Fast and Accurate Time Series Classification with WEASEL

Patrick Schäfer
Humboldt University of Berlin, Germany
patrick.schaefer@hu-berlin.de

Ulf Leser
Humboldt University of Berlin, Germany
leser@informatik.hu-berlin.de

ABSTRACT

Time series (TS) occur in many scientific and commercial applications, ranging from earth surveillance to industry automation to the smart grids. An important type of TS analysis is classification, which can, for instance, improve energy load forecasting in smart grids by detecting the types of electronic devices based on their energy consumption profiles recorded by automatic sensors. Such sensor-driven applications are very often characterized by (a) very long TS and (b) very large TS datasets needing classification. However, current methods to time series classification (TSC) cannot cope with such data volumes at acceptable accuracy; they are either scalable but offer only inferior classification quality, or they achieve state-of-the-art classification quality but cannot scale to large data volumes. In this paper, we present WEASEL (Word ExtrAction for time SEries cLassification), a novel TSC method which is both fast and accurate. Like other state-of-the-art TSC methods, WEASEL transforms time series into feature vectors, using a sliding-window approach, which are then analyzed through a machine learning classifier. The novelty of WEASEL lies in its specific method for deriving features, resulting in a much smaller yet much more discriminative feature set. On the popular UCR benchmark of 85 TS datasets, WEASEL is more accurate than the best current non-ensemble algorithms at orders-of-magnitude lower classification and training times, and it is almost as accurate as ensemble classifiers, whose computational complexity makes them inapplicable even for mid-size datasets. The outstanding robustness of WEASEL is also confirmed by experiments on two real smart grid datasets, where it out-of-the-box achieves almost the same accuracy as highly tuned, domain-specific methods.

KEYWORDS

Time series; classification; feature selection; bag-of-patterns; word co-occurrences

1 INTRODUCTION

A time series (TS) is a collection of values sequentially ordered in time. TS emerge in many scientific and commercial applications, like weather observations, wind energy forecasting, industry automation, mobility tracking, etc. One driving force behind their rising importance is the sharply increasing use of sensors for automatic and high resolution monitoring in domains like smart homes [17], starlight observations [30], machine surveillance [27], or smart grids [15, 39].

CIKM'17, November 6–10, 2017, Singapore.

ACM ISBN 978-1-4503-4918-5/17/11…$15.00
DOI: https://doi.org/10.1145/3132847.3132980

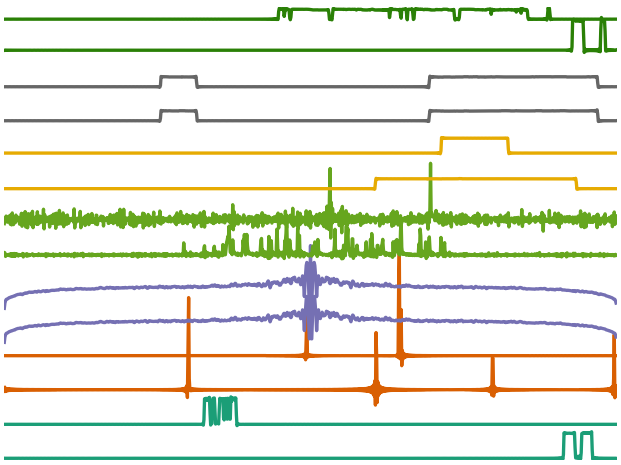


Figure 1: Daily power consumption of seven appliances with two samples per class. Bottom to top: dishwasher, microwave oven, digital receiver, coffee-maker, amplifier, lamp, monitor.

Research in TS is diverse and covers topics like storage, compression, clustering, etc.; see [10] for a survey. In this work, we study the problem of time series classification (TSC): Given a concrete TS, the task is to determine to which of a set of predefined classes this TS belongs to, the classes typically being characterized by a set of training examples. Research in TSC has a long tradition [2, 10], yet progress was focused on improving classification accuracy and mostly neglected scalability, i.e., the applicability in areas with very many and/or very long TS. However, many of today's sensor-driven applications have to deal with exactly these data, which makes methods futile that do not scale, irrespective of their quality on small datasets. Instead, TSC methods are required that are both very fast and very accurate.

As a concrete example, consider the problem of classifying energy consumption profiles of home devices (a dish washer, a washing machine, a toaster etc.). In smart grids, every device produces a unique profile as it consumes energy over time; profiles are unequal between different types of devices, but rather similar for devices of the same type (see Figure 1). The resulting TSC problem is as follows: Given an energy consumption profile (which is a TS), determine the device type based on a set of exemplary profiles per type. For an energy company such information helps to improve the prediction of future energy consumption [12, 13]. For approaching these kinds of problems, algorithms that are very fast and very accurate are required. Regarding **scalability**, consider millions of customers each having dozens of devices, each recording one measurement per second. To improve forecasting, several millions of classifications of time series have to be performed every hour, each

considering thousands of measurements. Even with TS sampling or adaptive re-classification intervals the number of classifications remains overwhelming and can only be approached with very fast TSC methods. Regarding **accuracy**, it should be considered that any improvement in prediction accuracy may directly transform into substantial monetary savings. For instance, [15, 39] report that a small improvement in accuracy (below 10%) can save tens of millions of dollars per year and company. However, achieving high accuracy classification of home device energy profiles is non trivial due to different usage rhythms (e.g., where in a dishwasher cycle has the TS been recorded?), differences in the profiles between concrete devices of the same type, and noise within the measurements, for instance because of the usage of cheap sensors.

Current TSC methods are not able to deal with such data at sufficient accuracy and speed. Several high accuracy classifiers, such as Shapelet Transform (ST) [6], have bi-quadratic complexity (power of 4) in the length of the TS; even methods with quadratic classification complexity can be infeasible. The current most accurate method (COTE [3]) even is an ensemble of dozens of core classifiers many of which have a quadratic, cubic or bi-quadratic complexity. On the other hand, fast TSC methods, such as BOSS VS [35] or Fast Shapelets [33], perform much worse in terms of accuracy compared to the state of the art [2]. As a concrete example, consider the (actually rather small) PLAID benchmark dataset [12], consisting of 1074 profiles of variable-length (10^2 to 10^3) measurements each stemming from 11 different devices. Figure 2 plots classification times in log scale (including all preprocessing steps) versus accuracy for nine state-of-the-art TSC methods and the novel algorithm presented in this paper, WEASEL. Euclidean distance (ED) based methods are the fastest, but their accuracy is far below standard. Dynamic Time Warping methods (DTW, DTW CV) are common baselines and show a moderate runtime of 10 to 100 ms but also low accuracy. Highly accurate classifiers such as ST [6] and BOSS [36] require orders-of-magnitude longer prediction times. For this rather small dataset, the COTE ensemble classifier has not yet terminated training after 12 CPU weeks, thus we cannot report the accuracy, yet. In summary, the fastest methods for this dataset require around 1ms per prediction, but have an accuracy below 80%; the most accurate methods achieve 85%-88% accuracy, but require 80ms up to 32sec for prediction.

In this paper, we propose a new TSC method called *WEASEL: Word ExtrAction for time SEries cLassification*. WEASEL is both fast and very accurate; for instance, on the dataset shown in Figure 2 it achieves the highest accuracy while being the third-fastest algorithm (requiring only 4ms per prediction). Like several other methods, WEASEL conceptually builds on the bag-of-patterns (BOP) approach: It moves a sliding window over a TS and extracts discrete features per window which are subsequently fed into a machine learning classifier. However, the way of constructing and filtering features in WEASEL is completely different from previous methods:

(1) **Discriminative feature generation**: WEASEL derives discriminative features based on the dataset labels. Specifically, our novel supervised symbolic representation determines the most discriminative Fourier coefficients using an ANOVA f-test and finally applies information gain binning for choosing appropriate discretization boundaries. This differs from previous approaches, like SFA [37] or SAX [20], which rely on fixed, data-independent intervals, possibly leading to features equally frequent across classes.

(2) **Co-occurring words & variable-length windows**: WEASEL extracts windows at multiple lengths and also considers the order of windows (using bi-grams as features) instead of considering each fixed-length window as independent feature. It then builds a single model from the concatenation of feature vectors. So, instead of training $O(n)$ different models, and picking the best one, we weigh each feature based on its relevance to predict the class.

(3) **Feature selection**: The wide range of features (bigrams, unigrams and window lengths) considered introduces many irrelevant features. Therefore, WEASEL first applies an aggressive statistical feature selection to remove irrelevant features from each class, without negatively impacting accuracy and heavily reducing runtime. The resulting feature set is highly discriminative, which allows us to use fast logistic regression instead of more elaborated, but also more runtime-intensive methods.

(4) **Evaluation**: We performed a series of experiments on a total of 85 TS datasets and two use cases to assess the impact of these improvements. WEASEL outperforms the best state-of-the-art core-classifiers in terms of accuracy while also being one of the fastest methods; it is almost as accurate as the current overall best ensemble classifier (COTE) but multiple orders-of-magnitude faster in training and in classification times.

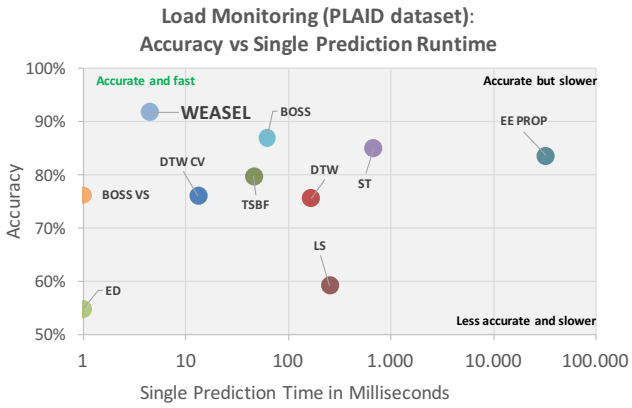


Figure 2: Classification accuracy and single prediction runtime (log scale) for different TSC methods on the energy consumption dataset PLAID. Prediction times include all preprocessing steps (feature extraction, etc.). Methods are explained in Section 2, the system used is described in Section 5.

The rest of this paper is organized as follows: In Section 2 we present related work. Section 3 briefly recaps bag-of-patterns classifiers and feature discretization using Fourier transform. In Section 4 we present WEASEL's novel way of feature generation and selection. Section 5 presents evaluation results.

2 RELATED WORK

With time series classification (TSC) we denote the problem of assigning a given TS to one of a predefined set of classes. The techniques used for TSC can be broadly categorized into two classes: whole series-based methods and feature-based methods [21]. *Whole series* similarity measures make use of a point-wise comparison of entire TS. These include 1-NN Euclidean Distance (ED) or 1-NN Dynamic Time Warping (DTW) [32], which is commonly used as a baseline in comparisons [2, 22]. Typically, these techniques work well for short but fail for noisy or long TS [36]. Furthermore, DTW has a computational complexity of $O(n^2)$ for TS of length n. Techniques like early pruning of candidate TS with cascading lower bounds [32], or clustering and indexing [8, 28] have been applied to reduce the effective runtime.

In contrast, *feature-based* classifiers rely on comparing features generated from substructures of TS. The most successful approaches can be grouped as either using shapelets or bag-of-patterns (BOP). *Shapelets* are defined as TS subsequences that are maximally representative of a class. In [26] a decision tree is built on the distance to a set of shapelets. The Shapelet Transform (ST) [6, 23], which is the most accurate shapelet approach according to a recent evaluation [2], uses the distance to the shapelets as input features for an ensemble of different classification methods. In the Learning Shapelets (LS) approach [14], optimal shapelets are synthetically generated. The drawback of shapelet methods is the high computational complexity resulting in rather long training and classification times. In [18] a forest of shapelet-based decision trees is built using a random subset of shapelets for each decision tree. After 300 CPU days of training and correspondence with the authors, we had to omit this classifier for the evaluation, as we were unable to exactly reproduce the reported results.

The alternative approach within the class of feature-based classifiers is the *bag-of-patterns (BOP)* model [21]. Such methods break up a TS into a bag of substructures, represent these substructures as discrete features, and finally build a histogram of feature counts as basis for classification. The first published BOP model (which we abbreviate as BOP-SAX) uses sliding windows of fixed lengths and transforms these measurements in each window into discrete features using Symbolic Aggregate approXimation (SAX) [20]. Classification is implemented as 1-NN classifier using ED of feature counts as distance measure. SAX-VSM [38] extends BOP-SAX with *tf-idf* weighing of features and uses the Cosine distance; furthermore, it builds only one feature vector per class instead of one vector per sample, which drastically reduces runtime. Another current BOP algorithm is the TS bag-of-features framework (TSBF) [4], which first extracts windows at random positions with random lengths and next builds a supervised codebook generated from a random forest classifier. The BOP-based algorithm BOSS (Bag-of-SFA-Symbols) [36] uses the Symbolic Fourier Approximation (SFA) [37] instead of SAX, and is currently the most accurate BOP-based method. In contrast to shapelet-based approaches, BOP-based methods typically have only linear complexity for prediction.

The most accurate current TSC algorithms are ensembles. These classify a TSC by a set of different core classifiers and then aggregate the results using techniques like bagging or majority voting.

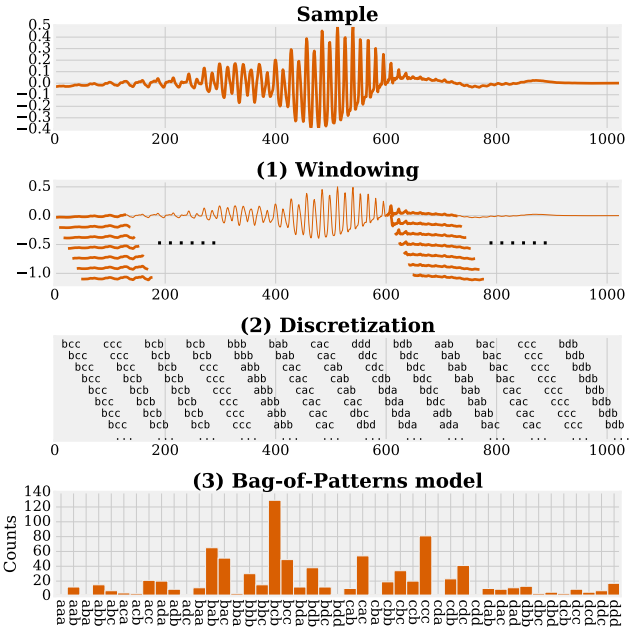


Figure 3: Transformation of a TS into the Bag-of-Patterns (BOP) model using overlapping windows (second to top), discretization of windows to words (second from bottom), and word counts (bottom).

The Elastic Ensemble (EE PROP) classifier [22] uses 11 whole series classifiers. The COTE ensemble [3] is based on 35 core-TSC methods including EE PROP and ST. Very recently, there has been an extension HIVE-COTE to COTE [24] that incorporates classifiers from five domains and claims superior accuracy. WEASEL should be seen as a possible replacement of BOSS in HIVE-COTE. If designed properly, ensembles combine the advantages of their core classifiers, which often lead to superior results. However, the price to pay is excessive runtime requirement for training and for classification, as each core classifier is used independently of all others.

In [41] deep learning networks are applied to TSC. Their best performing full convolutional network (FCN) out-of the-box performs not significantly different from state of the art, without any pre-preprocessing or feature crafting. In [41] multi-scale convolutional neural networks (MCNN) are presented for TSC. The MCNN extracts features at different scales and frequencies, similar to the windowing approach in BOP. In their experimental evaluation their MCNN is not significantly different from COTE and BOSS. Experiments in both papers are based on only 44 out of the 85 UCR datasets tested here. Thus, we can not directly compare these results to the ones in our experiments. Furthermore, both implementations need a GPU, as opposed to the CPU based time measurements we did here. Thus, we have to omit these two papers in our evaluation.

3 TIME SERIES, BOP, AND SFA

In this work, a *time series (TS)* T is a sequence of $n \in \mathbb{N}$ real values, $T = (t_1, \ldots, t_n)$, $t_i \in \mathbb{R}$. As we primarily address TS

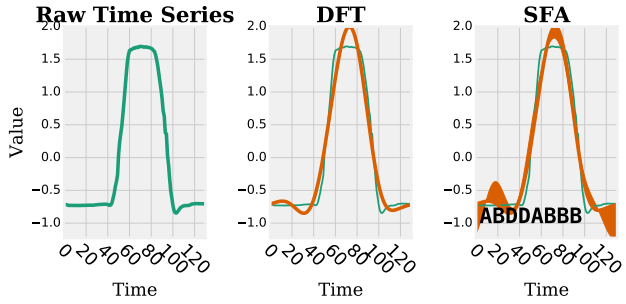


Figure 4: The Symbolic Fourier Approximation (SFA): A time series (left) is approximated using the truncated Fourier transform (center) and discretized to the word *ABDDABBB* (right) with the four-letter alphabet ('a' to 'd'). The inverse transform is depicted by an orange area (right), representing the tolerance for all signals that will be mapped to the same word.

generated from automatic sensors with a fixed sampling rate, we ignore time stamps. Given a TS T, a *window* S of length w is a subsequence with w contiguous values starting at offset a in T, i.e., $S(a, w) = (t_a, \ldots, t_{a+w-1})$ with $1 \leq a \leq n - w + 1$. We associate each TS with a class label $y \in Y$ from a predefined set Y. *Time series classification (TSC)* is the task of predicting a class label for a TS whose label is unknown. A TS classifier is a function that is learned from a set of labeled time series (the training data), takes an unlabeled time series as input and outputs a label.

Algorithms following the BOP model build this classification function by (1) extracting windows from a TS, (2) transforming each window of real values into a discrete-valued *word* (a sequence of symbols over a fixed alphabet), (3) building a feature vector from word counts, and (4) finally using a classification method from the machine learning repertoire on these feature vectors. Figure 3 illustrates these steps from a raw time series to a BOP model using overlapping windows.

BOP methods differ in the concrete way of transforming a window of real-valued measurements into discrete words (discretization). For example, the basis of the BOSS model is a symbolic representation called SFA, which generates features independent of the actual class labels. In [37] SFA works as follows: (1) Values in each window are normalized to have standard deviation of 1 to obtain amplitude invariance. (2) Each normalized window of length w is subjected to dimensionality reduction by the use of the truncated Fourier transform, keeping only the first $l < w$ coefficients for further analysis. This step acts as a low pass filter, as higher order Fourier coefficients typically represent rapid changes like dropouts or noise. (3) Each coefficient is discretized to a symbol of an alphabet of fixed size c to achieve further robustness against noise. Figure 4 exemplifies this process. Each of these steps ignores class labels.

4 WEASEL

In this section, we present our novel TSC method WEASEL (Word ExtrAction for time SEries cLassification). WEASEL specifically addresses the major challenges any TSC method has to cope with

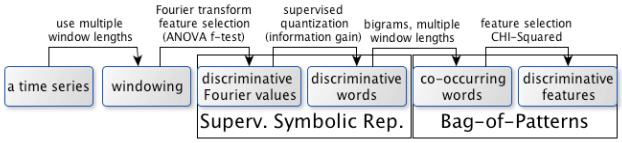


Figure 5: WEASEL Pipeline: Feature extraction using our novel supervised symbolic representation, and our novel bag-of-patterns model.

when being applied to data from sensor readouts, which can be summarized as follows (using home device classification as an example):

(1) *Invariance to noise*: TS can be distorted by (ambiance) noise as part of the recording process. In a smart grid, such distortions are created by imprecise sensors, information loss during transmission, stochastic differences in energy consumption, or interference of different consumers connected to the same power line.

(2) *Scalability*: TS in sensor-based applications are typically recorded with high sampling rates, leading to long TS. Furthermore, smart grid applications typically have to deal with thousands or millions of TS.

(3) *Variable lengths and offsets*: TS to be classified may have variable lengths, and recordings of to-be-classified intervals can start at any given point in time. In a smart grid, sensors produce continuous measurements, and the partitioning of this essentially infinite stream into classification intervals is independent from the usages of devices. Thus, characteristic patterns may appear anywhere in a TS (or not at all), but typically in the same order.

(4) *Unknown characteristic substructures*: Feature-based classifiers exploit local substructures within a TS, and thus depend on the identification of recurring, characteristic patterns. However, the position, form, and frequency of these patterns is unknown; many substructures may be irrelevant for classification. For instance, the idle periods of the devices in Figure 1 are essentially identical.

We carefully engineered WEASEL to address these challenges. Our method conceptually builds on the BOP model in BOSS [36], yet uses rather different approaches in many of the individual steps. We will use the terms *feature* and *word* interchangeably throughout the text.

The main contribution of WEASEL is not the approach itself, but the several pieces that compose the solution related to feature crafting. WEASEL is composed of the building blocks depicted in Figure 5: our novel supervised symbolic representation for discriminative feature generation and the novel bag-of-patterns model for building a discriminative feature vector. First, WEASEL extracts normalized windows of varying lengths from a time series. Next, each window is approximated using the Fourier transform, and the real and imaginary Fourier values are kept that best separate TS from different classes using the ANOVA F-test - as opposed to using the truncated Fourier transform. These Fourier values are discretized into a word based on information gain binning, which also chooses discretization boundaries to best separate the TS classes; More detail

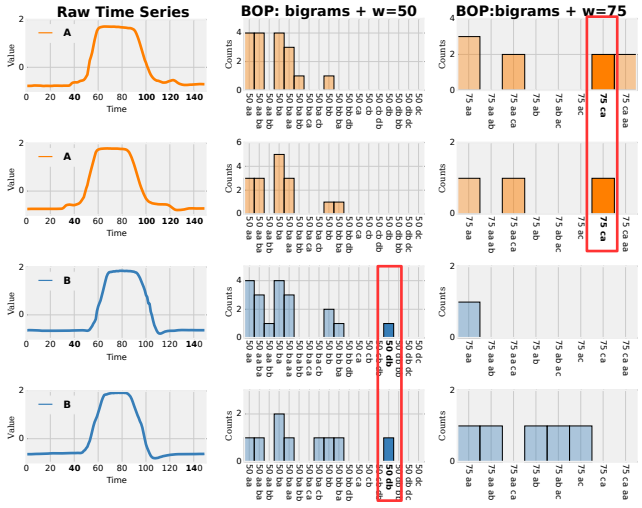


Figure 6: Discriminative feature vector: Four time series, two from class 'A' and two from class 'B' are shown. Feature vectors contain unigrams and bigrams for the two exemplary window lengths 50 and 75. The discriminative words are highlighted.

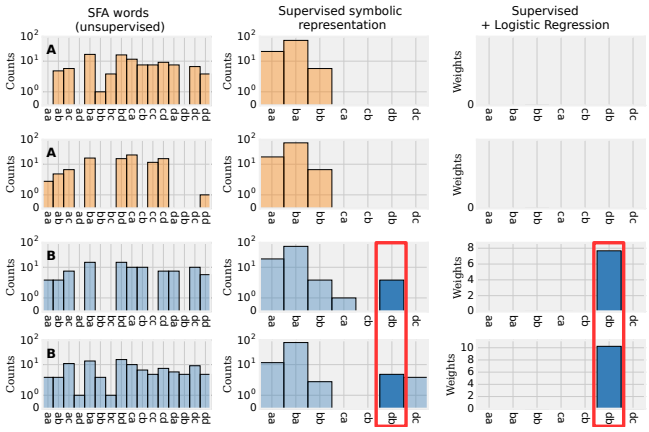


Figure 7: Influence of the word model for feature extraction. From left to right: SFA words, our novel supervised representation words, and the weighed words after logistic regression.

is given in Subsection 4.2. Finally, a single bag-of-patterns is built from the words (unigrams), neighboring words (bigrams), and all window lengths. To filter irrelevant features, the Chi-squared test is applied to this bag-of-patterns (Subsection 4.1). As WEASEL builds a highly discriminative feature vector, a fast linear time logistic regression classifier is applied (Subsection 4.1).

BOP-based methods have a number of parameters, which heavily influence their performance. Of particular importance is the window length w. An optimal value for this parameter is typically learned for each new dataset using techniques like cross-validation. This does not only carry the danger of over-fitting (if the training samples are biased compared to the to-be-classified TS), but also leads to substantial training times. In contrast, WEASEL removes the need to set this parameter, by constructing a single high-dimensional feature vector, in which the whole feature space is encoded. Thus, each feature in this vector is the concatenation of the concrete parameter values: i.e., '50 db' (an unigram for length 50) or '75 aa ca' (a bigram for length 75). All window-lengths (there are $O(n)$ window lengths), unigrams and bigrams are enumerated in this way in a (sparse) feature vector. So, instead of training $O(n)$ different models, and picking the best one for prediction, we weigh each feature based on its importance to predict a class label, and remove those that are irrelevant.

Figure 6 illustrates our use of unigrams, bigrams and two exemplary window lengths. The depicted dataset contains two classes 'A' and 'B' with two samples each. The time series are very similar and differences between these are difficult to spot, and are mostly located between time stamps 80 and 100 to 130. The center (right) column illustrates the features extracted for window length 50 (75). Feature *'75 aa ca'* is characteristic for the *A* class, whereas the feature *'50 db'* is characteristic for the *B* class. Thus, we use different window lengths, bigrams, and unigrams to capture subtle differences between TS classes. We show the impact of varying window lengths and bigrams to classification accuracy in Section 5.5.

4.1 Feature Extraction: A Novel Supervised Symbolic Representation

A symbolic representation is used to transform a real-valued TS window to a word using an alphabet of size c. Our representation is based on SFA [37]. The problem with SFA is that it (a) filters the high frequency components of the signal, just like a low-pass filter. But for instance, the pitch (frequency) of a bird sound is relevant for the species but lost after low-pass filtering. Furthermore, it (b) does not distinguish between class labels when quantizing values of the Fourier transform. Thus, there is a high likelihood of SFA words to occur in different classes with roughly equal frequencies. For classification, we need discriminative words for each class. Instead, our approach is based on two steps:

(1) *Discriminative approximation* by applying feature selection to the approximation step and keeping those Fourier values whose distribution best separates the class labels in disjoint groups, as opposed to using the first ones.

(2) *Discriminative quantization* by using information gain [31] to minimize the entropy of the class labels for each split. I.e., the majority of values in each partition correspond to the same class label.

In Figure 7 we revisit our sample dataset. This time with a window length of 25. When using SFA words (left), the words are evenly spread over the whole BOP for both prototypes. There is no single feature whose absence or presence is characteristic for a class. However, when using our novel supervised symbolic representation (center), we observe less distinct words, more frequent counts and the word *'db'* is unique within the *'B'* class. When training a logistic regression classifier on these words (right), the word *'db'* gets boosted and other words are filtered. Note, that the counts of the word *'db'* differ for both representations, as it represents other frequency ranges for both approaches.

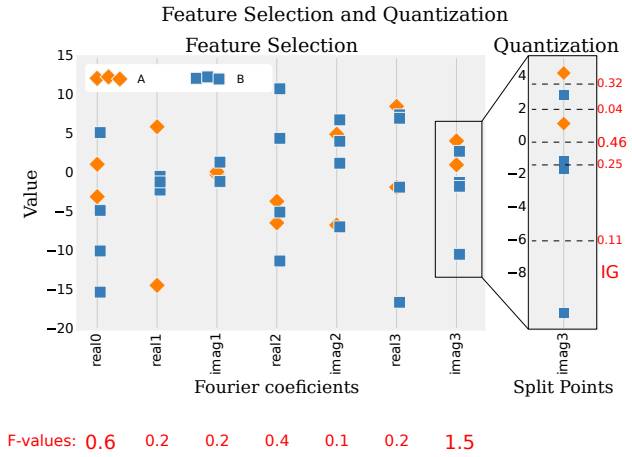


Figure 8: On the left: Distribution of Fourier coefficients for our sample dataset. The high F-values on imag3 and real0 (red text at bottom) should be selected to best separate the samples from class labels 'A' and 'B'. On the right: Zoom in on imag3. Information gain splitting is applied to find the best bins for the subsequent quantization. High information gain (IG) indicates pure (good) split points.

4.1.1 Discriminative Approximation using One-Way ANOVA F-test.

For approximation, each TS is Fourier transformed first. We aim at finding the top l real an imaginary Fourier values that best separate between class labels for a set of TS samples, instead of simply taking the first l ones. Figure 8 (left) shows the distribution of the Fourier values for the sample-dataset. The Fourier value that best separates between the classes is $imag_3$ with the highest F-value of 1.5 (bottom).

We use a one-way ANOVA F-test [25] to select those l real and imaginary Fourier values, that best separate between class labels for a set of TS samples, instead of simply taking the first l ones. The one-way ANOVA F-test checks the hypothesis that two or more groups have the same normal distribution around the mean. The analysis is based on two estimates for the variance existing within and between groups: *mean square within* (MS_W) and *mean square between* (MS_B). The F-value is then defined as: $F = \frac{MS_B}{MS_W}$. When used as part of feature selection, we are interested in the largest F-values, equal to large differences between group means. We keep those l real or imaginary Fourier values with the largest F-values. In Figure 8 these are $real_0$ and $imag_3$ for $l = 2$ with F-values 0.6 and 1.5.

The ANOVA F-test assumes that the data follows a normal distribution with equal variance. The BOP (WEASEL) approach extracts subsequences for z-normalized time series. It has been shown that subsequences extracted from z-normalized time series perfectly mimic normal distribution [19]. Furthermore, the Fourier transform of a normal distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{-x^2}{2\sigma^2}}$$

with $\mu = 0, \sigma = 1$ results in a normal distribution of the Fourier coefficients [7]:

$$F(t) = \int f(x) \cdot e^{-itx} = e^{i\mu\sigma} e^{-\frac{1}{2}(\sigma t)^2} = e^{-\frac{1}{2}(\sigma t)^2}$$

Thus, the Fourier coefficients follow a symmetrical and uni-modal normal distribution with equal variance.

The ANOVA F-test further assumes that the samples are independently drawn. To guarantee independence, we are extracting disjoint subsequences, i.e., non-overlapping, to train the quantization intervals. Using disjoint windows for sampling further decreases the likelihood of over-fitting quantization intervals.

4.1.2 Discriminative Quantization using Entropy / Information Gain.

A supervised quantization step is applied to find for each selected real or imaginary Fourier value the best split points, so that in each partition a majority of values correspond to the same class. Our quantization is based on binning (bucketing): the value range is partitioned into disjoint intervals, called bins. Each bin is labeled by a symbol. A real value that falls into an interval is represented by its symbol. Common methods to partition the value range include equi-depth or equi-width bins, as in SAX or SFA. These ignore the class label distribution and splits are solely based on the value distribution. Here we introduce entropy-based binning (information gain [31]). This leads to disjoint feature sets. Let $Y = \{(s_1, y_1), \ldots, (s_N, y_N)\}$ be a list of value and class label pairs with N unique class labels. The multi-class entropy is then given by: $Ent(Y) = \sum_{(s_i,y_i)\in Y} -p_{y_i} \log_2 p_{y_i}$, where p_{y_i} is the relative frequency of label y_i in Y. The entropy for a split point sp with all labels on the left $Y_L = \{(s_i, y_i) | s_i \leq sp, (s_i, y_i) \in Y\}$ and all labels on the right $Y_R = \{(s_i, y_i) | s_i > sp, (s_i, y_i) \in Y\}$ is given by:

$$Ent(Y, sp) = \frac{|Y_L|}{|Y|} Ent(Y_L) + \frac{|Y_R|}{|Y|} Ent(Y_R) \quad (1)$$

The information gain for this split is given by:

$$Information\ Gain = Ent(Y) - Ent(Y, sp) \quad (2)$$

We fix the alphabet size c to 4, as it has been shown in the context of BOP models that using a constant $c = 4$ is very robust over all TS considered [21, 36, 38].

4.2 Feature Selection and Weighting: Chi-squared Test and Logistic Regression

The dimensionality of this feature space is $O(\min(Nn^2, c^l))$ for word length l and c symbols. The number of TS N and length n affects the actual word frequencies. But in the worst case each TS window can only produces a distinct word, and there are Nn^2 windows.

For common parameters like $c = 4$, $l = 4$, $n = 256$ this results in a sparse vector with $4^4 = 256$ dimensions for a TS. WEASEL uses bigrams and $O(n)$ window lengths, thus the dimensionality of the feature space rises to $O(\min(Nn^2, c^{2l} \cdot n))$. For the previous set of parameters this feature space explodes to up to $4^8 \cdot 256 = 256^3$.

WEASEL uses the Chi-squared (χ^2) test to identify the most relevant features in each class to reduce this feature space to a few hundred features prior to training the classifier. This statistical test determines if for any feature the observed frequency within a specific group significantly differs from the expected frequency,

assuming the data itself is discrete (nominal). Larger χ^2-values imply that a feature occurs more frequently within a specific class. Thus, we keep those features with χ^2-values above a threshold. This highlights subtle distinctions between classes. All other features can be considered superfluous and are removed. On average this reduces the size of the feature space by $30-70\%$ to 10^4 up to 10^5 features. In our pipeline, the main aim of this threshold is to be high enough for the logistic regression classifier to train a model in reasonable time (when set too low, training takes longer).

We use sparse vectors to store the features for each time series, as each feature vector only contains a few features after feature selection. We implemented our classifier using liblinear [11] as it scales linearly with the dimensionality of the feature space [29], which is bound by $O(\min(Nn^2,\ c^{2l}\cdot n))$. This results in a moderate runtime compared to Shapelet or ensemble classifiers with cubic and bi-quadratic runtimes in n, which is orders of magnitude slower (compare Section 5.3).

5 EVALUATION

5.1 Experimental Setup

We mostly evaluated our *WEASEL* classifier using the full UCR benchmark dataset of 85 TSC problems [43][1]. Furthermore, we compared its performance on two real-life datasets from the smart grid domain; results are reported in Section 5.6. Each UCR dataset provides a train and test split set which we use unchanged to make our results comparable to prior publications. It can be computationally very expensive to run classifiers and we spent 800 CPU days for training. Thus, we had to limit our comparison of WEASEL to the 9 state-of-the-art TSC methods (following [2]), namely COTE [3], BOSS [36], BOSS VS [35], LS [14], EE (PROP) [22], TSBF [4], ST [6], and the baselines DTW and DTW CV [22]. All experiments ran on a server running LINUX with 2xIntel Xeon E5-2630v3 and 64GB RAM, using JAVA JDK x64 1.8. We measured runtimes of all methods using the implementation given by the authors [5] wherever possible, resorting to the code by [2] if this was not the case. For DTW and DTW CV, we make use of the state-of-the-art cascading lower bounds from [32]. We measure CPU time, to address parallel and single threaded codes. Regarding accuracy, we report numbers published by each author [1, 3, 4, 14], complemented by the numbers published by [40], for those datasets where results are missing (due to the growth of the benchmark datasets). All numbers are accuracy on the test split. For WEASEL we performed 10-fold cross-validation on the training datasets to find the most appropriate value for the SFA word length $l \in [4, 6, 8]$. We kept $c = 4$ and $chi = 2$ constant, as varying these values has only negligible effect on accuracy (data not shown). We used *liblinear* with default parameters ($bias = 1, p = 0.1$ and solver L2R_LR_DUAL). To ensure reproducible results, we provide the WEASEL source code and the raw measurement sheets. [42].

5.2 Accuracy

Figure 9 shows a critical difference diagram (introduced in [9]) over the average ranks of the different TSC methods. Classifiers with the lowest (best) ranks are to the right. The group of classifiers that are not significantly different in their rankings are connected by a bar. The critical difference (CD) length represents statistically significant differences. The 1-NN DTW and 1-NN DTW CV classifiers are commonly used as benchmarks [22]. Both perform significantly worse than all other methods. ST, LS and BOSS have a similar rank and competitive accuracies. WEASEL has the lowest rank among all core classifiers (DTW, TSBF, LS, BOSS, BOSS VS, ST). Ensemble classifiers generally show compelling accuracies, with COTE being the most accurate. The advantage of WEASEL is its much lower runtime, which we address in Section 5.3.

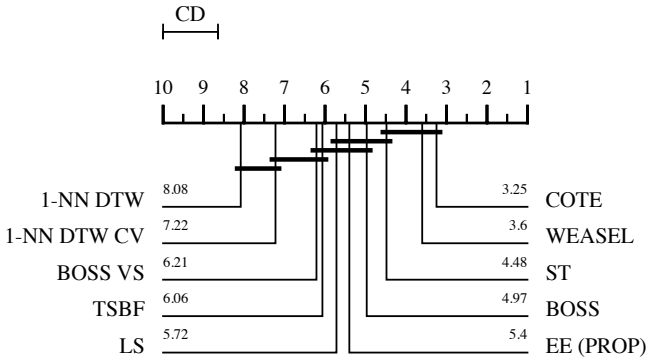


Figure 9: Average ranks on the 85 UCR datasets. WEASEL is as accurate as state of the art.

5.3 Scalability

Figure 10 plots for all TSC methods the total runtime on the x-axis in log scale vs the average rank on the y-axis for prediction. Runtimes include all preprocessing steps like feature extraction or selection. Because of the high wall-clock time of some classifiers, we had to limit this experiment to the 45 core UCR datasets, encompassing roughly $N = 17000$ train and $N = 62000$ test time series. The total time spent for training were 800 CPU days. Train times vary from more than 300 CPU days (EE prop and COTE) to 8 CPU days for DTW CV. WEASEL and BOSS have similar train times of one CPU day and as such train one to two orders of magnitude faster than the other core classifiers.

There are fast methods (BOSS VS, TSBF, LS, DTW CV) that require a few ms per prediction, but have a low average rank; and there are accurate methods (ST; BOSS; EE; COTE) that require hundredths of ms to seconds per prediction. The two ensemble methods in our comparison, EE PROP and COTE, show the highest prediction times. WEASEL is consistently among the best and fastest predicting methods, and competitors are (a) either at the same level of quality (COTE) but much slower or (b) faster but much worse in accuracy (LS, DTW CV, TSBF, or BOSS VS). As for all BoP approaches, WEASEL'S train times are among the lowest.

Of course, WEASEL is not faster and more accurate than every single competitor. For example, DTW CV has on average lower prediction times than WEASEL at a significantly lower accuracy. The complexity of DTW CV is $O(Nnr)$ with warping window size r.

[1]The UCR archive has recently been extended from 45 to 85 datasets.

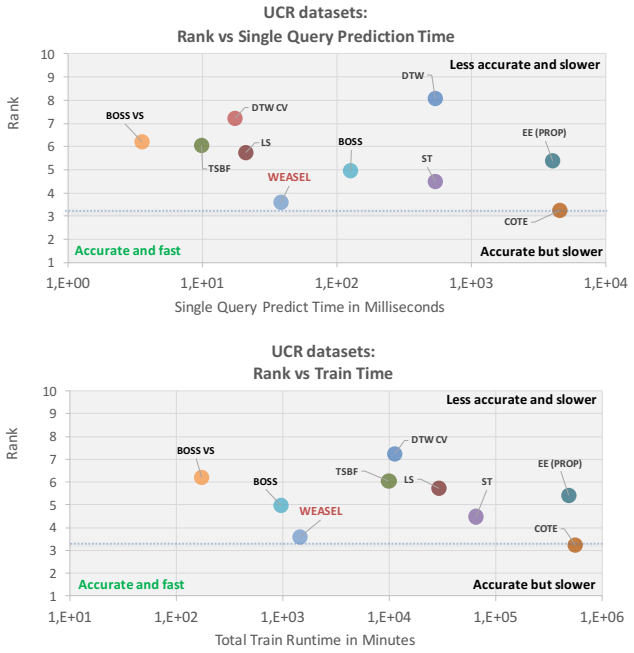


Figure 10: Average single prediction and training times in log scale vs average rank (lower is better). Runtimes include all preprocessing steps (feature extraction, bop model building, etc.) and cross-validation costs for model training. WEASEL has a similar average accuracy as COTE but is two orders of magnitude faster. A single prediction takes 38ms on average.

So, there is always a trade-off between accuracy and prediction/train times. However, this experiment underlines that WEASEL offers a very high accuracy at fast prediction and train times.

5.4 Accuracy by datasets and by domain

In this experiment we found that WEASEL performs well independent of the domain. We studied the individual accuracy of each method on each of the 85 different datasets, and also grouped datasets by domain to see if different methods have domain-dependent strengths or weaknesses. We used the predefined grouping of the benchmark data into four types: synthetic, motion sensors, sensor readings and image outlines. For this experiment, we only consider the non-ensemble classifiers. Figure 11 shows the accuracies of WEASEL (black line) vs. the six core classifiers ST, LS, BOSS, DTW, DTW CV, TSBF (orange area).

Overall, the performance of WEASEL is very competitive for almost all datasets. The black line is mostly very close to the upper outline of the orange area, indicating that WEASEL's performance is close to that of its best competitor. In total WEASEL has 36 out of 85 wins against the group of six core classifiers. On 69 (78) datasets it is not more than 5% (10%) to the best classifier. Overall, WEASEL has the highest percentage of wins in the groups of sensor readings, synthetic and image outline datasets. Within the group of motion sensors, it performs equally good as LS and ST.

The main advantage of WEASEL is that it adapts to variable-length characteristic substructures by calculating discriminative features in combination with noise filtering. Thus, all datasets that are composed of characteristic substructures benefit from the use of WEASEL. This applies to most sensor readings like all EEG or ECG signals, but also mass spectrometry, or recordings of insect wing-beats. These are typically noisy and have variable-length, characteristic substructures that can appear at arbitrary time stamps [16]. ST also fits to this kind of data but, in contrast to WEASEL, is sensitive to noise.

5.5 Influence of Design Decisions on WEASEL's Accuracy

We look into the impact of design decisions on the WEASEL classifier. Figure 12 shows the average ranks of the WEASEL classifier where each extension is disabled or enabled: (a) "one (single) window length, supervised and bigrams", (b) "(varying lengths) unsupervised and unigrams", (c) "(varying lengths) unsupervised and bigrams", (d) "(varying lengths) supervised and unigrams", and (e) "supervised and bigrams". The single window approach is least accurate. This underlines that the choice of window lengths is crucial for accuracy. The unsupervised symbolic representation approach with unigrams is equal to the standard BOP model. Using a supervised symbolic representation or bigrams slightly improves the ranks. All of WEASEL's extensions combined, significantly improve the ranks (unigrams, bigrams, varying lengths).

5.6 Use Case: Smart Plugs

Appliance load monitoring has become an important tool for energy savings [12, 13]. We tested the performance of different TSC methods on data obtained from intrusive load monitoring (ILM), where energy consumption is separately recorded at every electric device. We used two publicly available datasets ACS-F1 [13] and PLAID [12]. These datasets capture the power consumption of typical appliances. Each appliance has a characteristic shape. Some appliances show repetitive substructures while others are distorted by noise. The recordings are characterized by long idle periods and some high bursts of energy consumption when the appliance is active. When active, appliances show different operational states. Figure 2 and 13 show the accuracy and runtime of WEASEL compared to state of the art. WEASEL scores the highest accuracies with 92% and 91.8% for both datasets. With a prediction time of 10 and 100 ms it is also fast. Train times of WEASEL are comparable to that of DTW CV and orders lower than that of the other high accuracy classifiers (data not shown). On the large PLAID dataset WEASEL has a significantly lower prediction time than its competitors, while on the smaller ACS-F1 dataset the prediction time is slightly higher than that of DTW or BOSS. 1-NN classifiers such as BOSS and DTW scale with the size of the train dataset. Thus, for larger train datasets, they become slower. At the same time, for small datasets like PLAID, they can be quite fast. The results show that our approach naturally adapts to appliance load monitoring. These data show how WEASEL automatically adapts to idle and active periods

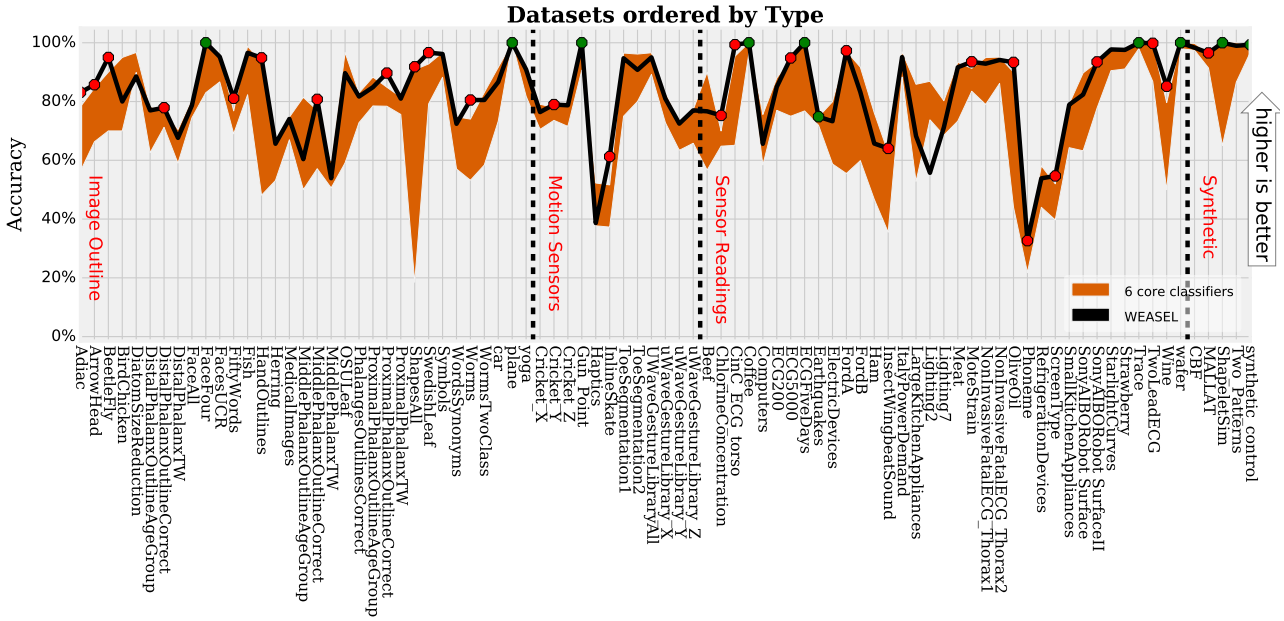


Figure 11: Classification accuracies for WEASEL vs the best six core classifiers (ST, LS, BOSS, DTW, DTW CV, TSBF). The orange area represents the six core classifiers' accuracies. Red (green) dots indicate where WEASEL wins (evens out) against the other classifiers.

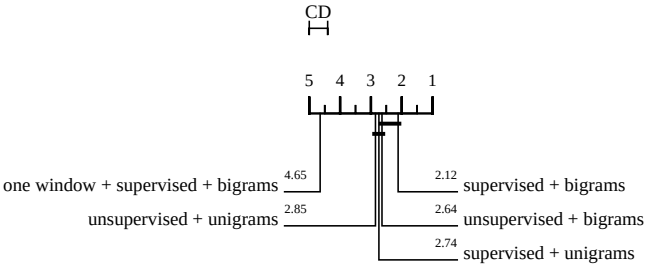


Figure 12: Impact of design decisions on ranks. The WEASEL (supervised+bigrams) classifier has the lowest rank over all datasets.

and short, repetitive characteristic substructures, which were also important in the sensor readings or image outline domains (Section 5.4). Note that the authors of the ACS-F1 dataset scored 93% [34] using a hidden Markov model and a manual feature set. Unfortunately their code is not available and the runtime was not reported. Our accuracy is close to theirs, while our approach was not specially adapted for the domain.

6 CONCLUSION

In this work, we have presented WEASEL, a novel TSC method following the bag-of-pattern approach which achieves highly competitive classification accuracies and is very fast, making it applicable in domains with high runtime and quality constraints. The novelty of WEASEL is its carefully engineered feature space using statistical feature selection, word co-occurrences, and a supervised symbolic representation for generating discriminative words. Thereby, WEASEL assigns high weights to characteristic, variable-length substructures of a TS. In our evaluation on altogether 87 datasets, WEASEL is consistently among the best and fastest methods, and competitors are either at the same level of quality but much slower or faster but much worse in accuracy.

REFERENCES

[1] Anthony Bagnall, Luke M. Davis, Jon Hills, and Jason Lines. 2012. Transformation Based Ensembles for Time Series Classification. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, Vol. 12. SIAM, 307–318.

[2] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. 2016. The Great Time Series Classification Bake Off: An Experimental Evaluation of Recently Proposed Algorithms. Extended Version. *Data Mining and Knowledge Discovery* (2016), 1–55.

[3] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. 2015. Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles. *IEEE Transactions on Knowledge and Data Engineering* 27, 9 (2015), 2522–2535.

[4] Mustafa Gokce Baydogan, George Runger, and Eugene Tuv. 2013. A bag-of-features framework to classify time series. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 11 (2013), 2796–2802.

[5] BOSS implementation. 2016. https://github.com/patrickzib/SFA/. (2016).

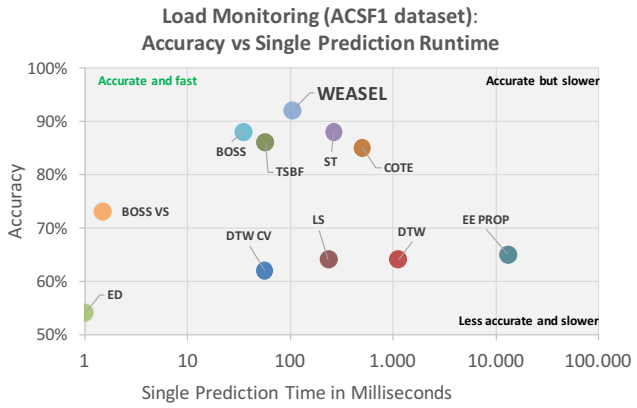
[6] Aaron Bostrom and Anthony Bagnall. 2015. Binary shapelet transform for multiclass time series classification. In *International Conference on Big Data Analytics and Knowledge Discovery*. Springer, 257–269.

[7] Wlodzimierz Bryc. 2012. *The normal distribution: characterizations with applications*. Vol. 100. Springer Science & Business Media.

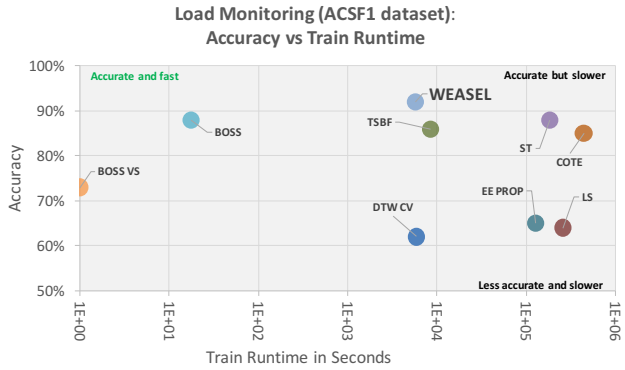
[8] G. Webb C. Tan and F. Petitjean. 2017. Indexing and classifying gigabytes of time series under time warping. In *SIAM SDM*.

[9] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7 (2006), 1–30.

[10] Philippe Esling and Carlos Agon. 2012. Time-series data mining. *ACM Computing Surveys* 45, 1 (2012), 12:1–12:34.



(a) Accuracy vs prediction time for ACS-F1 dataset.



(b) Accuracy vs train time for ACS-F1 dataset.

Figure 13: Prediction times on the ACS-F1 dataset.

[11] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research* 9 (2008), 1871–1874.

[12] Jingkun Gao, Suman Giri, Emre Can Kara, and Mario Bergés. 2014. PLAID: a public dataset of high-resoultion electrical appliance measurements for load identification research: demo abstract. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*. ACM, 198–199.

[13] Christophe Gisler, Antonio Ridi, Damien Zujferey, O Abou Khaled, and Jean Hennebert. 2013. Appliance consumption signature database and recognition test protocols. In *International Workshop on Systems, Signal Processing and their Applications (WoSSPA)*. IEEE, 336–341.

[14] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. 2014. Learning time-series shapelets. In *Proceedings of the 2014 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 392–401.

[15] Benjamin F Hobbs, Suradet Jitprapaikulsarn, Sreenivas Konda, Vira Chankong, Kenneth A Loparo, and Dominic J Maratukulam. 1999. Analysis of the value for unit commitment of improved load forecasts. *IEEE Transactions on Power Systems* 14, 4 (1999), 1342–1348.

[16] Bing Hu, Yanping Chen, and Eamonn Keogh. 2013. Time Series Classification under More Realistic Assumptions. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 578–586.

[17] Zbigniew Jerzak and Holger Ziekow. 2014. The DEBS 2014 Grand Challenge. In *Proceedings of the 2014 ACM International Conference on Distributed Event-based Systems*. ACM, 266–269.

[18] Isak Karlsson, Panagiotis Papapetrou, and Henrik Boström. 2016. Generalized random shapelet forests. *Data Mining and Knowledge Discovery* 30, 5 (2016), 1053–1085.

[19] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. 2003. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD '03)*. ACM, New York, NY, USA, 2–11.

[20] Jessica Lin, Eamonn J. Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and knowledge discovery* 15, 2 (2007), 107–144.

[21] Jessica Lin, Rohan Khade, and Yuan Li. 2012. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems* 39, 2 (2012), 287–315.

[22] Jason Lines and Anthony Bagnall. 2014. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery* 29, 3 (2014), 565–592.

[23] Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall. 2012. A shapelet transform for time series classification. In *Proceedings of the 2012 ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 289–297.

[24] Jason Lines, Sarah Taylor, and Anthony Bagnall. 2016. HIVE-COTE: The Hierarchical Vote Collective of Transformation-based Ensembles for Time Series Classification. In *IEEE ICDM 2016 conference*.

[25] Richard Lowry. 2014. *Concepts and applications of inferential statistics*. http://vassarstats.net/textbook/ch14pt1.html. Chapter 14 pages.

[26] Abdullah Mueen, Eamonn J. Keogh, and Neal Young. 2011. Logical-shapelets: an expressive primitive for time series classification. In *Proceedings of the 2011 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1154–1162.

[27] Christopher Mutschler, Holger Ziekow, and Zbigniew Jerzak. 2013. The DEBS 2013 grand challenge. In *Proceedings of the 2013 ACM International Conference on Distributed Event-based Systems*. ACM, 289–294.

[28] Rodica Neamtu, Ramoza Ahsan, and Elke Rundensteiner. 2016. Interactive Time Series Exploration Powered by the Marriage of Similarity Distances. In *Proceedings of the 2008 VLDB Endowment*, Vol. 10. VLDB Endowment, 169–180.

[29] Andrew Y Ng. 2004. Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In *Proceedings of the 2004 ACM International Conference on Machine Learning*. ACM, 78.

[30] Pavlos Protopapas, JM Giammarco, L Faccioli, MF Struble, Rahul Dave, and Charles Alcock. 2006. Finding outlier light curves in catalogues of periodic variable stars. *Monthly Notices of the Royal Astronomical Society* 369, 2 (2006), 677–696.

[31] J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning* 1, 1 (1986), 81–106.

[32] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 2012 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 262–270.

[33] Thanawin Rakthanmanon and Eamonn Keogh. 2013. Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM.

[34] Antonio Ridi, Christophe Gisler, and Jean Hennebert. 2013. Automatic identification of electrical appliances using smart plugs. In *International workshop on Systems, signal processing and their applications (WoSSPA)*. IEEE, 301–305.

[35] Patrick Schäfer. 2015. Scalable time series classification. *Data Mining and Knowledge Discovery* (2015), 1–26.

[36] Patrick Schäfer. 2015. The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery* 29, 6 (2015), 1505–1530.

[37] Patrick Schäfer and Mikael Högqvist. 2012. SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 2012 International Conference on Extending Database Technology*. ACM, 516–527.

[38] Pavel Senin and Sergey Malinchik. 2013. SAX-VSM: Interpretable time series classification using SAX and vector space model. In *Proceedings of the 2013 IEEE International Conference on Data Mining*. IEEE, 1175–1180.

[39] The Value of Wind Power Forecasting. 2016. http://www.nrel.gov/docs/fy11osti/50814.pdf. (2016).

[40] UCR and UEA Time Series Classification Repository. 2016. http://timeseriesclassification.com. (2016).

[41] Zhiguang Wang, Weizhong Yan, and Tim Oates. 2016. Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline. *arXiv preprint arXiv:1611.06455* (2016).

[42] Weasel Classifier Source Code and Raw Results. 2016. https://www2.informatik.hu-berlin.de/~schaefpa/weasel/. (2016).

[43] Y Chen, E Keogh, B Hu, N Begum, A Bagnall, A Mueen and G Batista . 2015. The UCR Time Series Classification Archive. http://www.cs.ucr.edu/~eamonn/time_series_data. (2015).