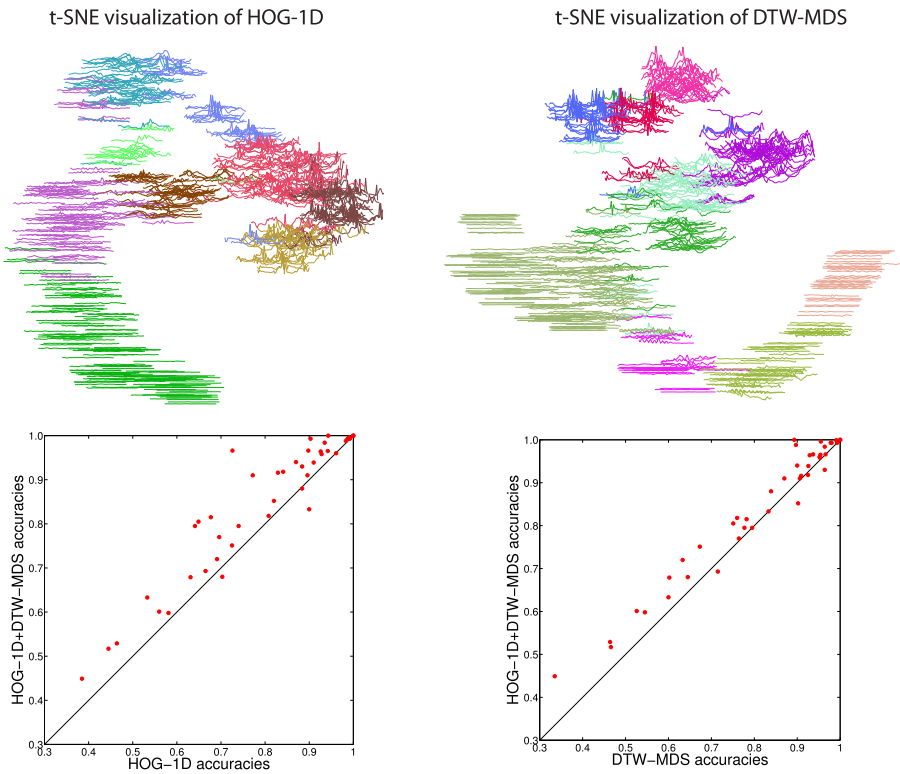

Fig. 8. Complementarity of HOG-1D and DTW-MDS descriptors. HOG-1D is invariant to *y*-axis shift and insensitive to noise, while DTW-MDS is more contraction, stretching and translation invariant. The fusion of two descriptors benefits from their individual advantages, and outperforms each separate descriptor on most of the UCR datasets. Two plots in the first row show t-SNE [41] visualization of HOG-1D and DTW-MDS descriptors of 246 subsequences. As we see from both plots, similar shaped subsequences are displayed proximately, indicating both descriptors capture shapes very well (Here, different subsequence colors are independent of t-SNE, but just for visual comfortability. 246 subsequences are k-means clustered into 10 clusters, with a random color assigned for each cluster). Two plots in the second row show performance comparisons between HOG-1D +DTW-MDS and HOG-1D (DTW-MDS). On most datasets, the fused descriptor outperforms each separate one. By running Wilcoxon signed rank test, p-value between the fused and HOG-1D (DTW-MDS) is $6.7 \cdot 10^{-7}$ and $1.5 \cdot 10^{-5}$, showing fused descriptor performs significantly better.

subsequences, and thus is more discriminative for classification tasks. In fact, HOG-1D and DTW-MDS are complementary to each other: HOG-1D is more robust to noise, and invariant to *y*-axis magnitude shift; while DTW-MDS is sensitive to noise and magnitude shift, but it is more stretching, contraction and temporal shifting invariant.

4.3 Comparison with Other Descriptors and the State-of-the-Art Algorithms

We compare our fused descriptor HOG-1D+DTW-MDS with three other subsequence descriptors: DFT, DWT and Zhang's [5], [26]. Additionally, we will compare with two state-of-the-art TSC algorithms: NNDTW and shapelet transform [2].

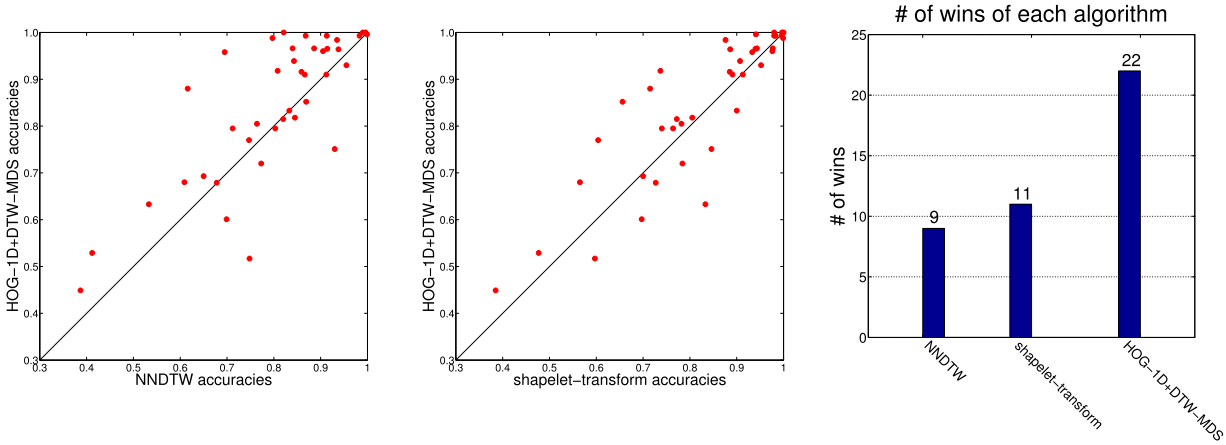

Fig. 9. Comparison with the state of the art TSC algorithms. We compare with two state-of-the-art algorithms: NNDTW (with warping window constraints) and shapelet transform [2]. The performance of our fused descriptor HOG-1D+DTW-MDS is obtained under BoW encoding and linear kernel SVM classification. The first two plots show performance comparisons between HOG-1D+DTW-MDS and NNDTW (Shapelet Transform), and as being observed, HOG-1D+DTW-MDS performs better on most datasets. The 3rd plot shows the number of datasets on which each algorithm wins (including ties) the other two, and ours gets the best performance on 22 out of 41. A Wilcoxon signed rank test shows that our method improves relative classification accuracies significantly compared to NNDTW ($p < 0.0017$) and shapelet transform ($p < 0.0452$).

TABLE 2
Wilcoxon Signed Rank Test on Relative Accuracy Boost by HOG-1D+DTW-MDS

signrank test on relative accuracy boost by HOG-1D+DTW-MDS					
methods	DFT	Zhang's	NNDTW	shapelet transform	DWT
p-values	5.2e-8	5.2e-8	0.0017	0.0452	6.2e-4

Under the significance level 5 percent, all five algorithms have significant relative accuracy boosts ($p < 0.05$).

Four local descriptors. As mentioned, we use the Bag-of-Words classification pipeline, hybrid sampling + local subsequence representation + Fisher Vector time series encoding + linear SVM classification, for 4 different local subsequence descriptors. Two tunable parameters, K in GMM and the power normalization factor α, are determined by cross-validation on training data, and classification performance on test data is reported.

Two state-of-the-art algorihtms. (1) NNDTW: we use DTW with warping window constraints as the distance measure and the label of the nearest neighbor as the predicted label for the test time series. Instead of running NNDTW again, we import results from UCR time series website [40]; (2) Shapelet Transform[2]: in the original paper, authors only tested on 29 UCR datasets, while they provided full results on their website [42]. Again, we directly import results from their website.

Beforehand, we define a terminology: **Relative accuracy boost**. It is defined to be the ratio between the boosted accuracy by HOG-1D+DTW-MDS and the original accuracy, i.e., $rA^i_\beta = (A^i_{HOG-1D+DTW-MDS} - A^i_\beta)/A^i_\beta$, where rA^i_β is the relative accuracy boost of algorithm β on dataset i ($\beta \in$ {DFT, Zhang's, NNDTW, shapelet transform, DWT}),

TABLE 3
Error Rates of Different Algorithms on UCR Time Series Datasets

datasets	HOG-1D	DTW-MDS	DFT	Zhang's	NNDTW	Shapelet Transform	DWT	HOG-1D+ DTW-MDS
50words	0.419	0.455	0.622	0.536	0.242	–	0.386	0.402
Adiac	0.297	0.355	0.570	0.422	0.391	0.435	0.322	**0.320**
Beef	0.467	0.400	0.367	0.467	0.467	**0.167**	0.467	0.367
CBF	0.057	0.000	0.029	0.287	0.004	0.003	**0.000**	**0.000**
ChlorineConcentration	0.335	0.285	0.449	0.433	0.350	0.300	**0.259**	0.307
CinC-ECG-torso	0.275	0.327	0.474	0.365	**0.070**	0.154	0.323	0.249
Coffee	0.000	0.107	0.179	0.179	0.179	**0.000**	**0.000**	**0.000**
Cricket-X	0.351	0.249	0.510	0.362	0.236	0.218	0.223	**0.195**
Cricket-Y	0.359	0.223	0.438	0.382	0.197	0.236	**0.187**	0.205
Cricket-Z	0.323	0.218	0.436	0.385	**0.180**	0.228	0.190	0.185
DiatomSizeReduction	0.065	0.036	0.085	0.023	0.065	0.124	0.039	**0.016**
ECG200	0.130	0.100	0.210	0.120	0.120	–	0.090	0.060
ECGFiveDays	0.017	0.103	0.139	0.059	0.203	**0.001**	0.020	0.012
FaceAll	0.159	0.075	0.436	0.331	0.192	0.263	0.095	**0.082**
FaceFour	0.102	0.034	0.273	0.136	0.114	0.057	**0.034**	**0.034**
FacesUCR	0.228	0.094	0.464	0.314	0.088	**0.087**	0.128	0.090
Gun-Point	0.013	0.020	0.147	0.067	0.087	0.020	**0.007**	**0.007**
Haptics	0.536	0.536	0.614	0.568	0.588	0.523	0.516	**0.471**
InlineSkate	0.615	0.665	0.695	0.680	0.613	0.615	0.673	**0.551**
ItalyPowerDemand	0.116	0.036	0.095	0.096	**0.045**	0.048	0.093	0.070
Lightning2	0.180	0.098	0.230	0.148	**0.131**	0.344	0.180	0.148
Lightning7	0.260	0.205	0.438	0.301	0.288	0.260	0.219	**0.205**
MALLAT	0.058	0.046	0.178	0.090	0.086	0.060	0.050	**0.035**
MedicalImages	0.304	0.236	0.424	0.305	0.253	0.396	0.251	**0.230**
MoteStrain	0.104	0.130	0.216	0.236	0.134	0.109	0.118	**0.090**
OSULeaf	0.116	0.161	0.351	0.318	0.384	0.285	0.136	**0.120**
OliveOil	0.100	0.167	0.267	0.233	0.167	**0.100**	0.167	0.167
SonyAIBORobotSurface	0.072	0.047	0.276	0.251	0.305	0.067	**0.017**	0.042
SonyAIBORobotSurfaceII	0.171	0.091	0.255	0.329	0.141	0.115	**0.049**	0.084
StarLightCurves	0.039	0.048	0.091	0.096	0.095	**0.024**	0.060	0.040
SwedishLeaf	0.090	0.074	0.366	0.184	0.157	0.093	**0.048**	0.061
Symbols	0.074	0.071	0.155	0.231	0.062	0.114	0.076	**0.036**
Trace	0.000	0.000	**0.000**	**0.000**	0.010	0.020	**0.000**	**0.000**
Two-Patterns	0.010	0.045	0.253	0.367	**0.001**	0.059	0.046	0.004
TwoLeadECG	0.008	0.022	0.067	0.115	0.132	**0.004**	0.005	0.007
WordsSynonyms	0.555	0.534	0.688	0.619	**0.252**	0.403	0.475	0.483
fish	0.274	0.063	0.389	0.360	0.160	**0.023**	0.097	0.034
synthetic-control	0.097	0.007	0.080	0.403	0.017	0.017	**0.007**	**0.007**
uWaveGestureLibrary-X	0.309	0.367	0.479	0.527	0.227	**0.216**	0.316	0.280
uWaveGestureLibrary-Y	0.440	0.474	0.580	0.594	**0.301**	0.303	0.448	0.399
uWaveGestureLibrary-Z	0.369	0.398	0.522	0.526	0.322	**0.273**	0.376	0.321
wafer	0.001	0.009	0.046	0.027	0.005	0.002	0.001	**0.001**
yoga	0.192	0.240	0.379	0.324	**0.155**	0.195	0.235	0.182

The lowest error rate on each dataset is highlighted in bold font (HOG-1D and DTW-MDS are excluded for performance comparison).

A^i_β and $A^i_{HOG-1D+DTW-MDS}$ are the accuracy of algorithm β and reference algorithm HOG-1D+DTW-MDS on dataset i.

Since shapelet transform [42] does not provide classification accuracies on two datasets—'50words' and 'ECG200', all the comparisons (including plots in Fig. 9 and p-values in Table 2) in this section are reported on the remaining 41 datasets. The left two plots in Fig. 9 show performance comparisons between HOG-1D+DTW-MDS and NNDTW (Shapelet Transform), and most points still lie above the diagonal, showing better performances of our algorithm. The right plot shows the number of datasets on which each algorithm wins (including ties) the other two. As shown, on 22 out of 41 datasets, HOG-1D+DTW-MDS wins both NNDTW and Shapelet Transform. By running a Wilcoxon signed rank test on the relative accuracy boost of NNDTW (shapelet transform), we obtain p-values 0.0017 (0.0452), which indicates the relative accuracy boost by HOG-1D+DTW-MDS is significant ($p < 0.05$).

We do Wilcoxon signed rank test on relative accuracy boost of five algorithms, and report the p-values in Table 2. Under the significance level 5 percent, the relative accuracy boost by HOG-1D+DTW-MDS is significant for all five algorithms.

Texas Sharpshooter Fallacy. although our algorithm outperforms other algorithms according to the Wilcoxon signed rank test, knowing this is not useful unless we can tell in advance on which problems it will be more accurate, as stated in [11]. In this section we use the Texas sharpshooter plot [11] to show when our algorithm has superior performance on the test set as predicted from performance on the training set, compared with NNDTW. We run leave-one-out cross validation on training data to measure the accuracies of HOG-1D+DTW-MDS and NNDTW, and we calculate the expected gain: accuracy(HOG-1D+DTW-MDS)/accuracy(NNDTW). We then measure the actual accuracy gain using the test data. The results are plotted in Fig. 10. Most points (88.4 percent) fall in the TP and TN regions, indicating we can confidently predict that our algorithm will be superior/inferior to NNDTW. There are only five points falling in the FP region, but as seen they just represent minor losses or gains.

For consistency with convention in reporting error rates in TSC problems, we document error rates of eight algorithms in Table 3. The lowest error rate on each dataset is highlighted in bold font (HOG-1D and DTW-MDS are excluded from performance comparison).

4.4 Empirical Time Complexity

A practical and important issue is the efficiency of the proposed algorithm. As analyzed theoretically, in general our algorithm has a higher time complexity than NNDTW. However, the majority of our time cost is during training (usually not a problem as it is done offline), while during test, our algorithm is usually much more efficient than NNDTW, especially when the training data is large. We compare empirical running time of three different algorithms, including ours, NNDTW and NN-DTW-DDTW, on 43 UCR datasets. Here, NN-DTW-DDTW is the nearest neighbor classifier under the fused distance metric, i.e., $(DTW, \beta \cdot DDTW)$, where DDTW is derivative Dynamic time warping [43] and β is a weighting factor between two metrics, tuned by

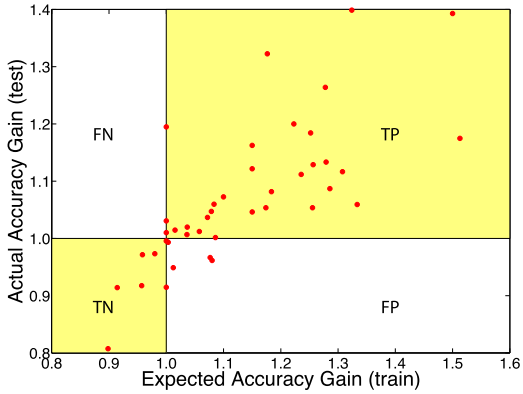

Fig. 10. Texas Sharpshooter plot: HOG-1D+DTW-MDS versus NNDTW. TP: true positive (our algorithm was expected from the training data to outperform NNDTW, and it actually did on the test data). TN: true negatives, FP: false positives, FN: false negatives. Each dot is one UCR dataset (43 dots total). Our algorithm performed on the test set as predicted from the training performance in nearly 90 percent of the datasets (38 out of 43 dots are in the TP or TN regions).

cross-validation on training data. We report running time using the following machine: Ubuntu 12.04 64-bit, Intel i7 CPU 960, eight cores, 12G RAM, matlab2015a. As seen in Fig. 11, although in general our algorithm has higher time cost (including training and test) than NNDTW and NN-DTW-DDTW, we are more efficient during test, which is important for real-time prediction.

4.5 Sensitivity Analysis

The above results are reported under a fixed parameter setting, however, the performance robustness to different parameter settings is important for an algorithm. In the supplementary materials, available online, we evaluate the performance sensitivity of our pipeline to each parameter, and do the evaluation under the philosophy that: vary one parameter at a time while maintaining other parameters fixed. Extensive experiments show that our algorithm performs well under wide ranges of subsequence lengths l, strides s and DTW-MDS dimensionalities h, and therefore is largely insensitive to them. We demonstrate as well that (1) dense sampling with a small stride outperforms sparse sampling with a large stride; (2) hybrid sampling outperforms both uniform and random sampling; (3) Fisher Vector encoding is superior to the ordinary Bag-of-words frequency encoding. See supplementary materials, available online, for detailed results.

5 Conclusions

In this work, we focus on developing better local feature extractors and better local subsequence descriptors of time series data. We introduced a simple but effective feature point detector from real case time series data, proposed to sample subsequences both from feature points and flat regions, and experimentally showed this hybrid sampling method performed significantly better than traditional uniform and random sampling. Further more, two novel descriptors, HOG-1D and DTW-MDS, are developed to represent local subsequences. Experimental results show that both descriptors are quite descriptive of local subsequence shapes and complementary of each other, and the fused descriptor outperformed individual descriptors significantly.

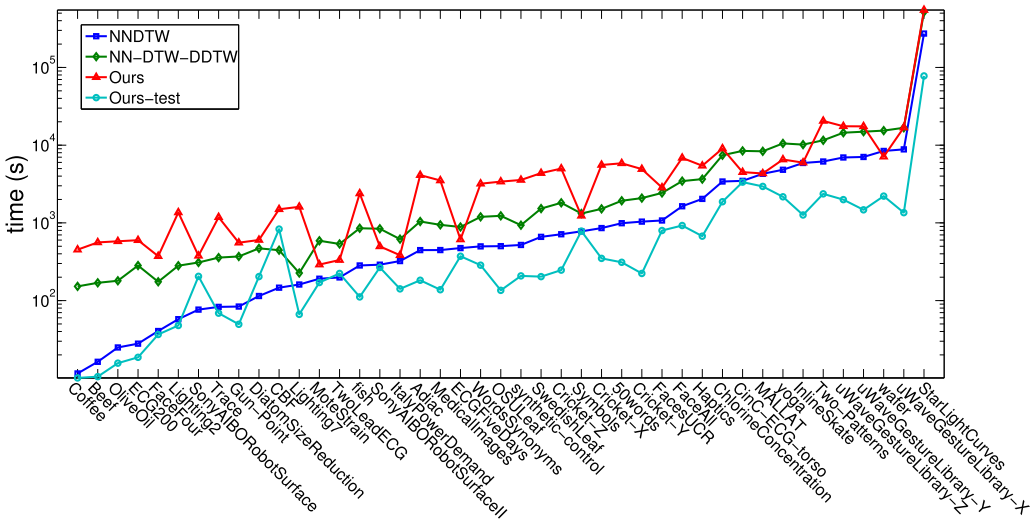

Fig. 11. Empirical time cost of our algorithm, NNDTW and NN-DTW-DDTW. Since our algorithm has two steps, training and test, we report the total running time (including training and test, red curve) and running time at test only (cyan curve). Although our algorithm in general has higher time cost than NNDTW and NN-DTW-DDTW, we are usually much more efficient at test, making our algorithm suitable for online prediction.

We tested our fused descriptor extensively on 43 standard UCR time series datasets, compared with two state-of-the-art competing algorithms, NNDTW and shapelet transform, and three other local subsequence descriptors, including DFT, Zhang's and DWT, and experimental results showed our fused descriptor performs significantly better than all competing algorithms and local descriptors. To the best of our knowledge, the feature point detector and two local descriptors are first introduced here.

Since our system is essentially a Bag-of-Words pipeline, the temporal information in the time series is not encoded by its Fisher Vector representation. In the case when the temporal or the phase information is important to distinguish time series from different classes, we could enhance the representation of each subsequence by its temporal span, and then process the modified subsequence descriptors by our proposed pipeline.

Acknowledgments

This work was supported by the US National Science Foundation (grant number CCF-1317433), the Office of Naval Research (N00014-13-1-0563), and the Army Research Office (W911NF-11-1-0046 and W911NF-12-1-0433). The authors affirm that the views expressed herein are solely their own, and do not represent the views of the United States government or any agency thereof.

References

[1] L. Ye and E. Keogh, "Time series shapelets: A new primitive for data mining," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 947–956.

[2] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Mining Knowl. Discovery*, vol. 28, no. 4, pp. 851–881, 2014.

[3] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 289–297.

[4] A. Mueen, E. Keogh, and N. Young, "Logical-shapelets: An expressive primitive for time series classification," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 1154–1162.

[5] M. Zhang and A. A. Sawchuk, "A feature selection-based framework for human activity recognition using wearable multimodal sensors," in *Proc. 6th Int. Conf. Body Area Netw.*, 2011, pp. 92–98.

[6] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Inf. Sci.*, vol. 239, pp. 142–153, 2013.

[7] J. J. Rodríguez, C. J. Alonso, and H. Boström, "Boosting interval based literals," *Intell. Data Anal.*, vol. 5, no. 3, pp. 245–262, 2001.

[8] J. J. Rodríguez, C. J. Alonso, and J. A. Maestro, "Support vector machines of interval-based features for time series classification," *Knowl.-Based Syst.*, vol. 18, no. 4, pp. 171–178, 2005.

[9] M. G. Baydogan, G. Runger, and E. Tuv, "A bag-of-features framework to classify time series," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2796–2802, Nov. 2013.

[10] J. Wang, P. Liu, M. F. She, S. Nahavandi, and A. Kouzani, "Bag-of-words representation for biomedical time series classification," *Biomed. Signal Process. Control*, vol. 8, no. 6, pp. 634–644, 2013.

[11] G. E. Batista, X. Wang, and E. J. Keogh, "A complexity-invariant distance measure for time series," in *Proc. SIAM Int. Conf. Data Mining*, 2011, vol. 11, pp. 699–710.

[12] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures," *Proc. VLDB Endowment*, vol. 1, no. 2, pp. 1542–1552, 2008.

[13] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining Knowl. Discovery*, vol. 26, no. 2, pp. 275–309, 2013.

[14] T. Rakthanmanon and E. Keogh, "Fast shapelets: A scalable algorithm for discovering time series shapelets," in *Proc. 13th SIAM Int. Conf. Data Mining*, 2013, pp. 668–676.

[15] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[16] I. Laptev, "On space-time interest points," *Int. J. Comput. Vis.*, vol. 64, no. 2-3, pp. 107–123, 2005.

[17] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, 2005, vol. 1, pp. 886–893.

[18] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Proc. 9th Eur. Conf. Comput. Vis.*, 2006, pp. 428–441.

[19] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Proc. IEEE 9th Int. Conf. Comput. Vis.*, 2003, pp. 1470–1477.

[20] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2009, pp. 1794–1801.

[21] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Proc. 11th Eur. Conf. Comput. Vis.*, 2010, pp. 143–156.

[22] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods," in *Proc. Brit. Mach. Vis. Conf.*, 2011, pp. 76.1–76.12.
[23] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, C. Schmid, et al., "Evaluation of local spatio-temporal features for action recognition," in *Proc. Brit. Mach. Vis. Conf.*, 2009, pp. 124.1–124.11.
[24] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proc. ACM SIGMOD Workshop*, 2003, pp. 2–11.
[25] J. Lin, R. Khade, and Y. Li, "Rotation-invariant similarity in time series using bag-of-patterns representation," *J. Intell. Inf. Syst.*, vol. 39, no. 2, pp. 287–315, 2012.
[26] J. Windau and L. Itti, "Situation awareness via sensor-equipped eyeglasses," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 5674–5679.
[27] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proc. 18th Int. Conf. Data Eng.*, 2002, pp. 673–684.
[28] L. Chen and R. Ng, "On the marriage of Lp-norms and edit distance," in *Proc. 30th Int. Conf. Very Large Databases*, 2004, pp. 792–803.
[29] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2005, pp. 491–502.
[30] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 262–270.
[31] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, "Feature-based classification of time-series data," *Int. J. Comput. Res.*, vol. 10, no. 3, pp. 49–61, 2001.
[32] J. Grabocka and L. Schmidt-Thieme, "Invariant time-series factorization," *Data Mining Knowl. Discovery*, vol. 28, nos. 5/6, pp. 1455–1479, 2014.
[33] K. S. Candan, R. Rossini, X. Wang, and M. L. Sapino, "sDTW: computing DTW distances using locally relevant constraints based on salient feature alignments," *Proc. VLDB Endowment*, vol. 5, no. 11, pp. 1519–1530, 2012.
[34] X. Wang, K. S. Candan, and M. L. Sapino, "Leveraging metadata for identifying local, robust multi-variate temporal (rmt) features," in *Proc. IEEE 30th Int. Conf. Data Eng.*, 2014, pp. 388–399.
[35] A. Klaser and M. Marszalek, "A spatio-temporal descriptor based on 3d-gradients," in *Proc. 19th Brit. Mach. Vis. Conf.*, 2008, pp. 275:1–275:10.
[36] S. Gudmundsson, T. P. Runarsson, and S. Sigurdsson, "Support vector machines and dynamic time warping for time series," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2008, pp. 2772–2776.
[37] M. Cuturi, J.-P. Vert, Ø. Birkenes, and T. Matsui, "A kernel for time series based on global alignments," in *Proc. Int. Conf. Acoust., Speech Signal Process.*, 2007, vol. 2, pp. II-413–II-416.
[38] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2010, pp. 3360–3367.
[39] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," *arXiv preprint arXiv:1405.4506*, 2014.
[40] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. (2015, Jul.). The UCR time series classification archive [Online]. Available: www.cs.ucr.edu/ eamonn/time_series_data/
[41] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.
[42] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall (2014) [Online]. Available: https://www.uea.ac.uk/computing/machine-learning/shapelets/shapelet-results
[43] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *Proc. 1st SIAM Int. Conf. Data Mining*, 2001, vol. 1, pp. 5–7.

Jiaping Zhao received the bachelor's and master's degrees from Wuhan University in 2008 and 2010, respectively. He is currently working toward the PhD degree at iLab, University of Southern California, working under the supervision of Laurent Itti. His research interests include computer vision, data mining, visual attention, and probabilistic graphical model. He is a student member of the IEEE.

Laurent Itti received the MS degree in image processing from the Ecole Nationale Superiere des Telecommunications in Paris in 1994 and the PhD degree in computation and neural systems from the California Institute of Technology in 2000. He is currently an professor of computer science, psychology, and neurosciences at the University of Southern California. His research interests include computational neuroscience, neural networks, visual attention, and brain modeling. He is a member of the IEEE.