Visualizing Feature-Sets in Time Series Classification

Nicolai Schneider

Summer Term 2019

Contents

1 Introduction

One very exciting field that came along with a lot of potential for AI and machine learning was the smart watch: during a study, researchers from the University of California in San Francisco (UCSF) used a deep neural network, created by the company Cardiogram, to detect early signs of diabetes in a patient's heart rate and step count (both were measured through smartwatches). In case of a detection, the device would alert the user to check in with a doctor. During their study the researchers reached a success rate of 85% [1]. Eventhough an algorithm has to be trained to forecast prediabetes (in the given example) the rising question is about the interpretability of the results: e.g. which aspects of the analyzed data are responsible for the final forecast? Approaching this research question by use of visualization will be the thesis' topic.
Two important concepts are that of *time series* (TS) and that of *time series classification* (TSC). A TS is a series of values ordered in time, in our case, the given TS represents the measured heart rate and step count. In [2] the TSC problem is defined as: "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." TSC is intuitively depicted in Figure 1, which shows a person's hands movement while drawing a gun. The TSC problem consists of two classes: a person holding a gun or pointing its finger during the action. The colorization describes how much the final categorization was affected by each measurement. For instance, the radical incline (dark red) seems to be defining for some movements, while the plateau (blue) is less considered [3].
TSC methods can be subdivided into the following categories [4]:

1. *Whole Series.* A point-wise comparison of entire time series which are often realized through 1-Nearest Neighbour (1-NN) Euclidian Distance and 1-NN Dynamic Time Wrapping. Due to their high time complexity (Dynamic Time Wrapping runs in $O(n^2)$ at a time series length n) whole-series methods do not scale to longer TS. [2]

2. *Intervals.* Features over one (or more) random subsequences at fixed offsets[1] are selected from the TS to then be used as input for classification.

3. *Shapelets.* Approaches in this group identify short subsequences of TS that are representative for the entire class through a learning procedure. These subsequences are called Shapelets and indicate the class by their presence or absence which is measured through a distance function.

4. *Bag-Of-Patterns.* A class of methods that breaks the TS into a bag of subsequences which are "translated" into discrete words. The similarity of two TS is measured by their word frequency concordance.

5. *Deep Learning.* Here a deep neural network is built to learn a hierarchical representation of the data. For each classification it returns a probability distribution of the possible classes. Approaches in this category typically do not rely on manual feature extraction (as part of preprocessing) but the classifier rather learns these as part of the classification process. [3]

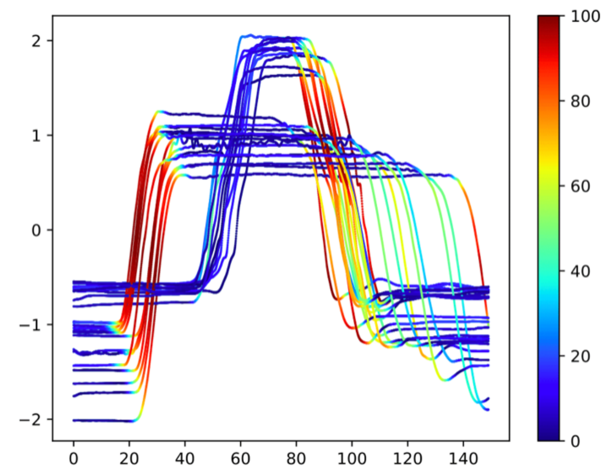

Figure 1: The gun point dataset shows the hand's y-position while drawing either a gun or just imitating the movement with bear hand. The color represents the contribution of a value to the classification result: blue meaning almost no importance while red representing a high contribution. The radical incline at the start visualizes the hand movement to shoulder level, while the blue plateau represents the action of steady pointing. (image taken from [3])

For this thesis we will be focusing on Bag-of-Patterns and Shapelets. *WEASEL* (Word ExtrAction for time SEries cLassification), a TSC model that uses linear regression to classify a TS [2], will be used as the representative for the Bag-of-Patterns approach, while *FSS* (Fast Shapelet Selection) [5] could be a candidate for a Shapelet-based approach. This thesis will focus on visualizing the outcome of both models to comparatively highlight which segments of a TS were crucial for classification (for each algorithm).

2 Background and Related Work

Next, we will present the state-of-the-art methods that will be used throughout the thesis.

2.1 Symbolic Fourier Approximation

SFA (Symbolic Fourier Approximation), as part of the BOSS transformation, transforms a subsequence of length w of real-valued measurements into discrete words. The final result is achieved through the following steps: [2]

1. The values of each sequence are normalized to have a standart deviation of 1.
2. SFA applies a truncated Fourier transformation on each window only keeping $l < w$ Fourier Coefficients to low-pass filter the TS and eliminate noise (top right image in Figure 2).

[1] Definition: a sequence inside a time series bounded by an interval.

3. Breakpoints are learned from the data distribution using e.g. binning based on information gain. Using these breakpoints the features are discretized into a symbol of a predefined alphabet for further noise cancellation (bottom images in Figure 2) thus building a word for each window.

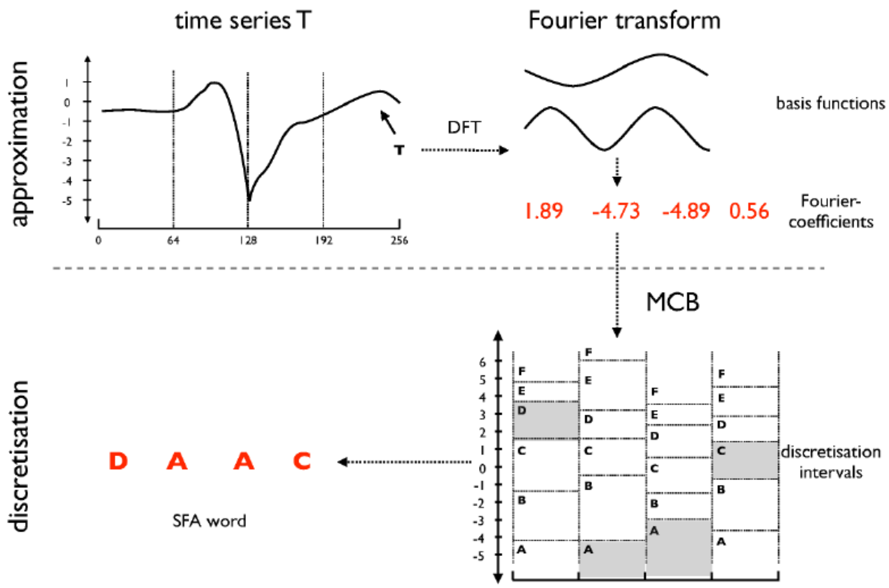

Figure 2: SFA decomposes all TS into a sum of basic functions which are represented by Fourier Coefficients. Discretization breakpoints are learned using the *multiple coefficient binning* (MCB) method which creates the discretization boundaries for each basis function (Fourier coefficient) independently. Finally, based on those boundaries, SFA generates a symbol for each basic function. [6]

2.2 WEASEL

WEASEL applies a sliding window approach and discretizes each subsequence into a word using SFA. Nevertheless, depending on the window size, the applied discretization can lead to overshadowing smaller, but still crucial, characteristics. Therefore, instead of extracting subsequences of one fixed window length w, WEASEL repeatedly slides through the TS, each time with a different w. Each window undergoes the SFA transformation and the computed feature is concatenated with the window length. In addition, the feature frequency is counted with every iteration.
With progressing window, bigrams are computed, then appended to a bag-of-patterns (Figure 3). Once all defined window lengths have been processed, WEASEL applies the CHI-squared test on the final bag-of-patterns. This test filters irrelevant words such as a word frequently occurring among all classes.
Finally, the linear classifier assigns each word a weight component to define its importance for

separating classes. These weights are trained on labeled train samples. Linear classifiers use the dot-product of the feature vector and its correspondent weight vector for their prediction. [2]

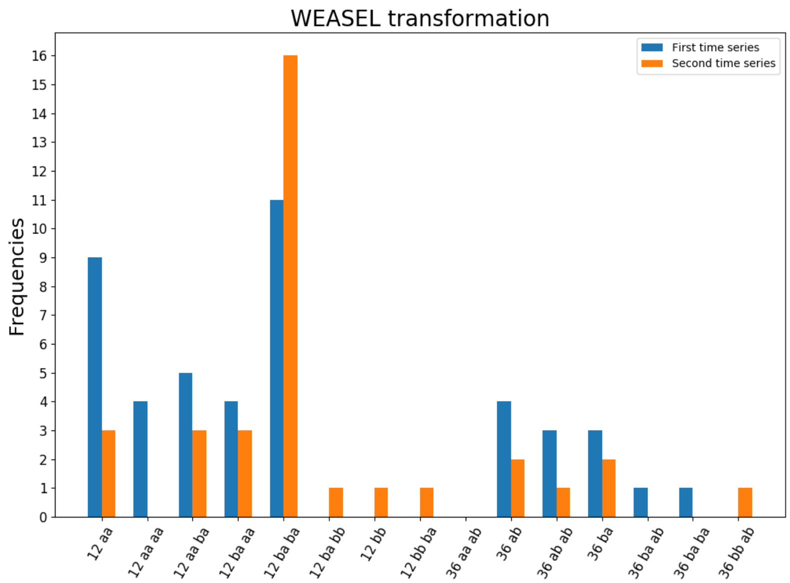

Figure 3: A WEASEL histogram showing frequencies of words. The number on the x-axis prior to the word describes the window length. (taken from [7])

2.3 Shapelets

The Shapelet-approach is based on the idea of identifying the most discriminative subsequences (Shapelets) for each class and computing their smallest distance to a time series. Based on the obtained distance and a learned threshold, a TS can be assigned to a class.
First, Shapelets need to be identified during a brute force learning procedure: all subsequences of all possible lengths are stored in a list and inspected in terms of how well the distance to the Shapelet separates the samples of the classes. After iterating through the entire list the Shapelet with the highest discriminative power is returned.
One Shapelet-approach is based on a decision tree as shown in Figure 4. Each node stores a single shapelet and a distance threshold used for branching to the left or right subtree classifier while the leaf nodes contain the predicted class. Starting from the root we compute the distance from a given TS to the Shapelet in that node. A split point reveals whether to branch to the right or left subtree depending on the distance. This procedure is recursively repeated until the algorithm reaches a leaf node. [8]
Figure 4 shows a dataset with two types of leaves: *Verbena urticifolia* and *Urtica dioica*. Since the most discriminative feature is the angle between stem and leaf the algorithm learns that subsection as a Shapelet. The resulting decision tree predicts *Verbena urticifolia* for all TS with a distance no bigger than 5.1 and *Urtica dioica* for the rest.

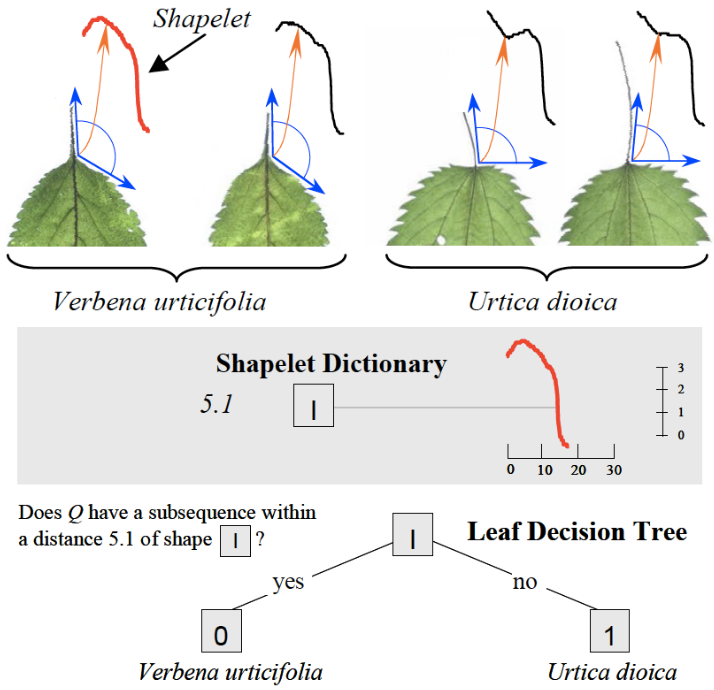

Figure 4: The Figure displays two types of leaves and the subsequence chosen to be the Shapelet. On the bottom we see the correspondent decision tree. (image taken from [8])

3 Aims and Objectives

The main motivation of this thesis is the analysis of the interpretability of TSC models based on features and feature weighing using visualization. First, we extract data from WEASEL's model. This data contains the values of the selected TS, the feature weights assigned by liblinear and the features for each window size.
The goal is to build a visualization tool to understand which segments of a TS are crucial for the final classification. There have been approaches in this area, a commonly used visualization is the heat map (as also used in [3]). Since the classification decision is created by a (black-box) classifier it is not always obvious or easy to interpret the outcome. Considering that SFA is in concept similar to a discretized Short-time Fourier transform (STFT) [9], we can use methods for visualizing STFTs such as spectrograms of WEASEL's words. We will also implement a heat map.
The heat map will map the values and their weights through color and line thickness. Finally, we will study several examples to interpret the visual outcome. E.g. Figure 5 shows how a heat map of a gunpoint TS could look like. The gunpoint dataset measures a person's hand position while either drawing a gun or imitating the movement with the hand. The little bump in the bottom left, just before the radical incline (Figure 5), indicates that the person was holding a gun, since the gun has to be taken out of the holster. WEASEL considered the bump as important indicated by the red color, which means that the feature weight was high in that area.

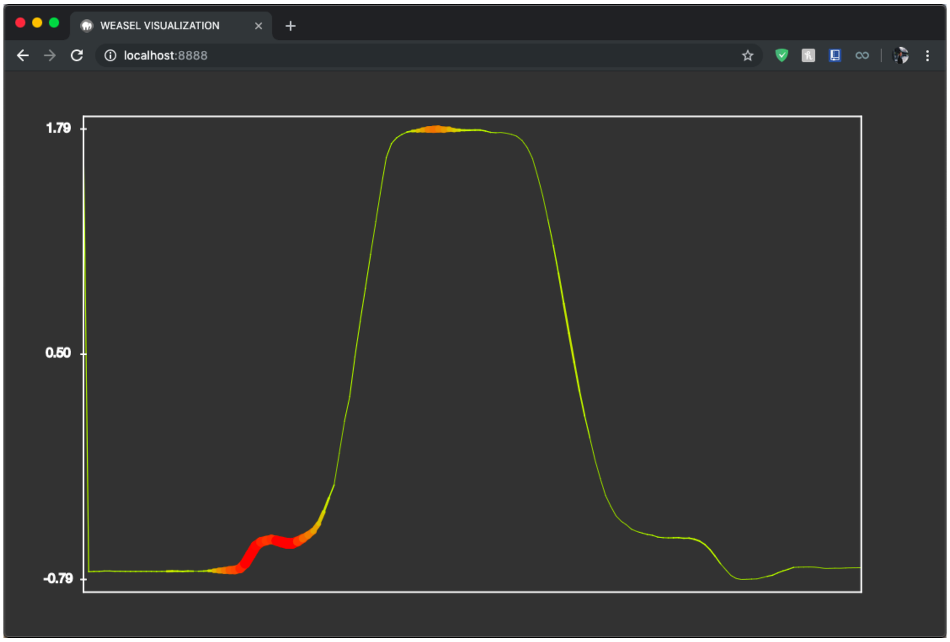

Figure 5: An example of what the tool could look like. The heat map shows a sample from the GunPoint dataset computed in WEASEL, specifically the action of drawing a gun.

Since each dataset contains many different samples we will enable the user to scroll through all the available ones. A specific sample can also be selected from a collection view where each TS could be pictured as a bitmap (bitmap-method described in [10]). Lastly, to showcase how the heatmap of each label approximately looks like, we will compute an average over all samples (of the same label).
For comparison to WEASEL we will use a Shapelet-model such as FSS [5] or FS [11] since the identification of Shapelets using brute force has a cubic time complexity in terms of TS lengths [8]. We have to access the Shapelets and their distance to each TS which we consider like the weight of a feature (learned by WEASEL). Therefore, the model can be implemented using the same visualization.
Additionally, we will implement a spectrogram, similar to the ones used in STFT, to visualize features. Our approach is depicted in Figure 6.

Furthermore, we plan to identify frequent occurrences of subwords within the same basis function over time to detect if certain patterns appear more often than others. Frequent subwords could for example be identified using a suffix tree. These patterns can be illustrated inside the spectrogram and the heatmap (see the thin blue line in Figure 6 right).
Based on the final software we will analyze a few examples to (i) examine if we can rely on the visualized outcome based on previous literature, (ii) get a better understanding of why WEASEL perceives certain subsequences as crucial to the classification and compare it to a Shapelet-based model and last (iii) further study the STFTs to find helpful patterns.
Based on the available information we will use the p5.js library to implement a visualization tool. The project will be available on Github for everybody accessible.

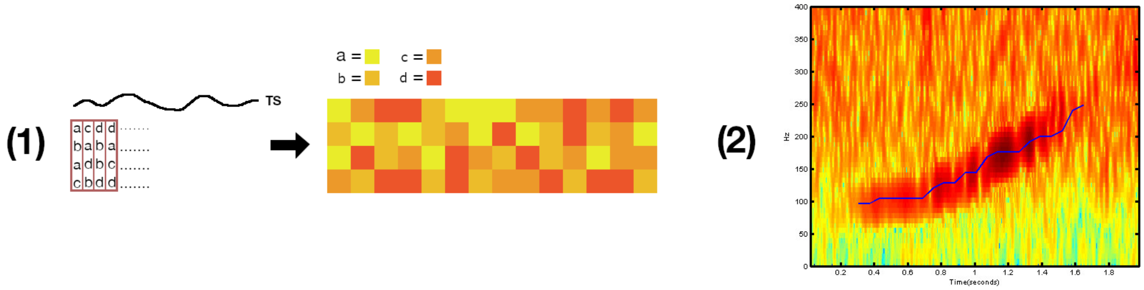

Figure 6: Image on the left (1) shows how we can generate a spectrogram: let the word length be 4 and the used alphabet $A = \{a, b, c, d\}$. Each symbol is assigned to an individual color, for example on a color spectrum from yellow to red (e.g. a = yellow, b = light orange, c = dark orange, d = red). If we vertically align all words in sequential order next to each other we get a spectogram. Evidently, the features need to be of one fixed window length. The final result could look like image (2) (taken from [12]).

References

[1] Eric Wicklund. mhealth wearables, ai used to detect diabetes in one's heart rate. `https://mhealthintelligence.com/news/mhealth-wearables-ai-used-to-detect-diabetes-in-ones-heart-rate`, 2018.

[2] Patrick Schäfer and Ulf Leser. Fast and accurate time series classification with WEASEL. *CoRR*, abs/1701.07681, 2017.

[3] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *CoRR*, abs/1809.04356, 2018.

[4] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, May 2017.

[5] Cun Ji, Chao Zhao, Shijun Liu, Chenglei Yang, Pan Li, Lei Wu, and Xiangxu Meng. A fast shapelet selection algorithm for time series classification. *Computer Networks*, 148, 01 2019.

[6] Patrick Schäfer and Mikael Högqvist. Sfa: A symbolic fourier approximation and index for similarity search in high dimensional datasets. pages 516 – 527, 03 2012.

[7] Johann Faouzi. Word extraction for time series classification (weasel). `https://pyts.readthedocs.io/en/latest/auto_examples/transformation/plot_weasel.html#sphx-glr-auto-examples-transformation-plot-weasel-py`, 2019.

[8] Lexiang Ye and Eamonn J. Keogh. Time series shapelets: a new primitive for data mining. pages 947–956, 06 2009.

[9] Ervin Sejdic, Igor Djurovic, and Jin Jiang. Time–frequency feature representation using energy concentration: An overview of recent advances. *Digital Signal Processing*, 19:153–183, 01 2009.

[10] Nitin Kumar, Venkata Nishanth, Lolla Eamonn, Keogh Stefano, Lonardi Chotirat, Ann Ratanamahatana, and Li Wei. Time-series bitmaps: A practical visualization tool for working with large time series databases. In *In proceedings of SIAM International Conference on Data Mining (SDM '05*, 2005.

[11] Thanawin Rakthanmanon and Eamonn Keogh. *Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets*, pages 668–676. 05 2013.

[12] Daniel Nouri. Using deep learning to listen for whales. `http://danielnouri.org/notes/2014/01/10/using-deep-learning-to-listen-for-whales/`, 2014.