Time Series: Defining a Search Engine Algorithm

Philipp Beer

A Dissertation

Presented to the Faculty

of University of Nicosia

in Candidacy for the Degree

of Master of Science

Recommended for Acceptance

by the Department of

Computer Science

February, 2022

Abstract

This is abstract.

Acknowledgements

These are the acknowledgments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Time series is often described as "anything that is observed sequentially over time" which usually are observed at regular intervals of time [14]. They can be described as collection of observations that are considered together in chronological order rather than as individual values or a multiset of values. Their representation can be described as ordered pairs: $S = (s_1, s_2, \ldots, s_n)$ where $s_n = (t_n, v_n)$. t_n can be a date, timestamp or any other element that defines order. v_i represents the observation at that position in the time series.

Time series are utilized to analyze and gain insight from historic events/patterns with respect to the observational variable(s) and their interactions. A second area of application is forecasting. Here time series are utilized to predict the observations that occur in future under the assumption that the historic information can provide insight into the behavior of the observed variables.

Fu in his work categorized time series research into (1) representation, (2) indexing, (3) similarity measure, (4) segmentation, (5) visualization and (6) mining [9]. Research in these different fields started taking off in the second half of the 20^{th} century. For example in [41] the authors worked on questions of representation via sampling the sampling of time series in 1969. All these different research areas always have to deal with the challenges that inhibit time series data. Generally data sets in this domain are large. Through this time series data incorporates the similar obstacles as high dimensional data, namely the "curse of dimensionality" [36] and requiring large computational efforts in order to generate insights. And as will be discussed in 1.1 there are applications fields

where vast amounts of time series are generated and a comparison between them is required.

In this thesis we will focus on creating an algorithm which allows the fast and meaningful comparison of an input time series or template against a database of time series. This is an important field of research for many different reason. In part, because time series are omnipresent, they are growing in volume and length, and the permeate many areas of science. Lots of work in this field has already been done. However, as we will see in 2 in application there is a tendency to revert to simple methods that accomplish the desired results. For time series those are mainly Euclidean Distance and Dynamic Time Warping. This is a good approach for as long as those methods achieve that and are feasible to execute on real-world data scenarios. We will see that this is not the case for both mentioned categories. This is were our proposed algorithm steps in and aims to provide comparable or subjectively better results with significantly less computational effort.

We will present an algorithm that intends to balance the user requirements of a search engine, namely fast and accurate results that are ranked. Additionally it should be flexible to accommodate varying requirements in the type of similarity that the user wishes to achieve.

1.1 Applications

Before going into further detail we want to explore where we encounter time series in more detail. The short answer is everywhere. Any metric that is captured over time can utilized as time series. Granularity can be used as descriptor for the sampling rate of a series or more general how often measurements for a particular metric are taken. This granularity has a tendency to increase as well. As example consumer electronics that capture health and fitness data can be mentioned. Or sensors which are utilized in the automotive industry or heavy machine industry where they are employed to capture information for predictive maintenance applications. That usually requires a high sampling rate for each device generating data. And for such equipment many data generating devices are normal. Having fast methods of comparing time series can be of great service in quickly detecting deviations, possible change points or anomalies. This is turn can have a very positive impact of maintenance times and possibly help with error prevention.

In the financial industry time series are a fundamental component of decision making, for exam-

ple the development of stock prices over time or financial metrics of interest. The same is true for macro economic information or metrics concerning social structures in society, etc. Faster analytics processes can be a value driver in this industry.

In the medical field time series are also ubiquitous. Whether they relate to patient date like blood pressure. The bio statistics field utilizes electro-graphical data like electrocardiography, electroencephalography and many others. In more aggregate medical analysis like toxicology analysis of drug treatments for vaccine approvals they are utilized and in many forms of risk management, for example, population level obesity levels. A search engine identifying in real time diagnosis based on comparing time series can be of great value to research in this field and to patient outcomes.

In engineering fields the utilization of time series is often times similar to the above. However, it also requires that information that is captured in time series is transferred between locations in an efficient manner. For example voice calls are required to be transferred between the participants in a fast manner and with minimized levels of noise in the data. Another interesting industrial example in the biomedical technology field is Neuralink which aims to implement a brain-machine-interface (BMI) utilizing mobile hardware like smartphones as the basis for its computation. Here a large of amount of time series data is generated which requires quick processing to generate real-time information. **(author?)** describes a recording system with 3072 electrodes generating time series data [28] that is used to capture the brain information and visualized in real-time [34]. Any improvement in the analytics of such time series can help with costs or even enable completely new application areas.

Time series data is paramount to a wide variety of areas, relating to many different fields. Looking at the trajectory it seems likely that going forward more time series data on a higher granularity will be generated. This in turn increases the need to be able process, analyze, compare and respond to the data with methods that are faster than today's standard options.

1.2 Organization of this thesis

In the remainder of this thesis we The rest of this thesis is organized as follows. We start by reviewing existing work (2). Next, we review the required theoretical underpinnings required for our method (3). With this we will introduce our proposed algorithm (4). This is followed by an analysis of the data used for the developing the algorithm and for the formal evaluation (5). We introduce the M4 competition data as well as the UCR Time Series classification archive. Thereafter, the formal results are presented (6) and we close by discussing our results and their implications.

Chapter 2

Related work

Related work addressing the idea of time series search engine focuses often on the system architecture and the data processing and pipelining aspect of such a system and the generally architecture [39]. In other research Keogh et all. applied a dimensionality reduction technique (Piecewise Constant Approximation) to execute fast search similarity search in large time series databases [19]. This is an approach with similar considerations to our approach. Other papers address domain specific questions like the introduction of a "Time-series Subimage Search Engine for archived astronomical data" [17].

Before we can describe what a search engine is supposed to evaluate we need to introduce the notion of similarity in time series. A measure for similarity is required. In the literature various general measures and corresponding computation methods can be found. Wang et al. reviewed time series measures and categorized the similarity measures into 4 categories: (1) lock-step measures, (2) elastic measures, (3) threshold-based measures, and (4) pattern-based measures [37]. Other authors like Zhang et al. classify similarity measures in the categories: (1) time-rigid methods (Euclidean Distance), (2) time-flexible measures (dynamic time-warping), (3) feature-based measures (Fourier coefficients), and (4) model-based methods (auto-regression and moving average model) [40]. The different categories focus on different aspect on expressing similarity between time series. For example, Lock-step measures include the L_p-norms (Manhattan and Euclidean Distance) as well as Dissimilarity Measure (DISSIM). Elastic measures include metrics like Dynamic Time Warping

(DTW) and edit distance is based on measures like Longest Common Subsequence (LCSS), Edit Sequence on Real Sequence (EDR), Swale and Edit Distance with Real Penalty. The threshold-based measures are threshold query based similarity search (TQuEST). And Spatial Assembling Distance (SpADe) is an example for pattern-based measures. In another paper, Gharghabi et al. classify the space of similarity measures by the most common measures into: (1) Euclidean Distance, (2) Dynamic Time Warping (DTW), (3) Least Common Subsequence (LCSS), and (4) K-Shape [10].

Especially when the focus is not identifying novel metrics for similarity in time series a tendency of reverting to simple and straightforward methods can be found. A highly popular metric among the elastic measures is Dynamic Time Warping (DTW). Elastic in this case means that it is flexible in its comparison of points. This is in opposition to Euclidean Distance (ED) where each point is compared with its counterpart on the time series of comparison. DTW has been introduced by cite Berndt and Clifford in 1994 and its key advantage is the fact that comparison is applied on a one-to-many-basis allowing the comparison of regions from one series to regions of the other time series [3]. This gives it the capability to warp peaks or valleys between different time steps of the two series as the resulting distance metric. This is a very powerful method of finding similar data components even at an offset. We will show in section 3.2 this comes at the price of time complexity which renders it effectively useless in practice when applied to large scale data sets.

Other attempts are also made in introducing new distance metrics. Gharghabi et al. introduced a new metric called MPdist (Matrix Profile Distance) which is more robust than Euclidean Distance (ED) - more details can be found in section 3.1 - and Dynamic Time Warping (DTW) - more details can be found in section 3.2 - and computationally preferable [10]. Interestingly, due to the use of subsequences in the comparison of two time series its time complexity ranges from $\mathcal{O}(n^2)$ in the worst case, to $\mathcal{O}(n)$ in the best case and with this can provide a significant advantage of prevalent methods like ED or DTW.

The other research area of interest for our task is time series representation. It concerns itself with the optimal combination of reduction of the data dimensionality but adequate capture of its particular properties. With these methods feats like minimizing noise, managing outliers can be achieved. For many activities this is also the basis for the reduction of time complexity in the re-

sulting algorithms that analyze and compare the time series. This is relevant to our work as we use a Fourier transformed representation of the time series for the first step of identifying similarity. We will show that some aspects of the underlying time series transfer into the Fourier-based representation of our time series allowing for reviewing similarity under the transform.

According to Li et al. the following methods are common methods for this task: (1) Discrete Fourier Transformation (DFT), (2) Singular Value Decomposition (SVD), (3) Discrete Wavelet Transformation (DWT), (4) Piecewise Aggregate Approximation (PAA), (5) Adaptive Piecewise Constant Approximation (APCA), (6) Chebyshev polynomials (CHEB), (7) Symbolic Aggregate approXimation, and others [21]. In their paper, Pang et al. mention (1) Singular Value Decomposition (SVD), (2) Frequency-Domain transformation, (3) Piecewise Linear Representation (PLR), (4) model-based method, and (5) symbolic representation [29] as possible representation alternatives.

2.1 Dimensionality Reduction related to Singular Value Decomposition

Singular Value Decomposition is a fundamental matrix factorization technique with a plethora of applications and use cases. As our Fourier-transform is a counterpart to the SVD we take a brief excurse into the most important aspects and advantages of the SVD. It's value comes from the capability of generating low rank approximations of data matrices that allow to represent the matrix values via the unitary matrices $\boldsymbol{U} \in \mathbb{C}^{n \times n}$ and $\boldsymbol{V} \in \mathbb{C}^{m \times m}$. The columns in $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal. The remaining matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times m}$, is a diagonal matrix with non-negative entries.

The power of the SVD is its ability to provide a low-dimensional approximation to high-dimensional data [4]. High dimensional data is often determined by a few dominant patterns which can be described by a low-dimensional attractor. Therefore, a prime application for the SVD is dimensionality reduction. It is complementary to the Fast Fourier Transform (FFT) which lays at the core of this work. Brunton and Kutz describe it as the generalization of the FFT.

Principal Component Analysis (PCA) is a very common application of the SVD. It was developed by Pearson in 1901 [30]. The main idea of PCA is to apply the SVD to a data set centered

around zero and subsequently computing the covariance of the centered data set. Through the computation of the eigenvalues and their identifying the largest values the most important principal components are identified. Those are responsible for the largest variance in the data set. And similar to the SVD their ranking and subsequent filtering can be used to focus on the most important components that allow to recreate majority of the of the variance in the data set.

The Fast Fourier Transform (FFT) is based upon the Fourier Transform introduced by Joseph Fourier in early 19th century to analyze and analytically represent heat transfer in solid objects [8]. This transform is a fundamental component of modern computing and science in general. Its significance cannot be overstated. It has transformed how technology can be used in the in 20th century in areas such as image and audio compression and data transfer. In quantum physics the Fourier transform is the underlying method for changing the basis when describing the position or the momentum of a particle. The concept will be introduced in more detail in section 3.3. Its core idea is to represent the data to be transformed as the coefficients of a basis of sine and cosine eigenfunctions. It is similar to the principles of the SVD with the notable difference that the basis are an infinite sum of sine and cosine functions. The ability to reduce the transformed data to few key components is the same as in SVD and PCA and one of the fundamental properties we exploit in our algorithm.

2.2 Symbolic Aggregate approXimation

A dimensionality reduction technique that does not built on SVD and is geared directly towards time series is the Symbolic Aggregate approXimation (SAX) algorithm. Its core idea is to transform a time series into a set of strings via piecewise aggregate approximation (PAA) and a conversion of the results via a lookup table [22]. Starting with PAA the reduction of a time series T of length n in vector $\bar{S} = \bar{s_1}, \bar{s_2}, \ldots, \bar{s_w}$ of length w where $w < n$, can be achieved through the following computation:

$$\bar{s_i} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} s_j \tag{2.1}$$

This simply computes the mean of each of sub sequences determined through parameter w. An

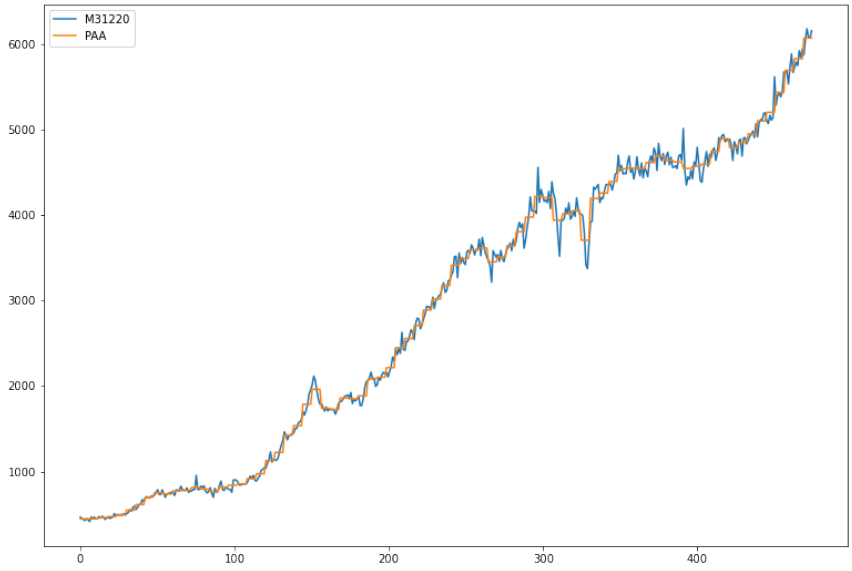

Figure 2.1: Piecewise Aggregate Approximation - M4 example: M31220 (window size - 6)

example from the M4 data set can be seen in figure 2.1. For its application in SAX the time series are standardized or mean normalized, so that the comparison happens on the same amplitude. From this representation the data is further transformed to obtain a discrete representation via the mapping of the values computed via PAA to a symbolic representation of a letter. The used discretization should accomplish equiprobability in the assignments of the symbols [23]. The authors show by example of taking subsequences of length 128 from 8 different time series that the resulting PAA transformation has a Gaussian distribution. This property does not hold for all series. And in place where it does not hold the algorithm performance deteriorates. If the assumption that the data distribution is Gaussian is true, breakpoints that will produce equal-sized areas can be obtained from a statistical table. The breakpoints are defined as $B = \beta_1, \beta_2, \ldots, \beta_{a-1}$ so that the area under a Gaussian curve $N(0,1)$ from β_{i} to $\beta_{i+1} = \frac{1}{a}$ (β_0 and β_{a} are defined as -inf and inf) [23]. Table 2.1 shows the value ranges for values of a from 3 to 10 and has been reproduced from [23].

Based on into which β category a value of PAA fits a symbol is assigned. "**a**" is reserved for values smaller than β_1 and values exceeding $\beta_{\text{a-1}}$ is assigned the last symbolic value which differs

Table 2.1: Lookup table - reproduced from [23]

β_i	3	4	5	6	7	8	9	10
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.29
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_4			0.84	0.43	0.18	0	-0.14	-0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

depending on how many categories are chosen.

As stated before, this method relies on the fact that the data is normally distributed. Therefore, it can be great to detect for example anomalies in streaming data. Also the distance computation is preserved on the PAA values. However, the distance computation is still based on Euclidean Distance (ED) and has the same time complexity as before, but for fewer data points compared to the original series.

Chapter 3

Theoretical background

In this section we explore the different underlying concepts relevant to our algorithm. We start by introducing Euclidean Distance (3.1) and Dynamic Time Warping (3.2), discuss how they work and what limitations they face. The last section in this chapter introduces the Fast Fourier Transform (3.3). It provides an overview on how it is derived and which properties are most important to our algorithm. We close this chapter with a section on window or apodization functions and their relevance for Fourier transforms.

3.1 Euclidean Distance

Euclidean Distance (ED) is the most widely used distance metric in the research of time series. It is either used as a metric on its own or a as metric used inside other methods to compute distances, for example, the computation of distances of subsections of time series data ([7]) or to compute the distance between various points of two time series (see section 3.2). Having two time series $S = \{s_1, s_2, \ldots, s_n\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$ both of length n the Euclidean distance can be computed as:

$$D(S, Q) = \sqrt{\sum_{i=1}^{n} (S_i, Q_i)^2} \tag{3.1}$$

It is a measure that is easy to compute and comprehend and gives intuitive input for the distance and hence similarity of two time series. If there multiple series involved for comparison the resulting

distances can be used for ranking or clustering the results. From the standpoint of time complexity the algorithm is applicable also to larger data sets with $\mathcal{O}(n)$. Its simplicity also creates some limitations for real-world scenarios. For example, to compute the Euclidean Distance between two series their length needs to be the same. Consider, having two time series with differing lengths. The maximum points that can be compared are determined by the shorter time series. This would lead to the obscure situation that the distance for some of points of the longer series are not computed at all and hence would not be considered when trying to compare and rank the similarity between multiple series. Furthermore, ED can be easily impacted in its results by the presence of outliers or increased levels of noise. Depending on the magnitude of the outlier, that single distance measure may overshadow the remainder of the series. In addition, it is not elastic with respect to the warping of information between two series in which effects that could indicate similarity happen even at slightly disparate steps.

Despite Euclidean Distance limitations it is a prominent metric and widely used for distance calculations for time series that can abide by its constraints and are not impacted by its short comings. Some of its limitations are addressed by more sophisticated metrics that utilize ED as component in a more sophisticated approach. We discuss the popular elastic approach of Dynamic Time Warping next.

3.2 Dynamic Time Warping

Berndt and Clifford introduced the Dynamic Time Warping algorithm in 1994. It reveals the minimized alignment between two time series computed through a cost matrix and identifying the minimal total path through the matrix starting from the final elements of each time series stopping at the first element in each series. This warps the points in time between the different series as shown in figure 3.1.

Two series $S = \{s_1, s_2, \ldots, s_n\}$ of length n and $Q = \{q_1, q_2, \ldots, q_m\}$ of length m are considered. For the series a n-by-m cost matrix M is constructed. Each element in the matrix represents the respective i$^{\text{th}}$ and j$^{\text{th}}$ element of each of the two series which contains the distance of those to

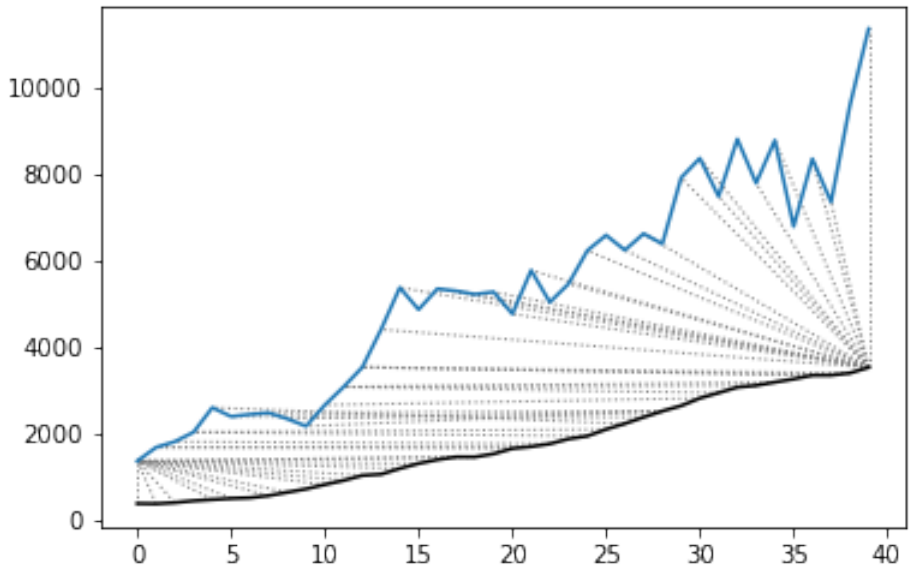

Figure 3.1: Dynamic Time Warping - M4 Example: Y5683 and Y5376

points:

$$m_{ij} = D(s_i, q_j). \tag{3.2}$$

Often time Euclidean Distance is used as distance function $D(s_i, q_j) = (s_i - q_j)^2$. From the matrix a warping path P is chosen, $P = p_1, p_2, \ldots, p_k, \ldots, p_K$ where:

$$\max(m, n) \leq k < m + n - 1 \tag{3.3}$$

The warping path is bound with the following condition $p_1 = (1, 1)$ and $p_K = (m, n)$. In consequence, both the first elements of each series, as well as, the last element of each series are bound to each other in the computation. The warping path also is continuous; from each chosen element p_k only the neighboring elements to the left, right and diagonally can be chosen for the continuation of the path: $p_k = (a, b)$ and $p_{k-1} = (a', b')$ with $a - a' <= 1$ and $b - b' <= 1$. The path elements p_k are also monotonous, meaning that $a - a' \geq 0$ and $b - b' \geq 0$. From the resulting matrix considering the mentioned constraints a cumulative distance $\gamma(i, j)$ is computed recursively:

$$\gamma(i, j) = D(s_i, q_j) + \min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\} \tag{3.4}$$

Therefore, the path can obtained by the following definition:

$$DTW(S,Q) = \min_{P:WarpingPath} \left\{ \sum_{k=1}^{K} \sqrt{p_k} \right\} \tag{3.5}$$

Figure 3.2 provides an example for a warping path result.

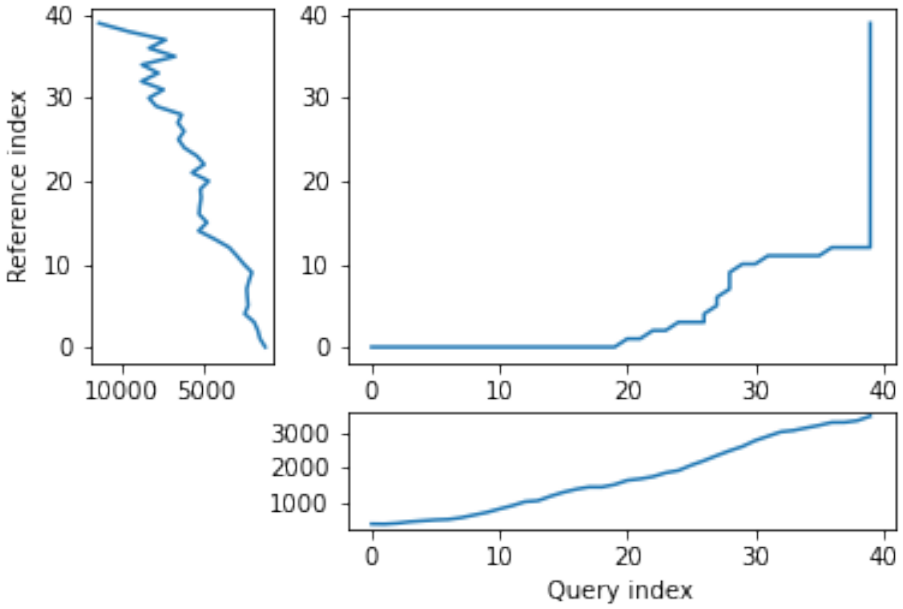

Figure 3.2: Warping path example - M4 data: Y5683 and Y5376

DTW therefore has an exhaustive search across the possible mapping space between two series and find the one that minimizes the total distance. It is a much in depth analysis compared to standard Euclidean distance allowing for distortions in the compared time series that are well captured by the warping of the matched data points. As will been seen in the section on the formal evaluation 6, the results of Dynamic Time Warping are optimal for finding the cumulative smallest possible Euclidean Distance between the data points of two time series. In favor of DTW needs to be stated, that it is flexible with regards to the time series used. The compared time series do not require to have the same length and can still be compared. This is a property that is not available with Euclidean Distance. However, the user also needs to be aware of outliers in either data set which can lead to a clustering of the warping path or pathological matches around those extreme points in the series.

The challenge the application of DTW brings is the time complexity of the algorithm: $\mathcal{O}(m*n)$ due to the fact that the distance computation needs to be executed for each combination of elements between both time series. Various methods for speed improvements have been introduced. A popular principle was described by Ratanamahatana et al. The researchers introduced an adjustment window condition where it is assumed that the optimal path does not drift very far from the diagonal of the cost matrix [32]. However, this does not change the fundamental nature of the algorithm's time complexity and computing DTW for multiple time series against a database of time series will require days of computation time even on modern computer architectures. Additionally, the method is not scale-invariant against the length of time series. It is an non-linear relationship that increases with the length of the series with $m * n$ or n^2 in case both time series have the same length.

In practice, Dynamic Time Warping is not a method suitable for comparing a single time series against a large array of series when speed is an important criterion as well as the handling of outliers in the data set. For our work it should not be considered for building a time series search engine algorithm due to time complexity in the comparison.

3.3 Fast Fourier Transform

In Fourier analysis the Fast Fourier Transform (FFT) is a more efficient implementation of the Discrete Fourier Transform (DFT) that utilizes specific advantageous properties of matrix computations. The DFT is based on the Fourier Transform (FT) which concerns itself with the representation of functions on a basis of sine and cosine functions. This is in turn derived from the Fourier series. We will give a brief introduction to them. A thorough introduction from which the following subsections heavily draw can be found in [4]. The principal idea Fourier analysis follows is that it can project (1) functions - via Fourier Transform - and (2) data vectors - via Discrete Fourier Transform - into a coordinate system defined by orthogonal functions - sine and cosine. To get the exact representation of a function or a data vector it has be done in infinitely many dimensions. We introduce the Fast Fourier Transform by showing the equivalence of the inner norm and the integral of their product (3.3.1). From this we build the 2π-periodic Fourier Series (3.3.2) and transfer it to a non-2π-periodic basis of length L and develop it further to the non-periodic Fourier Transform (3.3.3).

Next we make it applicable to discrete sets of data (3.3.4) via the Discrete Fourier Transform. Finally we introduce a computational trick that enables the dramatic time complexity reduction of the Fast Fourier Transform (3.3.5). We close off this section by discussing the important properties that make the FFT scale invariant with respect to the length of the time series to which it is applied (3.3.6), explain the Power Spectrum (3.3.7), the phenomenon of spectral leakage (3.3.8) and which methods exist to address it (3.3.9 and 3.3.10).

3.3.1 Inner Product of Functions and their norms

To get to the properties of data under the Fourier transform we must start with the Hermitian inner product ([33]) of functions in Hilbert spaces, $f(x)$ and $g(x)$ ($\overline{g}$ denotes the complex conjugate of g) in the domain $x \in [a, b]$:

$$\langle f(x), g(x) \rangle = \int_a^b f(x)\, \overline{g}(x) dx \tag{3.6}$$

We see that the inner product of the functions $f(x)$ and $g(x)$ are equivalent to the integral between a and b. This notion can be transferred to the vectors generated by these functions under discretization. We want to show that under the limit of data values n of the functions $f(x)$ and $g(x)$ approaching infinity, $n \to \infty$ the inner product of the vectors approach the inner product of the functions. We take $\vec{f} = [f_1, f_2, \ldots, f_n]^T$ and $\vec{g} = [g_1, g_2, \ldots, g_n]^T$ and define the inner product as:

$$\langle \vec{f}, \vec{g} \rangle = \sum_{k=1}^{n} f(x_k)\, \overline{g}(x_k). \tag{3.7}$$

This formula behaves as desired but grows in its value as more and more data points are added, meaning more data points correspond to higher values, which hinders comparison of series with shorter length. So a normalization is added to counter the effect. The normalization occurs through the domain chosen for the analysis $\Delta x = \frac{b-a}{n-1}$:

$$\frac{b-a}{n-1} \langle \vec{f}, \vec{g} \rangle = \sum_{k=1}^{n} f(x_k)\, \vec{g}(x_k) \Delta x. \tag{3.8}$$

This corresponds to the Riemann approximation of continuous functions [1]. As more data more data points are collected and therefore $n \to \infty$ the inner product converges to the inner product of the underlying functions.

The norm of the inner product of the functions can also be expressed as integral:

$$\|f\|_2 = (\langle f,\, f\rangle)^{\frac{1}{2}} = \sqrt{\langle f,\, f\rangle} = \left(\int_a^b f(x)\,\overline{f}(x)dx\right)^{\frac{1}{2}}. \tag{3.9}$$

The last required step is transferring the applicability from a finite-dimensional vector space to an infinite-dimensional vector space. For this we can use the Lebesgue integrable functions or square integrable functions $L^2([a,b])$. All functions with a bounded norm define the set of square-integrable functions [4]. Next we will show how a Fourier series is a projection of a function onto the orthogonal set of sine and cosine functions.

3.3.2 Fourier Series

As the name suggests the Fourier series is an infinite sum of sine and cosine functions of increasing frequency. The mapped function is assumed to be periodic. A simple case of 2π-periodic can be shown as:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k\cos(kx) + b_k\sin(kx)). \tag{3.10}$$

If one imagines that this transformation projects the function onto a basis of cosine and sine, a_k and b_k are coefficients that represent the coordinates of where in that space the function is projected.

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)dx \tag{3.11}$$

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(kx)dx \tag{3.12}$$

$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(kx)dx. \tag{3.13}$$

Those coefficients are are acquired through integration and multiplication of sine and cosine. This expression can be re-written in the form of an inner product:

$$a_k = \frac{1}{\|\cos(kx)\|^2}\langle f(x),\,\cos(x)\rangle \tag{3.14}$$

$$b_k = \frac{1}{\|\sin(kx)\|^2}\langle f(x),\,\sin(x)\rangle \tag{3.15}$$

The squared norms are $\|\cos(kx)\|^2 = \|\sin(kx)\|^2 = \pi$. However, this only works for 2π-periodic functions. For real world data this is obviously most often not the case. Therefore, another term

needs to be added that stretches the 2π-periodicity to length of the observed domain $[0, L)$ with $\frac{kx}{L} * 2\pi$. This L-periodic function is then given by:

$$f(x) = \frac{a_0}{2} + \sum \left(a_k \cos\left(\frac{2\pi kx}{L}\right) + b_k \sin\left(\frac{2\pi kx}{L}\right)\right) \tag{3.16}$$

This modifies the integrals for the coefficients to:

$$a_k = \frac{2}{L}\int_0^L f(x)\cos\left(\frac{2\pi kx}{L}\right) \tag{3.17}$$

$$b_k = \frac{2}{L}\int_0^L f(x)\sin\left(\frac{2\pi kx}{L}\right) \tag{3.18}$$

One can write the formula utilizing Euler's formula

$$e^{ikx} = \cos(kx) + i\sin(kx), \tag{3.19}$$

utilizing complex coefficients ($c_k = \alpha_k + i\beta_k$):

$$\begin{aligned} f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx} &= \sum_{k=-\infty}^{\infty} (\alpha_k + i\beta_k)(\cos(kx) + i\sin(kx)) \\ &= (\alpha_0 + i\beta_0) + \sum_{k=1}^{\infty} \left[(a_{-k} + a_k)\cos(kx) + (\beta_{-k} - \beta_k)\sin(kx)\right] + \\ &\quad i\sum_{k=1}^{\infty} \left[(\beta_{-k} + \beta_k)\cos(kx) - (\alpha_{-k} - \alpha_k)\sin(kx)\right]. \end{aligned} \tag{3.20}$$

For real-valued functions it needs to be ensured that $c_{-k} = \bar{c}_k$ through $\alpha_{-k} = \alpha_k$ and $\beta_{-k} = -\beta_k$. It also needs to be shown that the basis provided by sine and cosine are orthogonal. This is only the case if both functions have the same frequency. We define $\psi_k = e^{ikx}$ for $k \in \mathcal{Z}$. This means that our sine and cosine functions can only take integer values as frequencies. To show that those are orthogonal over the interval $[0, 2\pi)$ we look at the following inner product and equivalent integral:

$$\langle \psi_j, \psi_k \rangle = \int_{-\pi}^{\pi} e^{jkx} e^{-ikx} dx = \begin{cases} \text{if } j \neq k & \displaystyle\int_{-\pi}^{\pi} e^{i0x} = 2\pi \\ \text{if } j = k & \displaystyle\int_{-\pi}^{\pi} e^{i(j-k)x} = 0 \end{cases} \tag{3.21}$$

When $j = k$ the integral reduces to 1, leaving 2π as the result of the interval to be integrated. In case $j \neq k$ the expansion of the Euler's formula expression cancels out the cosine values and sine

evaluated integer multiples of π is equal to 0. Another way to express the inner product is via the Kronecker delta function:

$$\langle \psi_j, \psi_k \rangle = 2\pi\delta_{jk}. \tag{3.22}$$

This result can be transferred to a non-2π-periodic basis $e^{i2\pi\frac{kx}{L}}$ in $L^2([0, L))$. And the final step in the Fourier series is to show that any function f(x) is a projection on the infinite orthogonal-vector space that is spanned by cosine and sine functions:

$$f(x) = \sum_{k=-\infty}^{\infty} c_k \psi_k(x) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \langle f(x),\, \psi_k(x) \rangle \psi_k(x). \tag{3.23}$$

The factor $1/2\pi$ normalizes the projection by $\|\psi_k\|^2$.

3.3.3 Fourier Transform

So far, the Fourier series can only be applied to periodic functions. This means that after the length of the interval the function repeats itself. With the Fourier transform an integral is defined in which the domain goes to infinity in the limit such that functions can be defined without repeating itself. So if we define a Fourier series and its coefficients as:

$$\begin{aligned} f(x) &= \frac{a_0}{2} + \sum_{k=1}^{\infty} \left[a_k \cos\left(\frac{k\pi x}{L}\right) + b_k \sin\left(\frac{k\pi x}{L}\right) \right] \\ &= \sum_{k=-\infty}^{\infty} c_k e^{\frac{ik\pi x}{L}} \end{aligned} \tag{3.24}$$

$$c_k = \frac{1}{2L} \langle f(x),\, \psi_k \rangle = \frac{1}{2L} \int_{-L}^{L} f(x) e^{-\frac{ik\pi x}{L}} dx. \tag{3.25}$$

Our frequencies are defined by the $\omega_k = k\pi/L$. By taking a limit as $L \to \infty$ two properties are achieved:

1. the frequencies become a continuous range of frequencies

2. a infinite precision in the representation of our time series in the Fourier space is achieved.

We define $\omega_k = k\pi/L$ and $\Delta\omega_k = \pi/L$. As $L \to \infty$, $\Delta\omega \to 0$. We take the take the complex coefficient c_k in its integral representation and apply the limit to L:

$$f(x) = \lim_{\Delta\omega \to 0} \sum_{k=-\infty}^{\infty} \frac{\Delta\omega}{2\pi} \int_{-\frac{\pi}{\Delta\omega}}^{\frac{\pi}{\Delta\omega}} f(\xi) e^{-ik\Delta\omega\xi} d\xi \, e^{ik\Delta\omega x}. \tag{3.26}$$

An important side effect of our ω definition is that the frequencies become comparable between time series of different length. For example, we define for two time series S_1 and S_2 the following:

$$n_{S_1} = 6$$

$$n_{S_2} = 12$$

If we now take the frequency $k_{S_1} = 3$ for S_1 we get the following:

$$\omega_{S_1} = \frac{k\pi}{L} = \frac{3\pi}{6} = \frac{\pi}{3} \tag{3.27}$$

If we now adjust the frequency for the length of S_2 we get $k_{S_2} = 6$ - because a repetition of 3 times in a time series of length 6 is the same as the repetition of 6 times in a series of length 12 - we see that ω in both cases is the same as expected:

$$\omega_{S_2} = \frac{k\pi}{L} = \frac{6\pi}{12} = \frac{\pi}{3} \tag{3.28}$$

By taking the limit the inner product of the coefficient, i.e. the integral with respect to ξ turns into the Fourier transform of $f(x)$ and the first part of the Fourier transform pair written as $\hat{f}$ and defined as, $\hat{f} \triangleq \mathcal{F}(f(x))$:

$$\hat{f}(\omega) = \mathcal{F}(f(x)) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx \tag{3.29}$$

The inverse Fourier transform utilizes $\hat{f}(\omega)$ to recover the original function $f(x)$:

$$f(x) = \mathcal{F}^{-1}(\hat{f}(\omega)) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{i\omega x} d\omega. \tag{3.30}$$

As long as $f(x)$ and $\hat{f}(\omega)$ belong to the Lebesgue integrable functions the integrals converge. In effect this means that functions have to tend to 0 as L goes to infinity.

3.3.4 Discrete Fourier Transform

In order to be able to apply the Fourier transform to time series a they need to be applicable to discrete data as well. The Discrete Fourier Transform (DFT) approximates the Fourier transform on discrete data $\mathbf{f} = [f_1, f_2, \ldots, f_n]^T$ where f_j is regularly spaced. The discrete Fourier transform pair is defined as:

$$\hat{f}_k = \sum_{j=0}^{n-1} f_j e^{-2\pi jk/n}, \tag{3.31}$$

$$f_k = \frac{1}{n} \sum_{j=0}^{n-1} \hat{f}_j e^{i2\pi jk/n}. \tag{3.32}$$

Via the DFT $\mathbf{f}$ is mapped into the frequency domain $\hat{\mathbf{f}}$. As before the output in the resulting DFT matrix is complex valued, meaning that it is heavily used for physical interpretations for example in engineering questions.

3.3.5 Fast Fourier Transform

So far we have shown that the Fourier Series and the Discrete Fourier Transform can provide an exact representation of any arbitrary function or data generating process without requiring any assumptions or parameter settings. In the time complexity however we are dealing with an implementation that has complexity $\mathcal{O}(n^2)$. As an example, let's consider the M4 data set, which will be introduced in section 5.0.1. The longest series has $n = 9919$ data points. Given the time complexity of the DFT this will include $\mathcal{O}(n^2) = 9919^2 = 9.8 \times 10^8$ or about 1 billion operations. With the Fast Fourier Transform this can be reduced to a time complexity of $\mathcal{O}(n \log(n))$. In our example this results to $\mathcal{O}(9919 \log(9919)) = 1.3 \times 10^5$ or roughly 130,000 thousand operations. This is a improvement of factor 7,538. It is also an indication that when applied to time series it still provides reasonable time complexity.

To be able to convert the DFT to the FFT a multiple of 2 data points in the vector $\mathbf{f}_n$ of length n is required. For example, take $n = 2^6 = 64$. In this case the DFT matrix can be written as follows:

$$\hat{\mathbf{f}} = \mathbf{F}_{64}\mathbf{f} = \begin{bmatrix} \mathbf{I}_{32} & -\mathbf{D}_{32} \\ \mathbf{I}_{32} & -\mathbf{D}_{32} \end{bmatrix} \begin{bmatrix} \mathbf{F}_{32} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{32} \end{bmatrix} \begin{bmatrix} \mathbf{f}_{\text{even}} \\ \mathbf{f}_{\text{odd}} \end{bmatrix}, \tag{3.33}$$

where $\mathbf{I}_{32}$ is the Identity matrix 32. $\mathbf{D}_{32}$ is:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & \omega & 0 & \dots & 0 \\ 0 & 0 & \omega^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \omega^{32} \end{bmatrix}. \tag{3.34}$$

$\mathbf{f}_{\text{even}}$ contain the even index elements of $\mathbf{f}$, i.e. $\mathbf{f}_{\text{even}} = [f_0, f_2, f_4, \ldots, f_n]$ and $\mathbf{f}_{\text{odd}} = [f_1, f_3, f_5, \ldots, f_{n-1}]$. This process is executed recursively. In our example it would continue like this: $\mathbf{F}_{32} \rightarrow \mathbf{F}_{16} \rightarrow \mathbf{F}_8 \rightarrow \ldots$ This is done down to $\mathbf{F}_2$ where the resulting computations are executed on 2×2 matrices, which is much more efficient than the DFT computations. Of course, it always has be broken down with the same process of taking the even and odd index rows of the resulting vectors. This significantly reduces the required computations to $\mathcal{O} = (n \log(n))$. Important is also that if a series does not have the length n of a multiple of two, it is expedient to just pad the vector with zeros up to the length of the next power of two.

3.3.6 Parseval's Theorem

One property that the Fourier Transform has is central to the approach in this work. It is called Parseval's Theorem. It states that the integral of square of a function is equal to the integral of the square of its transform. In other words, the L_2-norm is preserved. This can be expressed as:

$$\int_{-\infty}^{\infty} |\hat{f}(\omega)|^2 d\omega = 2\pi \int_{-\infty}^{\infty} |f(x)|^2 dx. \tag{3.35}$$

This property is important to us for multiple reasons. It tells us that angles and lengths are preserved in the frequency domain. This means, the different time series are comparable in the frequency domain they way they are in the time domain. And a second consequence that can be derived from this property is that frequencies with comparatively little power in the power spectrum (see section 3.3.7) can be removed from the representation in the frequency domain and still allow very similar reconstruction of the original time series. We will use this property in only comparing the top n most energetic frequencies of the all the frequencies computed in the Fourier transform (see section 4.2.1).

3.3.7 Power Spectrum

One important property of time series transformed into frequency space is the resulting power spectrum or power spectral density (PSD). This concept comes from the signal processing field. The power spectrum denoted as S_{xx} of a time series $f(t)$ describes the magnitude of the frequencies

from which a signal is composed. It describes how the power of a sinusoidal signal is distributed over frequency. Even in the case of non-physical processes it is customary to describe it as power spectrum or the energy of a frequency per unit of time [31].

To obtain the power spectrum we are converting our input vector via the FFT:

$$\begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix} \xrightarrow{FFT} \begin{bmatrix} \hat{f}_0 \\ \hat{f}_1 \\ \vdots \\ \hat{f}_n \end{bmatrix} \tag{3.36}$$

The resulting vector contains the complex values obtained through the FFT. We define the complex value contained in arbitrary value of the vector:

$$\hat{f}_j \triangleq \lambda \tag{3.37}$$

The complex value is represented as $\lambda = a + ib$. We compute the power of the particular frequency:

$$\hat{f}_j = \|\lambda\|^2 = \lambda\overline{\lambda} = (a + ib)(a - ib) = a^2 + b^2. \tag{3.38}$$

This is the magnitude of the particular frequency. In figure 3.3 an exemplatory time series from the M4 data set (see section 5.0.1) is visualized alongside the corresponding power spectrum of its Fourier Transform. The x-axis represents the corresponding frequencies obtained by the FFT, while the y-axis indicates the energy contained in the respective frequencies. The x-axis is plotted in log-scale.

3.3.8 Spectral Leakage

The Fast Fourier Transform (FFT) assumes that the signal continues infinitely in time and that there are no discontinuities. However, any signal in the real world, including time series have finite data points. If the time domain is an integer multiple of of the frequency k than each records connects smoothly to the next. Generally real world processes do not follow sinusoidal wave forms and can contain significant amounts of noise, as well as phase changes and changing trends. So if the signal is not an integer multiple of the sampling frequency k this signal leaks into the adjacent frequency

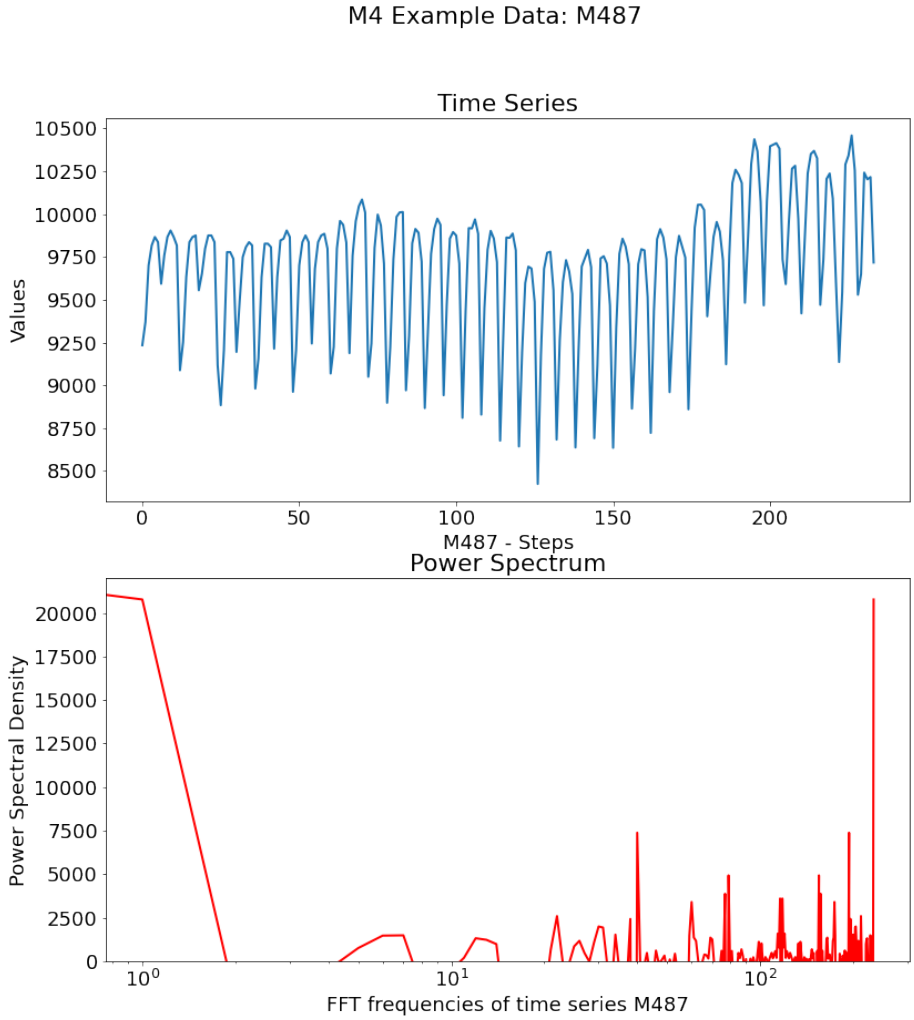

Figure 3.3: Power Spectrum M4 - Example: M487

bins. See figure 3.3 in the power spectrum plot around 10^1. Both on the left a likely example of spectral leakage can be observed. As we intend to use the frequencies ranked by energy level to look for similarities between time series this can be an issue as we want to avoid. that the leaked frequencies are utilized for the determination of the most important frequencies. We will look at window functions to address this issue.

3.3.9 Window Functions

In the field of signal processing a lot of research has been conducted to combat the spectral leakage described in section 3.3.8. One way of addressing spectral leakage are window functions, also called tapering or apodization functions. They help reduce the undesired effects of spectral leakage. They have been used successfully in various areas of signal processing, like speech processing, digital filter design and spectrum estimation [20]. Spectrum estimation is the field in which we will apply them here.

The windows applied to data signals affect several properties of harmonic processors like the Fast Fourier Transform (FFT), for example detectability, resolution, and others [13]. The window functions are designed such that in the spectral analysis they help reduce the side lobes next to the main beams of the spectral output of the Fast Fourier Transform (FFT). A side effect is that the main lobe broadens and thus the resolution is decreased [20]. The spectral power in a particular bin contains leakage from neighboring bins. The window function brings the data down to zero at the edges of the time series. An example applied to a series from the M4 data set can be seen in figure 3.4.

The Hamming window is named after R.W. Hamming. It is one of many window functions and is defined as

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{M-1}\right) \quad 0 \leq n \leq M-1, \tag{3.39}$$

with M being the length of time series to be covered. As can be seen in the figure, it minimizes the sidelobes created by the FFT, but it also minimizes valid signal at the edge of the time series data. This of course, negatively impacts the FFT results as some frequencies maybe overlooked or misidentified.

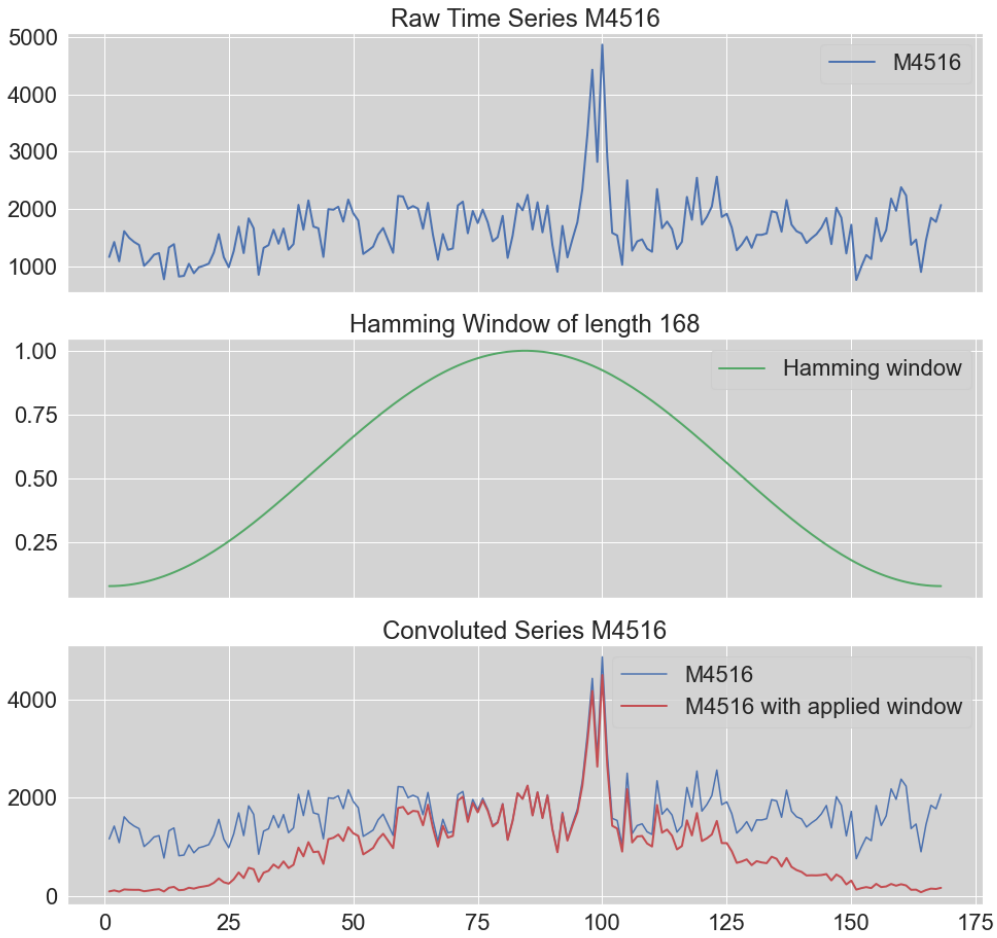

Figure 3.4: Hamming window example with M4 time series M4516

3.3.10 Bartlett's and Welch's Method

Another approach to address spectral leakage that is usually integrated with window functions is to average periodograms generated over multiple subsets of the time series. In this thesis we will use Welch's method which is based on Bartlett's method that is described in the following [2]. Let us denote the x[th] periodogram or power spectrum as $\hat{P}$. The principal idea is that the average of the computed periodograms is unbiased:

$$\lim_{N\to\infty} E\{\hat{P}_{per}(e^{j\omega})\} = P_x(e^{j\omega}) \tag{3.40}$$

So a consistent estimate of the mean, is a consistent estimate of the power spectrum. If we can assume that the realizations in the time series data are uncorrelated then they result in a consistent estimate of its mean. This means that the variance of the sample mean reduces with the number of measurements. They are inversely proportional. Therefore, averaging periodograms produces a the correct periodogram of the data. If we let $x_i(n)$ for $i = 1, 2, \ldots, K$ be K uncorrelated realizations of a random process $x(n)$ over the interval of length L with $0 \leq n < L$ and with $\hat{P}^{(i)}_{per}(e^{j\omega})$ the periodogram $x_i(n)$ is:

$$\hat{P}^{(i)}_{per}(e^{j\omega}) = \frac{1}{L}\left|\sum_{n=0}^{L-1} x_i(n)e^{-jn\omega}\right|^2 \quad ; \quad i = 1, 2, \ldots, K \tag{3.41}$$

These periodograms can then be averaged

$$\hat{P}_x(e^{j\omega}) = \frac{1}{K}\sum_{i=1}^{K} \hat{P}^{(i)}_{per}(e^{j\omega}) \tag{3.42}$$

and give us an asymptotically unbiased estimate of the power spectrum. Because of the assumption that the values are uncorrelated, the variance is inversely proportional to the number of measurements K

$$\text{Var}\left\{\hat{P}_x(e^{j\omega})\right\} = \frac{1}{K}\text{Var}\left\{\hat{P}^{(i)}_{per}(e^{j\omega})\right\} \approx \frac{1}{K}P_x^2(e^{j\omega}) \tag{3.43}$$

However, the assumption that the time series data is uncorrelated does not hold. Bartlett proposed to circumvent that to partition the data into K non-overlapping sequences of length L with a time series $X = \{x_1, x_2, \ldots, x_n\}$ of length N such that, $N = K \times L$.

$$\begin{aligned} x_i(n) = x(n + iL) \quad & n = 0, 1, \ldots, L-1 \\ & i = 0, 1, \ldots, K-1 \end{aligned} \tag{3.44}$$

In consequence, Bartlett's method can be written as:

$$\hat{P}_B(e^{j\omega}) = \frac{1}{N}\sum_{i=0}^{K-1}\left|\sum_{n=0}^{L-1} x(n+iL)e^{-j\omega}\right|^2 \tag{3.45}$$

An example of the split of time series can be seen in figure 3.5.

Figure 3.5: Bartlett's window example with k=3'from M4: D3720

Welch's method differs in how the windows are applied to the data set. For Welch's method the windows are not adjacent but are overlapping. The original data set is still split into K sequences of length L overlapping by D points with $0 \leq D < 1$. If the overlapping is defined to be 0, then this method is equivalent to Bartlett's method. An overlap of 50% can be achieved via $D = K/2$. The overlapping of the data segments effectively cures the fact that an applied window minimizes the data at the edges of the window. The i$^{\text{th}}$ sequence can be described by $x_i(n) = x(n + iD)$; $n = 0, 1, \ldots, L-1$ with L being the length of a sequence. N can be computed by $N = L + D(K-1)$

where K is the number of sequences. Welch's method is described by

$$\hat{P}_W(e^{j\omega}) = \frac{1}{KLU} \sum_{i=0}^{K-1} \left| \sum_{n=0}^{L-1} w(n)x(n+iD)e^{-jn\omega} \right|^2 \tag{3.46}$$

with

$$U = \frac{1}{L} \sum_{n=0}^{L-1} |w(n)|^2 \tag{3.47}$$

An example of time series split via Welch's method with $K = 4$ and no applied window can be seen in figure 3.6.

Figure 3.6: Welch's method windows example M4: D3720

Chapter 4

Methodology

In this chapter we describe the mechanism of our algorithm and various aspects of its implementation. First, we provide a general overview of our method (4.1) followed by a definition of our frequency ranges and how we assign the FFT results to these intervals and match time series through them.

4.1 General Overview

The main idea of our algorithm is to transform each of the time series into the frequency domain and utilize the identified underlying frequencies as their most important property for defining their similarity to other time series. In a second step additional statistical metrics are used to reduce the number of similar series such that the user of the application can decide which of those metrics should be used the comparison between time series.

The whole process consists of two general phases with further subdivisions of which only the second should be considered for computing the run-time of this method. Phase I is a preparatory step required to set up the pool of time series which serve as the database from which the closest matches are identified. Phase I consists of:

1. Data Transformation (see section 4.1.1)

2. Statistical Metrics Computation (see section 4.1.2)

Phase II describes how a single series considered as template series is matched against all available series in the database (see section 4.1.3).

4.1.1 Data Transformation

The preparation of the time series pool is done by executing the data transformation for all time series and computing the statistical metrics for all time series (section 4.1.2). The data transformation is based on the Fast Fourier transform (FFT) and is executed multiple times for each series with multiple transformation types: (1) FFT with original data, (2) FFT with applied Hamming window on each time series, and (3) FFT with Welch's method and a Hamming window applied on each sub series for each time series. For a shorthand in the following "FFT" or "regular FFT" is used to describe the Fast Fourier transform without modification to the original data, "Hamming" is used to describe the FFT with a Hamming window applied to the original data, and "Welch" is used to describe the Fast Fourier transformation while applying Welch's method with a Hamming window on each subseries. We abbreviate the transform type with τ. The results from all three transformations are kept separately for later comparison to the template series.

After the transformations have been created only the top K (in our case top 5) frequencies, meaning the 5 frequencies with the highest magnitude in the frequency domain are retained and frequency range intervals are created (see section 4.2.1). The top K frequencies are then associated with their respective frequency interval (see section 4.2.2). This process is depicted visually in figure 4.1.

With the completion of this step we have each time series associated with a list of K frequency intervals ordered from lowest magnitude to highest magnitude associated with the respective series. So each time series is described by 5 data points irrespective of the length of the original series. Aside from other benefits this already hints at the fact that comparing 5 data points per comparison will be executed significantly faster than comparing hundreds or thousands of data points.

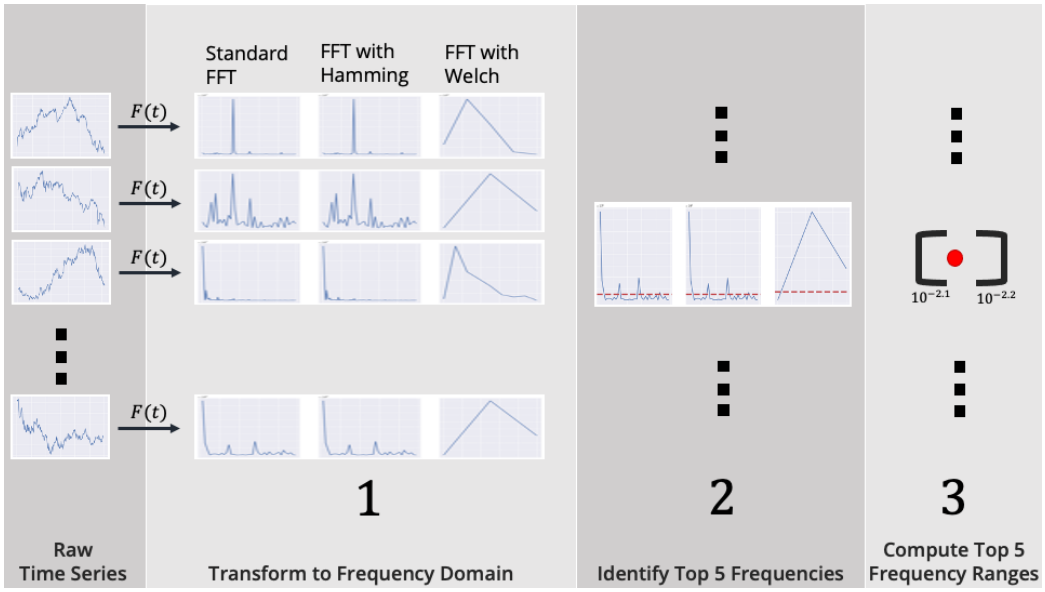

Figure 4.1: Phase 1a: convert time series pool to frequency space and identify top 5 frequency ranges

4.1.2 Statistical metrics computation

Describing a time series only by the top K frequency intervals in the Fourier domain is not sufficient to adequately describe the properties of a time series for matching it with other series. This, in part, is due to the fact that the magnitude of the particular frequency is not taken into account. In order to accommodate this aspect of the FFT we use other well understood metrics and compute additional statistical measures for the raw series and add them as additional data points describing the time series in the pool.

As shown in figure 4.2 the additional metrics are computed on the original time series, consisting of: (1) trend, (2) mean, (3) median, (4) standard deviation, (5) quantiles, and (6) minimum and max values. These metrics will be used flexibly to find similar series that match singular or multiple criteria. In essence the prior step of finding the underlying frequencies ensures that the time series follow similar periodicity or seasonality. The statistical metrics contain additional information that allow to find time series in the pool that, for example have similar value distribution through the standard deviation, etc and therefore match the users needs for a particular use case.

The trend mentioned above is not strictly a statistical measure. However, we compute the slope

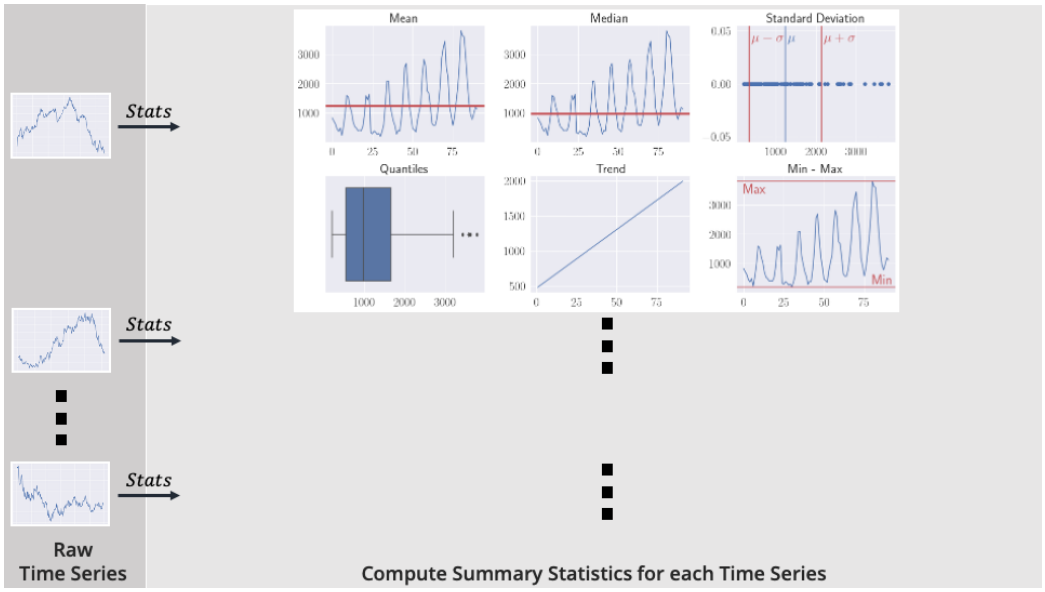

Figure 4.2: Phase 1a: compute simple statistical metrics in time series pool for later comparison

m via linear fit of equation:

$$f(x) = mx + b \tag{4.1}$$

through the data to identify the trend direction of the time series.

Noteworthy is also the fact that the time complexity of the statistical metrics does not exceed $\mathcal{O}(n \log(n))$ for most metrics and does not exceed $\mathcal{O}(n)$ for all metrics. It of course depends on the sorting algorithms used for the computation. Assuming quicksort or mergesort this holds true for all cases. This observation also includes the computation of the linear fit which is $\mathcal{O}(c^2 n)$ with c representing the number of features which for our case area $c = 1$, because we only have one feature or variable; hence time complexity for linear fit reduces to $\mathcal{O}(n)$. This observation let's us conclude that the computation for the statistical metrics will be feasible during the similarity search for the template time series even if n is very large.

4.1.3 Matching of time series

After the completion of phase I the time series pool is ready for use. When a new time series is to be matched against the pool phase I of our algorithm needs to be executed for the individual template time series, meaning the data transformation into frequency space and computation of the statistical

metrics. Subsequently, for each of the of the Fourier transform types τ (regular, Hamming, Welch) the highest matching score χ (see section 4.2.3) between the template time series S_t and each of series in the pool S_n is computed via:

$$\underset{\chi \in S_N^\tau}{\arg\max} f(\chi_{S_i}^\tau) = \chi_{S_i}^\tau. \tag{4.2}$$

This reduces the pool of the matching series to all time series from the pool per FFT type that are equivalent to the highest matching score for that transformation type. The remaining series are discarded. Next an additional limitation is applied that restricts the result set of matching series (named A) to having a trend that must match in general slope direction (up/down) to the slope of the template time series

$$A_{trend} = \{S_i \in S_N \mid \mathbb{1}\left(-\frac{m_{S_t}}{|m_{S_t}|} = -\frac{m_{S_i}}{|m_{S_i}|}\right)\}, \tag{4.3}$$

where m_{S_t} is the slope of the template time series and m_{S_i} is the slope of the time series of the time series pool S_N currently under investigation. This metric in our algorithm is used to rule out time series from the pool that have a trend that goes into the opposite direction of the template series. This property is not easily discernible from the coefficients found in the Fourier transform. For example if the series for which we want to find matching series in the pool has a negative trend, all series with a positive trend from the result set are ruled out before the other statistical metrics are utilized. However, if the trend for the investigation at hand is not relevant this step can easily be removed.

The last step in our algorithm to match series involves optimizing for one of the other statistical metrics computed on the original time series. With the metrics described in section 4.1.2 it makes sense to optimize for the lowest delta in desired statistical metric on the remaining result set after the previous matching steps. This selection is executed without regard for the transform method used as the metrics are comparable. The ranked difference between the template time series and the pool series is then used to select the most matching series

$$\underset{S_i \in S_N}{\arg\min} f(S_i) := |\phi_{S_t} - \phi_{S_i}| \tag{4.4}$$

where ϕ represents the chosen statistical metric at hand. S_t refers to the template series and S_i indicates a particular instance of the pool series with the whole pool of length N. Figure 4.3 provides a pictorial overview of the time series matching process.

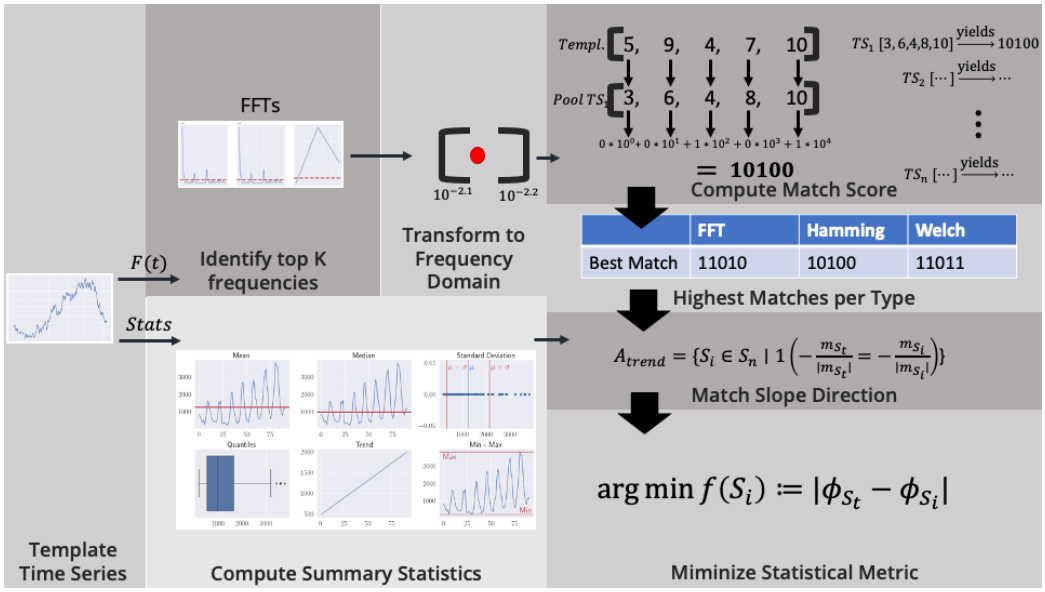

Figure 4.3: Matching pool time series to template time series process

From the resulting order of the series one or multiple elements can be used to identify the most similar series in the application of this algorithm. This can be done in multiple ways and is task dependent. This procedure does not impose some absolute truth in the results but rather a gradient of closeness that begins by the determining the frequencies contained in the Fourier domain as the most important descriptor of similarity between series. The remaining metrics then build upon the reduced result set to optimize for some aspect of similarity between the series.

4.2 Frequency handling in time series

In this section we analyze the setup of the chosen frequency intervals chosen used for the matching of time series in frequency domain (4.2.1). Further we describe the assignment mechanism to associate frequencies with their respective interval (4.2.2). And last we describe the mechanism of assigning a match score between two time series (4.2.3).

4.2.1 Frequency Intervals

We want to be able to compare the closeness of two time series by comparing their frequencies with each other. Due to Parseval's Theorem (see section 3.3.6) we know that properties of the raw series are partially preserved in the frequency domain. Equation (3.35) states that the energy contained in the norm of the frequency domain of the transformed time series is equal to the norm of $f(x)$. The energy in the norm of the transform is proportional to the norm of $f(x)$. What we can derive is that if there are coefficients in the transform that are very small, they can be ignored without meaningfully impacting the result of the integral in the transform. Therefore, a truncated Fourier transform ranked by the magnitude of the coefficients will still remain a very good approximation to its original series. Additionally, because the Fourier transform is a unitary operator, meaning, it preserves lengths and angles in the frequency domain different series are comparable within in the Fourier space. So the distance between two time series is preserved in the frequency domain.

We utilize these properties by selecting the frequencies with the n largest magnitudes for a comparison. We select multiple frequencies and rather than computing the distance between each of the same-ranked frequencies we want to assign them to a range band that can be used to capture whether two time series have frequencies at the same rank that matches within a certain bandwidth. This is an approximation of the distance as frequencies will be determined to be similar up to a certain distance and then be declared not matching. A second observation is that lower frequencies have a larger impact on the overall shape of a time series than mid level and often also higher frequencies. Therefore, a match at lower frequencies requires a more precise - hence narrower - band than a match at mid level and higher frequencies. To accommodate this observation the range band is defined by the set Ω' defined on a logarithmic scale

$$\Omega'_n = \{\omega' = 10^{\upsilon} \in \mathbb{R} \mid \upsilon = k \cdot \Delta \ \wedge \ k \in \left[\frac{a}{\Delta}, \frac{b}{\Delta}\right], \quad \Delta \in \mathbb{R}_+, \ k, a, b \in \mathbb{Z}\}, \tag{4.5}$$

where ω' denotes the identified frequency range, υ defines the power to which the base is raised, and Δ is a fixed value defining the step size between the range intervals; a and b are the lower and upper limit of the interval($a < b$). Generally $k \ll a$ and k must be an integer value to delineate the interval borders. An example can be seen in figure 4.4. For the figure a wider step size of was

chosen and the x-axis shown for both FFT and Hamming was limited to a smaller section so that the individual bins and their associated values are visible.

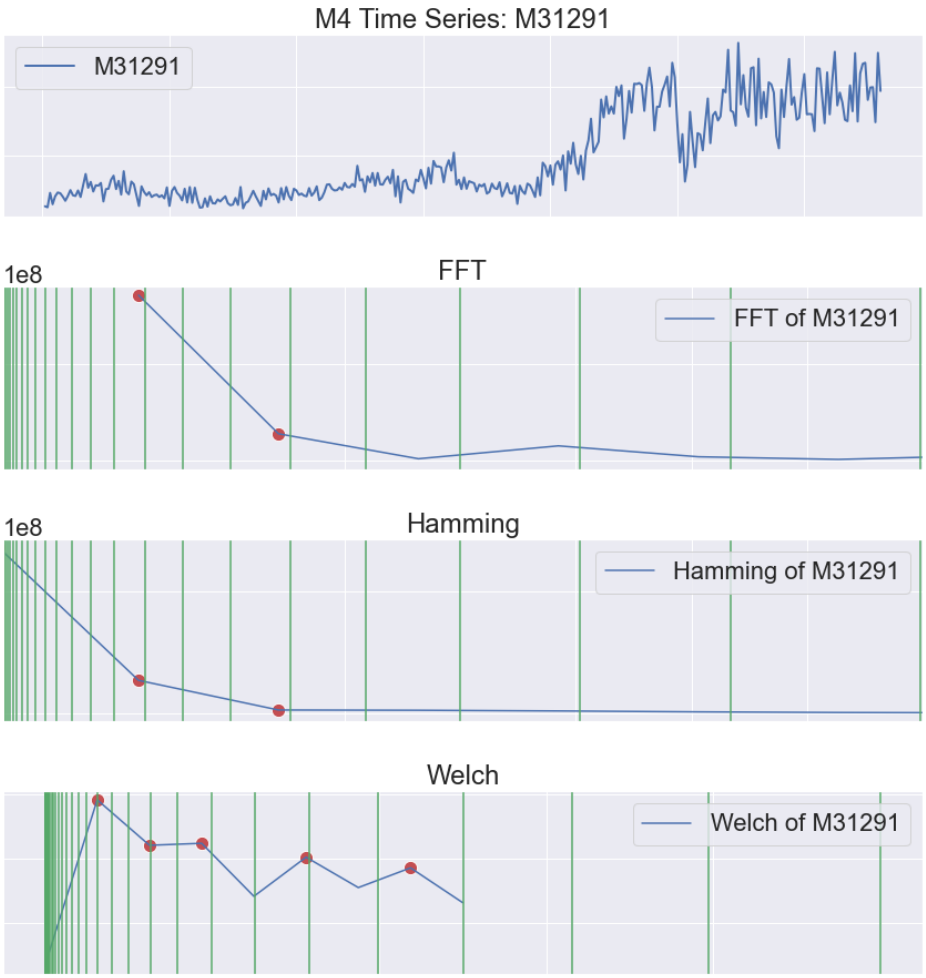

Figure 4.4: Frequency ranges definition - FFT example M4 data: M31291 with parameters $a = 10^{-4}$ to $b = 10^{0}$ with $\Delta = 0.1$

For the data processing in this thesis we work with: $a = -4$, $b = 0$ and $\Delta = 0.01$.

4.2.2 Assigning frequencies to an interval

The top K frequencies need to be assigned to their respective interval defined in section 4.2.1. The association is done via this mechanism:

$$M_n(\omega) = n\mathbb{1}\Omega'_n(\omega) \quad \omega \in \left[\frac{a}{\Delta}, \frac{b}{\Delta}\right]. \tag{4.6}$$

with ω representing one of the top K frequencies identified via the FFT and ω' the respective representation in the frequency interval set Ω'. As an example, imagine a frequency identified via the FFT of $\omega = 0.003$ with $a = -3$,$b = 0$, and $\Delta = 0.1$. The value of ω falls into the interval $[10^{-2.6}, 10^{-2.5}]$. If Ω' is indexed from 0, the result will be $M_n(\omega) = 6$. Note, that the result is an index values rather than interval or specific value inside said interval.

4.2.3 Matching frequencies between time series and ranking results

To match the frequencies between time series a mechanism is required that computes the rank of the match (the score χ) within the top K frequencies. We use the another logarithmic scale with base 10 to signify the importance of the match which can later be used for ranking the results with:

$$\chi = \sum_{k=0}^{K-1} 10^k \mathbb{1}\left(\omega_k'^{(S_1)} = \omega_k'^{(S_2)}\right) \tag{4.7}$$

where $\omega_k'^{(S_n)}$ represents the k$^{\text{th}}$ ranked frequency band ω' of time series S_n. The score is computed for each time series in the time series pool for each transform type τ, meaning regular FFT, Hamming, and Welch's.

For each transform type τ all series are ranked based on their matching score χ in descending order. A higher matching score χ means that the more dominant frequencies in the series match. In the algorithm all time series from the pool that have the highest match score per transform type τ are selected for further processing that utilize the statistical metrics.

4.3 Technological considerations

Various different technologies and programming languages are suitable for the implementation of the algorithm proposed in this thesis. These technologies must include: (1) ability to read comma

and tab-separated value files, (2) have signal processing methods, like Hamming and Welch's window available or accessible, (3) support Fast Fourier Transform, and (4) allow for multi-processing. Ideally, there should be a standardized method for Dynamic Time Warping (DTW) available as well to allow for a standardized formal comparison. Using integrated and tested functionality allows for easy reproduction of the results and render points 1 - 3 self-explanatory. Multi-processing is very useful for the implementation of this method because working on the full data sets for M4 and UCR (introduced in section 5) required significant computation time in the case of the FFT approach and would be prohibitive in the case of DTW.

GUI-integrated data science software platforms like Knime or H2O where not considered for this work. The main programming language candidates for this thesis where Python, Matlab and R under the utilization of freely available software modules supporting the above requirements and Wolfram Mathematica which also integrates all above required functionality. There is some overlap between these languages as well. For example the matrix operations in all these languages are based on LAPACK and BLAS Fortran subroutines [16]. According to the Mathematica help LAPACK and BLAS are also utilized by Wolfram Mathematica. For the Dynamic Time Warping implementation R offers a suitable software package [11]. The implementation also offers a Python interface which we utilized for the comparison algorithm.

The visualization capabilities in each of the aforementioned programming languages is outstanding and easily capable of all needs for this thesis.

Given this flexibility of software technology and the fact that the results are somewhat comparable due to the underlying modules we chose based on our familiarity with the language and decided for a Python-based implementation. Some auxiliary visualizations are implemented in Mathematica for their ease of implementation in the technology.

Chapter 5

Data Analysis

We generate our algorithm on a data set with which the subsequent results can be reproduced. The data set of our choice also needs to encompass real-world scenarios so that the methods proves its validity for the real world ideally in wide range of fields with differing time granularities. In the literature two widely used data sets can be found which are introduced in sections 5.0.1 and 5.0.2. For the process of developing the FFT-based similarity detection method the M4 competition data was used [24]. All parameter choices were done with the exploratory data analysis results of the M4 data. To verify their veracity the formal evaluation of our method's results were conducted with the UCR Time Series Classification Archive [6]. This was done ensure that the results found and parameter choices made are applicable between different data domains and time granularities, as well as providing reference points for quality of the method described in this thesis.

5.0.1 M4 competition data

In his popular book "The Black Swan - The Impact of the Highly Improbable" published in 2008 the author Taleb introduced the M-competitions and its merits to an international readership. By that time 3 M-competitions were already conducted with the first one done in 1982. Its inventor Prof. Makridakis held the first forecasting competition as a follow-up to a controversial paper published in 1979 [26]. The paper found that more sophisticated forecasting methods tended to lead to less accurate predictions, a view for which he was highly criticized and personally attacked.

The forecasting competition was an answer to the accusations to allow the experts to fine-tune their favorite forecasting methods to the best of their knowledge and compete for the most accurate predictions on the hold-out set [25]. The competition was based on 1001 different time series and provided an inside into the different properties of the various used forecasting methods. The data itself was selected with varying time granularities, different start and end times. It was chosen among data from different industries, firms and various countries. It consisted of macro-economic and micro-economic data. The results observed in the earlier work from 1979 was confirmed in the forecasting competition. The main observations were that statistically sophisticated methods on average provided not more accurate forecasts than simpler methods and accuracy improvement can be achieved by combining the results from various different methods [15].

With the M4 a random selection of 100,000 series was performed by Professor Makridakis and provided for the forecasting competition in 2018. It included data with a time granularity ranging from hourly, daily, weekly, monthly, quarterly, and yearly data. It came from various areas: micro-economical, industrial, macro-economical, finance, demographic and miscellaneous areas [27]. This is a wide field of mostly socio-economic data with varying time granularities, different time series length. What is not present or possibly underrepresented in the data set are time series generated by technical processes, like machine or sensor data. Nonetheless, these time series data are an ideal candidate to develop and test and method for discovering similar time series. This time series archive was chosen as the data set to develop the algorithm of identifying similar time series quickly based on their Fast Fourier transform.

The latest completed iteration of the Makridakis-competition is the M5 [35]. It was completed in 2021 and was set up with product sales in 3 different states in 10 different stores in the United States. It consisted of the sales of 3490 different products sold by Walmart. The data came from an identical time frame ranging from 2011 to 2016. Due to the similar nature of the data contained in this data set it was ruled as the basis for our investigation.

At the time of this writing in fall 2021, the next installment of the Makridakis-competition, the M6 is in planning to be conducted starting in February 2022.

5.0.2 UCR time series data

Another important data set with an even broader usage in time series research is the UCR Time Series Archive. It was first formed in 2002 by Prof. Keogh [5]. It's intention was to provide a baseline for time series research which prior to that point mostly relied on testing a single time series per paper. The creators concluded that this makes comparing the results between papers almost impossible. The data set was expanded in the subsequent decades with the last major expansion being conducted in 2018.

In his 2003 published paper Keogh and Kasetty describe the error of data bias which comes from testing new methods on a single time series or time series of the same kind, for example ECG data but extending the claim of the found results to various types of time series data or all time series data types [18]. With this in mind the UCR Time Series Archive was compiled and subsequently extended with various time series from various areas including: (1) Image, (2) Spectro, (3) Sensor, (4) Simulated, (5) Device, (6) Motion, (7) ECG, (8) Traffic, (9) EOG, (10) HRM, (11) Traffic, (12) EPG, (13) Hemodynamics, (14) Power, and (15) Spectrum time series data. This is a wide spectrum of data which is different from the socio-economic data of the Makridakis competition data sets. Therefore, this data set is a great candidate to validate the findings of the time series similarity search and conduct a formal evaluation of the results found via the M4 data set. Furthermore, it provides a classification category for each time series data set which in itself is made up of multiple time series. In this way running a formal evaluation, we can measure how many data sets are identified between the train and test set of the data that belong to the same data set and data set class. This metric can then be compared between the Dynamic Time Warping (DTW) algorithm and our method.

5.0.3 Exploratory data analysis

In order to be able to set parameters for the the utilized methods in the data transformation (see section 4.1.1) and the computation of the statistical metrics (see section 4.1.2) an understanding of the used data is necessary. Please note, that the decision for parameters was done based solely, on the M4 competition data set (section 5.0.1) and the UCR data set was only introduced during the

formal evaluation (section 5.0.2).

The first analyzed aspect is the length of the time series in the two repositories. In figure 5.1 can see that the lengths of the two repositories time series have different distributions. For the M4 data set has the wider range of $[13, 9919]$ while the UCR data set is distributed between a length of $[8, 2844]$. For the M4 data set the data is more concentrated around the length of roughly 100 data points and a second peak at 320 data points. Further, there are some time series with longer series concentrated around 4000 data points. The mean is 240 data points and the median is 97, meaning there are some outliers on the longer side of the distribution. This is confirmed by the boxplot of the lengths of the two repositories (see figure 5.2). The UCR repository doesn't have as many short or long series compared to M4. The main concentration is similar to M4, with a bimodal distribution around roughly 100 data points and a second higher peak at 650-700 data points. But the lengths are more concentrated in that region, confirmed by lower standard deviation of the UCR data set compared to the M4 data. The UCR data is also impacted by a few outliers leading to a higher

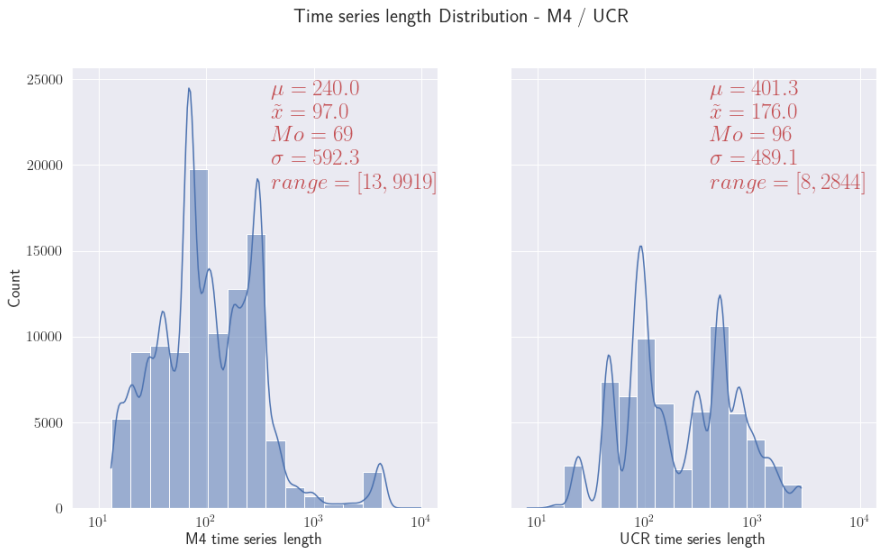

Figure 5.1: Histogram / KDE of time series length in repositories

mean than median as can be seen in the boxplot figure 5.2. But this is less so compared to the M4 data which has roughly half of the mean and median value to the UCR data.

These observations are interesting for multiple reasons. For one, they will reveal whether the

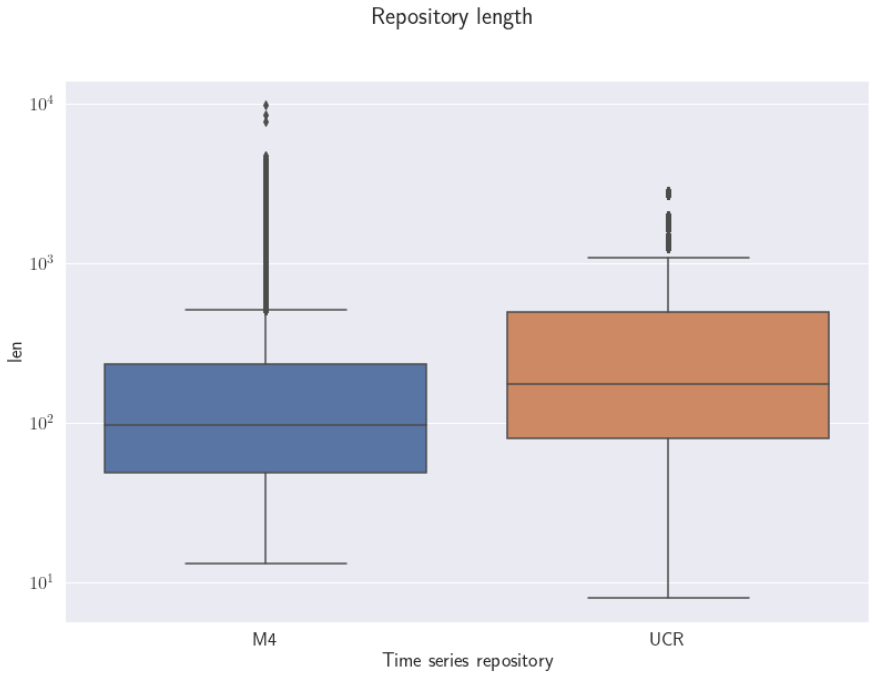

Figure 5.2: Boxplots of M4 and UCR time series length

devised method for finding similar time series works equally well irrespective of the length of the underlying pool time series. Furthermore, the data can be used to illustrate the compression levels achieved in the computation of the similar time series via the FFT. For the M4 data set with a $\mu = 240$, the reduction to the top 5 frequencies for the comparison with other time series leads to 60x reduction in data points required for comparison. The longest series in M4 is reduced by factor 1980. Aside from the algorithm being in a favorable time complexity class of $\mathcal{O}(n \log(n))$ also a constant term of very few data points is required for the comparison in the Fourier domain. The compression is even more favorable in the UCR data set. The $\mu = 401.3$ data sets lead to a compression factor of 80 on average.

The next area of analysis is the value distribution of the time series both in UCR and the M4 repositories. As can be seen in figure 5.3 the values in both data repositories are distributed very different. The M4 data set has more times series than the UCR data set, 100000 vs. ≈ 65000. But the UCR data set contains on average longer series totaling about $\approx$ 26 million data points, compared to $\approx$ 24 million data points for the M4 data set. The distribution of those data points is over a

wider value range for the M4 repository, spanning $[10, 703008]$. In contrast the UCR data set covers the range of $[-1110.8, 24929]$ including negative values but covering a much smaller spectrum of values. This is reflected in the mean values for both repositories, for M4 $\mu = 4841$ and for UCR $\mu = 7.96$. Both distributions respectively contain large positive outliers and therefore the median values are lower. M4 has a median of 3689, while for UCR the median is 0.00137. The difference of the distribution can also be seen in the different standard deviations of both distributions. M4 has a $\sigma = 5724$ and UCR has $\sigma = 99.67$.

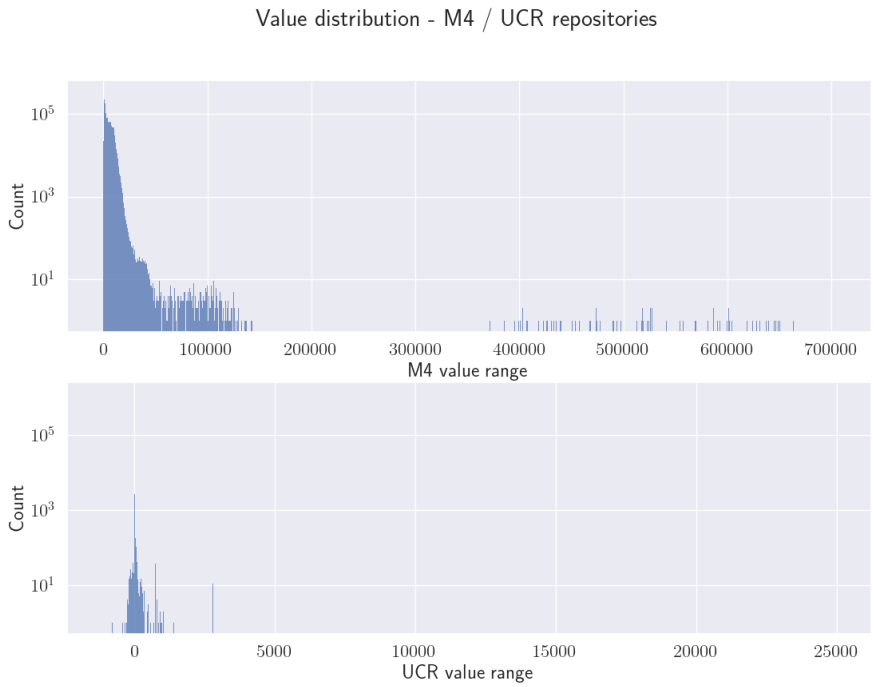

Figure 5.3: Value distribution for M4 and UCR data repositories

Another interesting area of comparison is the distribution of the two data repositories in the Fourier domain. The distribution of the top 5 frequencies for M4 and UCR data sets can be seen in figure 5.4 setup with a log-scale for the y-axis. We observe that the frequency distribution differs between the two data sets. First, we notice that the distribution of both the regular FFT and Hamming FFT are similar in both data sets. However, for the M4 data the Hamming windows the middle of the frequency spectrum around 0.5 shows a higher concentration of frequencies compared to the regular FFT. This is not observable for the UCR data set. Overall, the distribution for the UCR data

set is more smoothly distributed compared to the M4 data. Possibly, this can be explained by the fact that the data in the M4 data is more socio-economic data leading to more erratic Fourier transforms compared to the more technical time series from the UCR repository, where more regular patterns are observable. For both repositories a rise of frequencies along the edges can be seen. Especially the left edge at the lower frequencies is very important because the lower frequencies have a higher impact on the overall shape of the resulting time series. Therefore, a more granular setup of ranges is beneficial for the comparison of the series, as described in section 4.2.2. Noteworthy is also the distinct difference between the Hamming and regular FFT frequency spectrum on the one side and the Welch's method frequencies on the other. The averaging of the subsegments leads to overall lower frequency values. It also visualizes why the intra-FFT-type frequency comparison would result in misleading results.

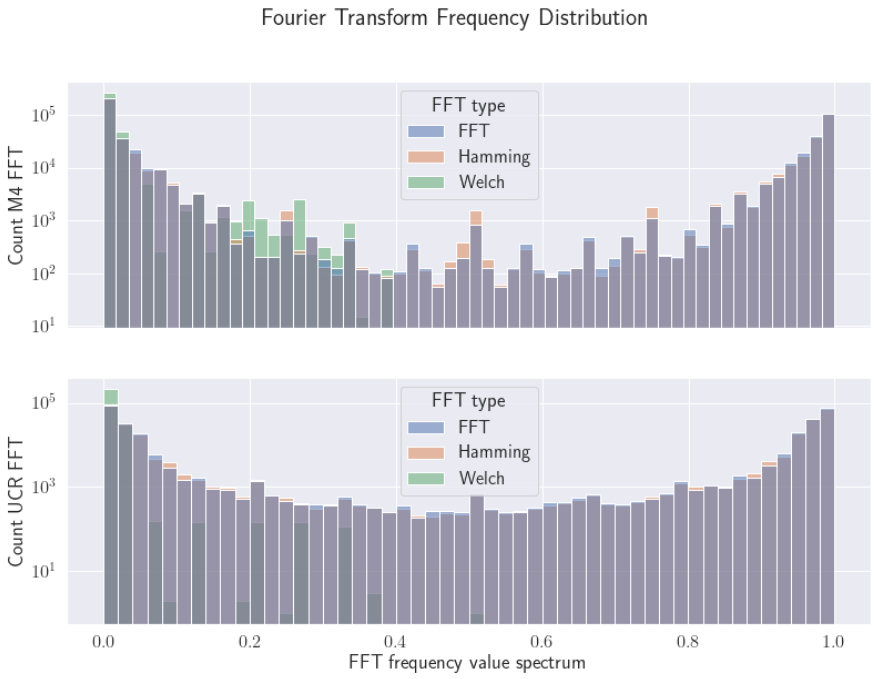

Figure 5.4: FFT frequency distribution for M4 and UCR data set

A further drill-down into the distributions reveals more differences between the two data repositories which can be observed in figure 5.5. Here, the frequency distributions are shown by rank of frequencies. For example, row 1 in the plot indicates the distribution of the highest ranked frequen-

cies by transform method separated by the M4 repository on the left and the UCR data set on the right side. We observe that different data makeup between the two repositories is even more obvious than before. For example, the M4 data set has a narrow distribution of frequencies ranked at the top stop. In fact, Hamming and regular FFT are exclusively zero and only the Welch's method has some spread, likely due to the averaging of the segments. Another, explanation are the higher average values of the M4 data set which requires a different y-axis offset. This is accomplished via frequency zero which results just in a flat line when inverse transformed from frequency domain to the original domain. The top 2 ranked frequencies for the M4 data set span the whole range of frequencies but a different distribution between the transform types can be observed. The Hamming FFT is found at the left and right edges whereas the regular FFT frequencies are distributed smoother with an increase towards the higher range of the frequency spectrum. The data is consistently occupies a smaller range between $[0.1, 0.4]$. For the UCR repository the distribution is comparable to the first ranked frequencies with a sharper drop-off in the middle of the frequency domain. Rank 3, 4 and 5 repeat the previous patterns of the respective distributions for M4 and UCR data. The most notable differences are the general deviation between the M4 and UCR repository and second that rank 4 for the UCR data set has the largest amount of frequencies gathered around zero for Welch's methods frequencies.

The review of the ranked frequency distributions does not reveal any information that indicates that the ranks should be treated differently from the process defined in section 4.2.

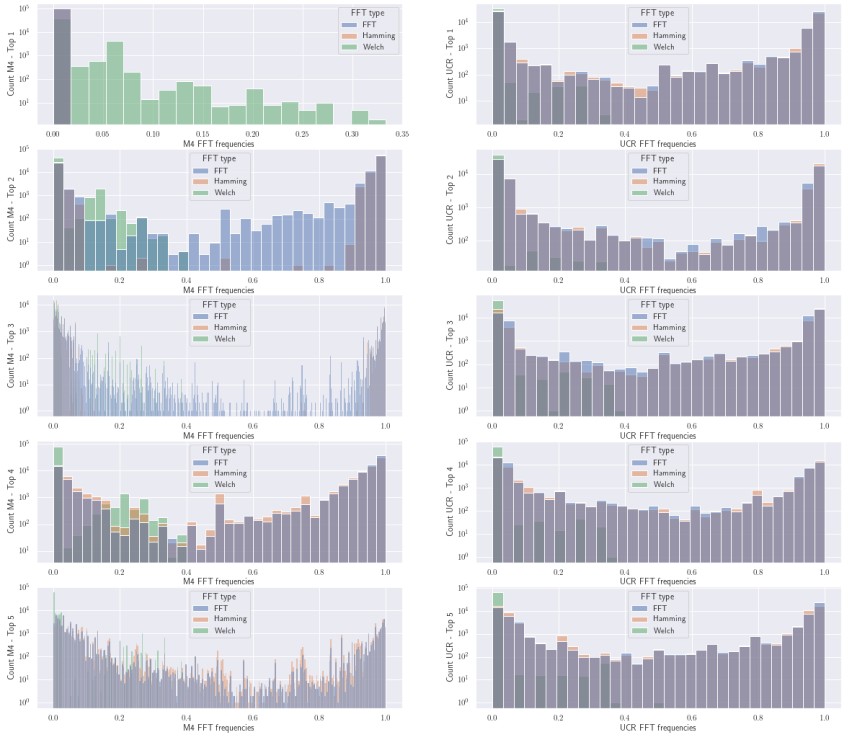

Figure 5.5: FFT frequency distribution by top k for M4 and UCR data set

Chapter 6

Formal Evaluation

The formal evaluation establishes a formal comparison between the algorithm presented in this work with the widely used and generally accepted method of Dynamic Time Warping for finding similar time series. We will introduce the evaluation method chosen, discuss the consequences of the different time complexities of the algorithms, and compare the accomplished results via the two methods.

6.1 Evaluation Method

To evaluate the performance of the algorithm proposed in this work the following procedure was applied to the UCR time series data set. To generate the results with our the method the whole data set was transformed and augmented as described in section 4. After the transformation and stratified random sample of 1420 was chosen from the UCR test data set, such that each different data repository inside UCR data set is represented.

The resulting data test time series have then been processed by both algorithms Dynamic Time Warping (DTW) and our algorithm described in section 4. The DTW method has been introduced by **(author?)** which was implemented via a statistical software package in R [12]. The only change to the standard call of the method was the utilization of the option to only compute the distance and not generate any additional data for plotting the data or other metrics to improve the runtime. The random sample was run over the time series pool of all training data irrespective of the category

the data was chosen from. For the DTW method the closest match from the time series pool was recorded. This results in one closest match per time series from the sample.

For the FFT-based method to find similar time series the procedure was similar in that each time series from the same sample was compared to all series from the time series pool of all training time series. There are multiple results through the different transform types τ with the regular FFT, Hamming and Welch's and the different summary statistics the lowest result for each combination was recorded. To illustrate, each time series converted to each window type, for example Hamming each KPI (mean, median, standard deviation, etc) was recorded based on the closest Δ value for each particular summary statistic ϕ. So per time series and window type 8 minimized KPI values were recorded. With these results the performance evaluation was conducted.

6.2 Time complexity and duration

Due to the different time complexities of the underlying algorithms two different computers were used for the execution. Both algorithms were parallelized to allow to utilize more powerful compute infrastructures. The FFT-based algorithm was executed on a personal laptop with 8 Intel-based CPU, 16GB RAM machine that simultaneously also ran other user-based activities. For the DTW a cloud-based machine with 32 CPU and 64 GB of RAM was chosen. Comparing the actual runtime on these different hardware and different operating system is not a good scientific measure of performance but is provided here to give an indication of the class difference between the two algorithms. The DTW required 2.9 days of execution time on the cloud hardware, compared to 22.5 minutes for the similarity search for the sampled time series via the FFT-based algorithm. The DTW method was only executed once, therefore no repeated measurements have been taken to confirm the execution time of the entire data set. However, repeated measurements have been taken to evaluate the performance of finding the best match for a single template series in the UCR time series pool. The results vary depending on the length of the time series but take on average 180 seconds for the UCR time series pool in our Python implementation for a single series. This operation is not parallelized but is executed on a single core. For the FFT-based algorithm computing the results for single template series across the UCR data set averages given all windows takes roughly

0.016 seconds on the 8-core machine. The results for the entire test set was run with parallelization but the search for one template series is executed on a single core. The difference amounts to an performance difference factor of roughly 11.250. The given results for the execution time include additional time complexity as well as wait times for writing files to the hard drive. Those additional steps are more for the FFT-based scenario and depend also some additional constraints used there. These steps could be left out if we are not interested in analyzing the intermediate steps. This would significantly increase the time performance advantage of the FFT-based method as the majority of time spent on reading and writing data from and to the hard drive.

To generalize these is results it is pertinent to look at time complexity of the two algorithms. For this multiple things need to be reviewed. For Dynamic Time Warping the case is straightforward, because only $\mathcal{O}(m * n)$ needs to be considered for the number of l series in the time series pool for:

$$\mathcal{O}(\sum_{i=0}^{l}(m_i * n)) \tag{6.1}$$

The similarity search based on FFT proposed in this work has multiple components consisting of the transformation of the time series, computing the summary statistics and running the comparison. The setup of the transformed time series pool and the summary statistics computation for said pool is not considered. The transformation of the template series to the frequency domain is combined with the computation of the summary statistics including also the linear fit to find the slope of the series. It can be noted as:

$$\mathcal{O}(o(n \log n) + pn) \tag{6.2}$$

with p being the number of computations that have complexity $n \log n$ and p the number of computations having complexity n. Of course, the constants are ignored and only the worst term is kept for the asymptotic behavior of this step resulting in:

$$\mathcal{O}(n \log n) \tag{6.3}$$

For the second step we compute the matching score of the frequencies and rank them accordingly. This is special insofar as the time complexity is linear with $\mathcal{O}(n)$ and n being the number of frequencies to be compared. However, for our analysis this n is constant as we are only comparing the

top k frequencies. This step is done for all series in the time series pool leading to:

$$\mathcal{O}(\sum_{i=0}^{l}(k)) \tag{6.4}$$

Next the delta's between the template time series and each pool time series needs to be computed:

$$\mathcal{O}((o+p-1)l) \tag{6.5}$$

with o and p still representing the number of summary statistics in each time complexity class and -1 because the comparison of frequencies is already captured. The last step is to filter and sort the result set which can be described by the time complexity of a sorting algorithm:

$$\mathcal{O}(l \log l) \tag{6.6}$$

Combining these steps, removing constants and keeping the worst component per variable the time complexity of our similarity search can be described by:

$$\mathcal{O}(l \log l) \tag{6.7}$$

Looking at the individual parts it makes sense that our method is most impacted by the size of the time series pool whereas DTW is impacted by both the size of the pool and the length of the series to be compared.

To be able to visualize the impact of these results we simplify the time complexity of DTW with the assumption that $m_i = n$, meaning that all time series have the same length. This changes the time complexity of DTW to:

$$\mathcal{O}(ln^2) \tag{6.8}$$

We now have only two input variables for the time complexity with n being the number of data points and l the number of series in the pool. The visualization of FFT-based algorithm only has l as input variable and could be done with a 2d-dimensional plot. However, to be able to compare with DTW we visualize it separately with a 3D plot to reveal its general shape (see figure 6.1) before visualizing both time complexities together in figure 6.2. We see the log-linear growth of the compute time that is respective of the length of l.

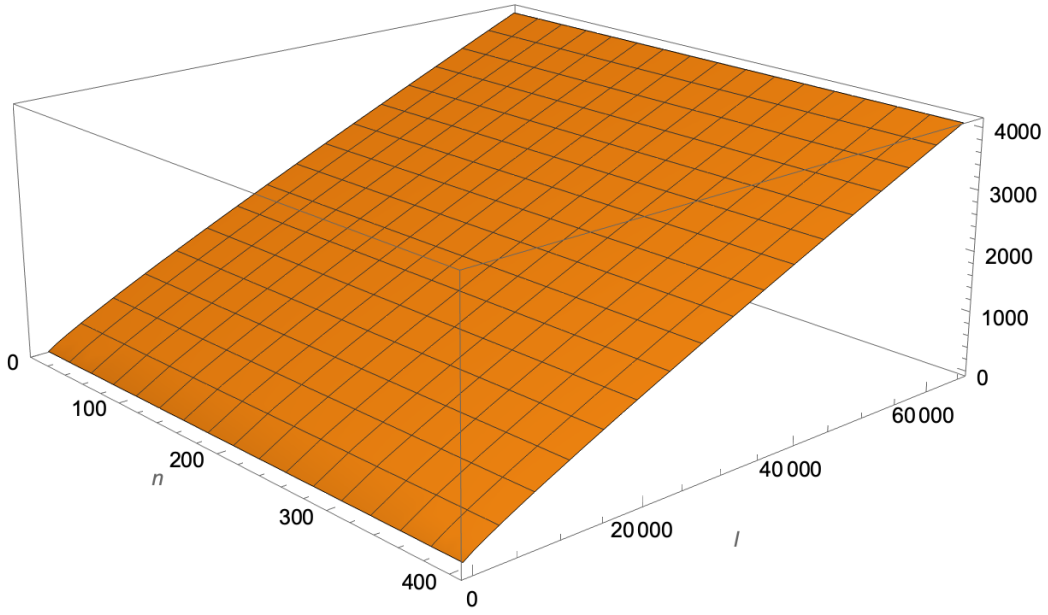

Figure 6.1: Time complexity of our FFT-based similarity search

When visualizing the time complexity of our similarity search algorithm (orange) together with the time complexity of the DTW algorithm (blue) the difference becomes obvious the more we approach the actual values that represent the UCR data set, meaning $n = 401$ and $l \approx 65000$.

It is clearly visible that performance of our proposed method outclasses the DTW-based method. In real-world scenarios this already true for very small values of n and l.

6.3 UCR results overview

The UCR data set provides a unique property that is interesting for a formal evaluation. Each time series in a category, for example ECG5000 in the test data is assigned a class value [5]. The way of classification of the time series is different between the different time series data categories in the repository. For example **(author?)** utilizes other published standards to classify the data into feeding states of insect vectors [38]. This time series repository is part of the UCR archive. In consequence, the data overall follow different principles for classification that cannot be generalized into a single method.

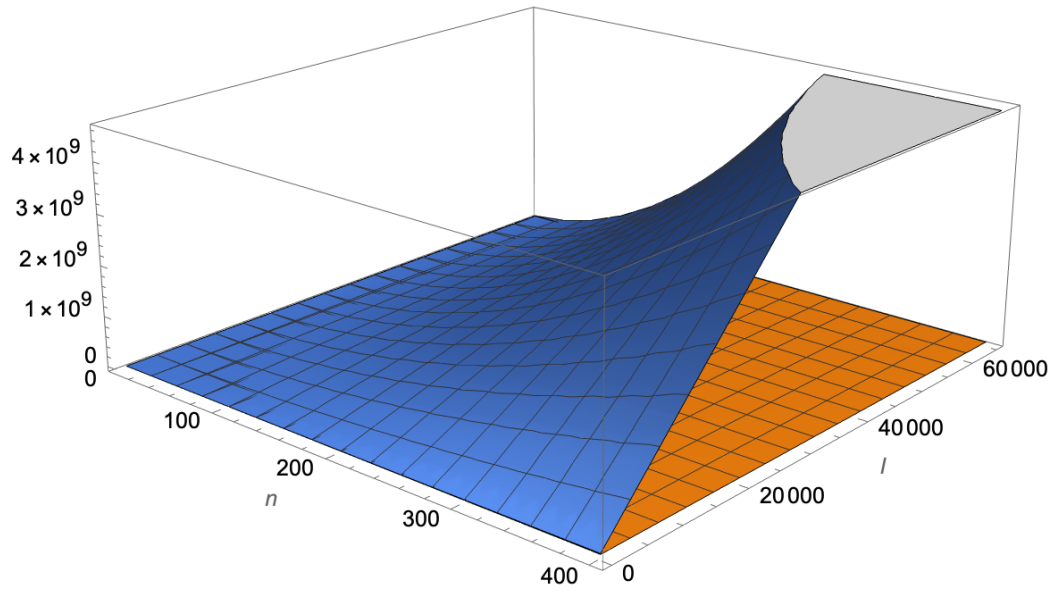

Figure 6.2: Time complexity of DTW and FFT-based algorithm

For Dynamic Time Warping in 63% of the cases (889 out of 1420) from the random sample was found within in the same data category. And from those within the same category 74% of the time series (658/889) where attributed to the same class. 112 categories of 128 original categories in the random sample have a match within the same category.

The results for the FFT-based time similarity have some similarities to the DTW results. The random sample of 1420 time series taken from the test set produced 823 cases of the same data category as result for the best window type and KPI combination (Welch and standard deviation). Recall, that for with the FFT-based method each series is transformed via 3 different window types and augmented with 8 summary statistic KPIs available for selection. This produces 24 results per time series from the random sample. This is indicative of an observation one can make when checking multiple samples of the matching series (see A). Very often the series found via the FFT-based method have a very similar shape and value level to template series and are comparable to the results achieved via the DTW. Important to note is that in order to keep data volume required for storing the match scores on a hard drive with reasonable file size a threshold for the match score was introduced. It was set to 10^3 as the minimum score to be needed to be allowed to be considered a match. However, the lowest score match was 11001 or 10^4+10^3+10^0.

Figure 6.3: Number of matched data categories in UCR repository

Notable for the distribution of the different window types is that their distribution between the different statistics is somewhat random with Welch's window having both the highest total of any metric with standard deviation (823) but having the least in all other categories. And FFT and Hamming are very similar in all categories with the biggest deviation in the trend category. Also interesting to observe is that quantiles, min, and max values tend to perform better to identify the same data category compared to trend and mean.

Where the results between the DTW and our approach differ are in the identification of the classes assigned to the time series. In figure 6.4 one can observe the distribution of matched classes of time series. Of course, this requires that also the data categories are matched. With 41 matches for the combination of Hamming FFT and the series trend as summary statistic this results only in a 5% match of the 755 matches from this analytics combination. Analogous to the data categories, Welch's method is most adequate for finding the same class when contrasted with the other window functions. From the summary statistics the best performing metric is the trend measure to identify the same class. It performs 2.7 times better than the second best measure. This information is useful

to decide which metrics to use when trying to identify the same data class. However, the overall performance of our algorithm is not sufficient to contest the results of Dynamic Time Warping for matching classifications of time series.

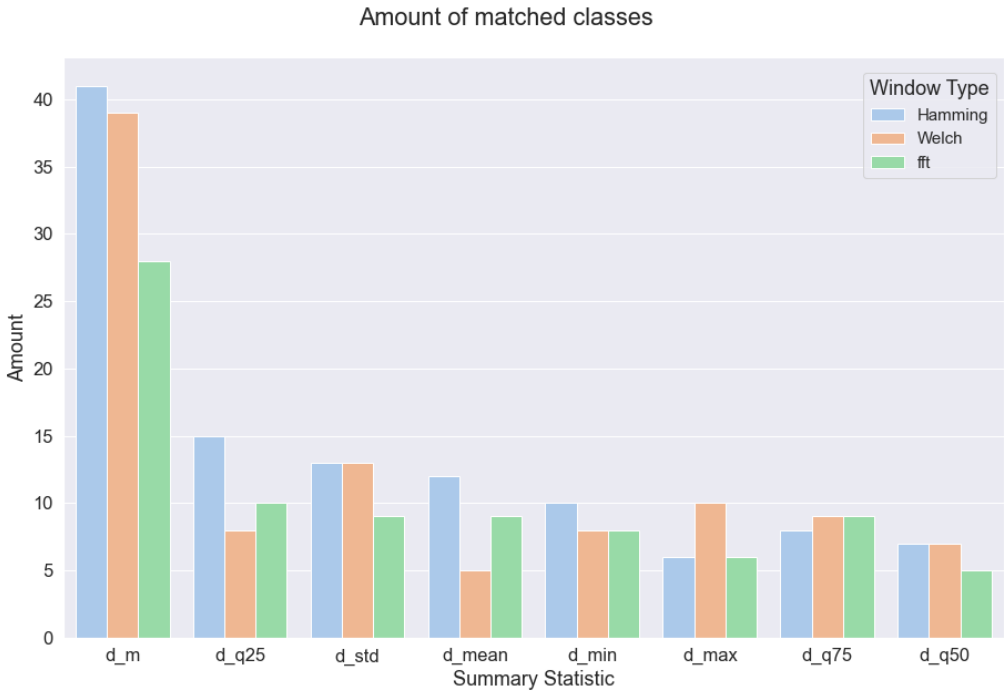

Figure 6.4: Number of matched classes in UCR repository

A visual inspection of the results produced by the different windows and summary statistics (see A) shows that the overall performance of finding similar time series is good with intermittent mishaps. For example figure A.1 and figure A.6 are good examples of the capabilities of our proposed approach.

Nonetheless, there are other examples were some summary statistics have good results but others produce bad results. See for example figure A.9. The data analysis in section 5.0.3 showed the increase of frequencies at the low and high end. However, at the high end our scoring method creates larger and larger intervals at the high frequency end, mismatching high end frequencies that should not be considered to be the same. Possible remedies could be a linear range setup that increase the compute time, or using classical clustering metrics like Euclidean Distance of the ranked frequencies, or creating a range that is sensitive on the low and high boundaries of the

frequency range and less sensitivity in the mid range frequencies. Another interesting observation are figures A.3 and A.5. Here the mechanism of the FFT becomes visible. One can observe that the general patterns of the wave form are matched but the exact place of their occurrence on the domain is not considered. And in the case of figure A.3 one can also observe that the matched time series have higher frequencies that do not match the template series with smaller magnitude and hence lower rank where probably outside of the top k frequency matches and therefore not considered. This could be remedied by increasing K to a higher level.

6.4 Match scores

The first aspect that determines which time series from the pool are considered are the ranked matching scores per transform type τ. The highest ranking scores are kept for each window type and the rest is discarded as described in 4.1.3. For the surviving candidates the Δ of the summary statistics are computed and the ranking is applied. This is done separately as well for each window type. With this result we can review each summary statistic for smallest delta without regard for the window type and declare that resulting time series as the closest match for that particular metric. The distribution of the match scores is visible in figure 6.5. We observe that the results for our random sample are distributed for the most part around 10^4 and higher values. Also here we confirm once again that Welch's method has the largest share of the top scores. It also has the smallest share of low results around the 10^3 score mark. The largest percentage of low scores is captured by the regular FFT, further underlying the effects of spectral leakage and the consequential mismatching of frequencies.

6.5 Window Type

As shown in the in section 6.3 the window type does play an important role in finding adequate results for the matching series.

Each window type can have side effects. As described in section 3.3.8 windows help address the side effects of spectral leakage. However, the window can also lead to other effects. Consider a

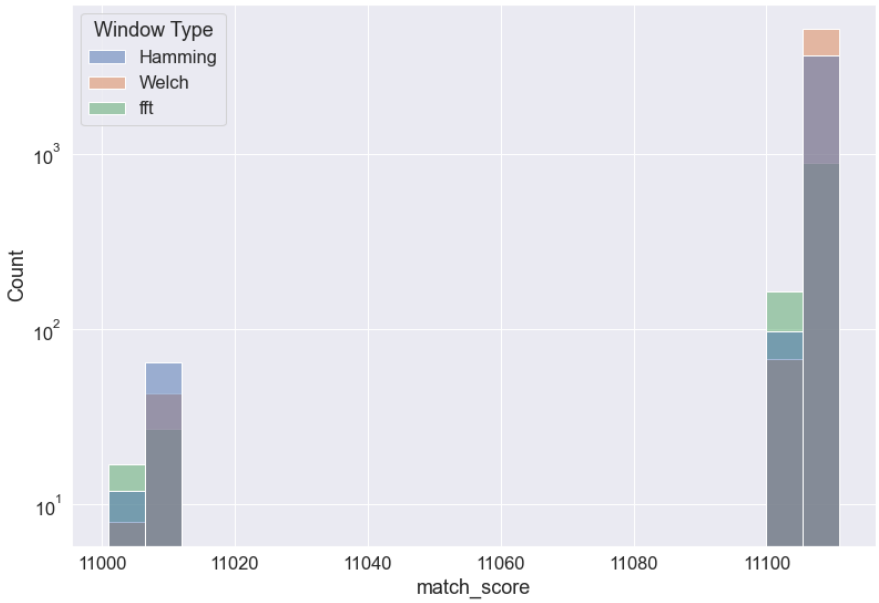

Figure 6.5: Distribution of match scores

template series like in figure 6.6 that has a particular dominant feature like the tooth shape visible in the template series. In the match that considers the median (d_{q50}) as summary statistic we see an interesting result regarding Welch's method. Due to the fact that Welch's method slices the time series into multiple segments, applies a window and then averages the found frequencies one can see how a series that has the same dominant shape but multiple occurrences of it produces a very similar result in the frequency domain. The median in this example must also have a similar value in both series as the ranges are similar. The second tooth in the pool series is just a repetition of the first, therefore the median is not affected significantly. And in consequence one receives a result where a similar pattern repeats and is not in similar place as DTW would identify. Such time series are still considered very similar. In the example here and other comparable setups standard deviation is a good metric to avoid this behavior as the repeating pattern in the matched series is very different in case the same pattern occurs more often than in the template series.

Now whether the result is desirable depends on the context of the analysis or subsequent process the time series are to be utilized for. Therefore, it is expedient to work with different window types

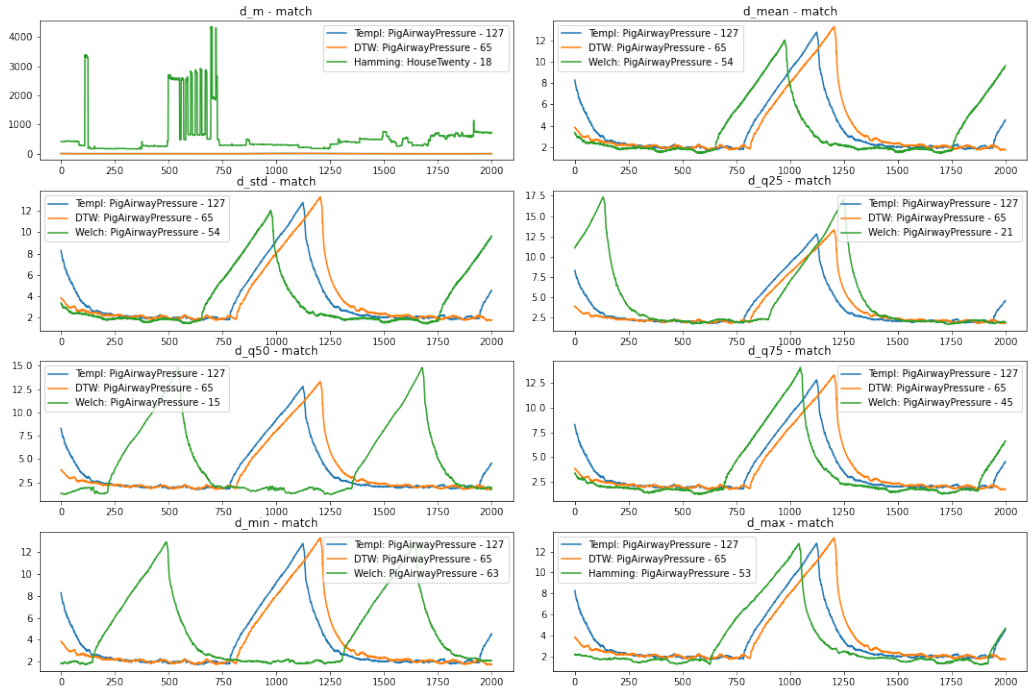

Figure 6.6: Repeating Pattern example - UCR: PigAirwayPressure - 127

and summary statistics adequate for ones analytics requirement when searching for similar time series. Our proposed algorithm offers the flexibility to do so, but also makes a visual inspection of the results important.

Some of the other the side-effects of our similarity search algorithm can be mitigated. For example with frequencies and chosen summary statistic may have a resulting time series from the pool that has a vastly different length compared to the template time series. This can be mitigated by imposing another restriction on the result set of the resulting series having to be within some threshold of length compared to the template. The DTW method mitigates this due to the fact that higher length series will lead to higher overall distance scores and hence will be ranked lower compared to shorter series whose points are also in a similar vicinity compared to the template series.

Chapter 7

Discussion

Due to the quality and the runtime of our algorithm it is feasible to use this method for real-time search engine that not only generates meaningful results of similar series but also allows flexibility in modifying the results in way that optimizes for particular statistical metrics. They can be chosen based on the subsequent data analysis or forecasting task at hand. Furthermore, some of the shortcomings of DTW are also are also addressed. Noise in the data is removed via the utilization of the top frequencies and the essence of each series pattern is captured.

Beneficial for the FFT-based algorithm is its scale-invariance. The Fourier domain is always reduced to the top k frequencies in the power spectrum, therefore the time compare two series is constant irrespective of the length. This advantage makes the analysis virtually independent of the time series length. This makes this application very interesting to fields where large quantities of time series need to be compared, like in financial analysis, or bio statistics, and others.

The frequency domain does not hold easily discernible information about the temporal domain. Therefore, it does not reveal information on trends, change points and the like. If those information are relevant to the time series comparison they need to be captured by the additional summary statistics. If computing such metrics is computationally expensive the overall effort in comparing the series increases. However, many scenarios can be covered with the simple and computationally cheap metrics covered in this thesis. This leads to another aspect that has to be carefully managed when utilizing the method proposed here. There is no single metric that guarantees good or even

meaningful results. As the analytical requirements are different from task to task are different so are the metrics that are most helpful in identifying similar time series. This cannot be avoided by the steps presented here. Other methods, like Dynamic Time Warping provide such straightforward and easily rankable information. Again, whether those are applicable is a question of the analytical task at hand.

In its current implementation the time series similarity search algorithm produces inconsistent results. Some matches are spot on or resemble the results produced by DTW. Without reviewing the results from multiple statistics and window types a user may receive a time series that is not similar. Especially if a user requires quick or many results without wanting to review the each result manually a further improvement of the algorithm is required. We believe that finding a better way of reflecting the higher frequencies in the selection process without taking the middle frequencies to much into account will help solve this issue and produce more consistent results.

Aspects that should be considered in future work are finding a matching score mechanism that captures the high end frequency intervals in more detail than the method presented in this work. In our thesis the raw time series were transformed to the frequency space. Future work should review the results when applying pre-processing steps like denoising and/or detrending the raw time series. Another aspect is the verifying whether averaging the match scores between the different window types improves the consistency of the results. Also working on smaller subsets of the data sets comparing results within different data categories may review interesting insights into our proposed algorithm.*

Chapter 8

Conclusion

In this work we have shown that our proposed algorithm based on the Fast Fourier Transform (FFT) with frequency match scores augmented by summary statistics can be utilized to identify similar time series from a pool for a provided template works very fast compared to the standard method of Dynamic Time Warping (DTW) and creates very good overall results. Additionally we have applied multiple windows to the data set to deal with the inherent short comings of the FFT itself and the various window techniques. The way of integrating the summary statistics gives a user the flexibility to adjust the algorithm to the needs of her specific analytics task for which the similar time series are to be found. This is in stark contrast to the Dynamic Time Warping mechanism is an accepted method of finding similar time series. The results of our proposed time series search algorithm are on average inconsistent compared to DTW but can possibly be improved further.

The purpose of building a time series search engine that retrieves similar time series quickly with flexible flexible search optimization is achieved with our algorithm. Results can be retrieved in sub seconds. This is a feat that would not be possible utilizing DTW for the same purpose.

The downside of utilizing our algorithm are that in its current form there are side effects which need to be considered, like retrieving series that have repeating patterns that do match in quantity or location depending on the used window type.

Appendix A

Visualizations of results

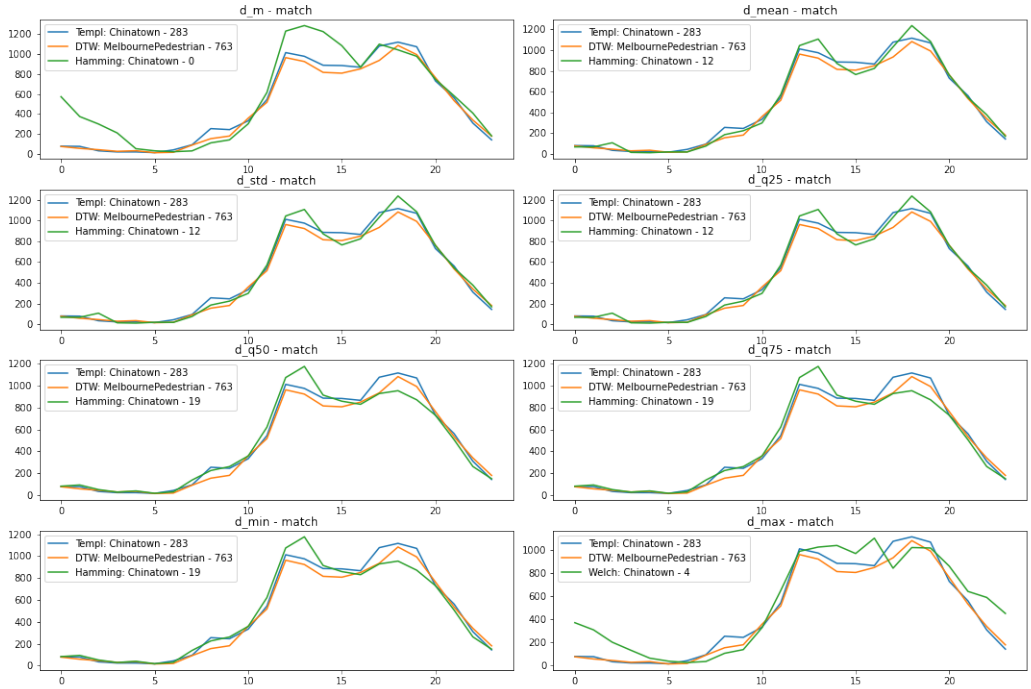

Figure A.1: Chinatown - 283

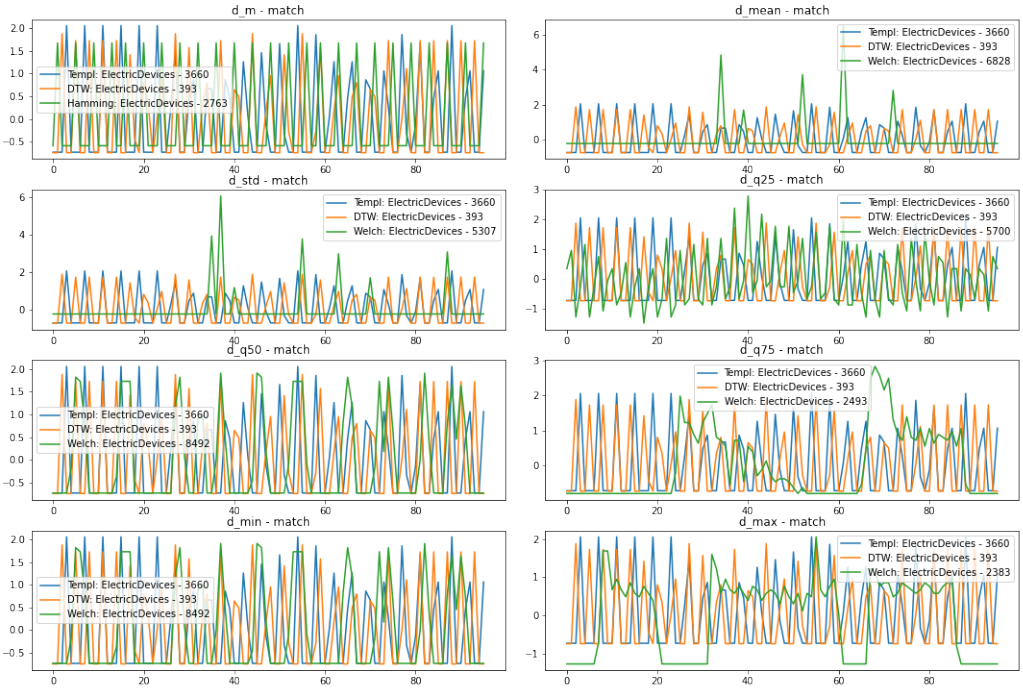

Figure A.2: ElectricDevices - 3660

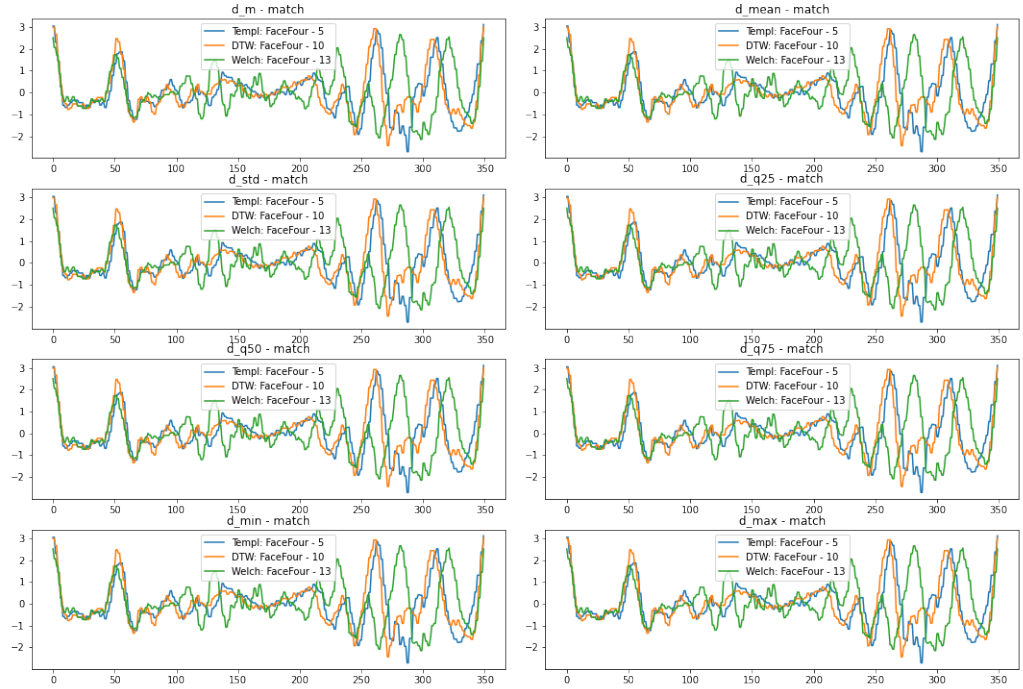

Figure A.3: FaceFour - 5

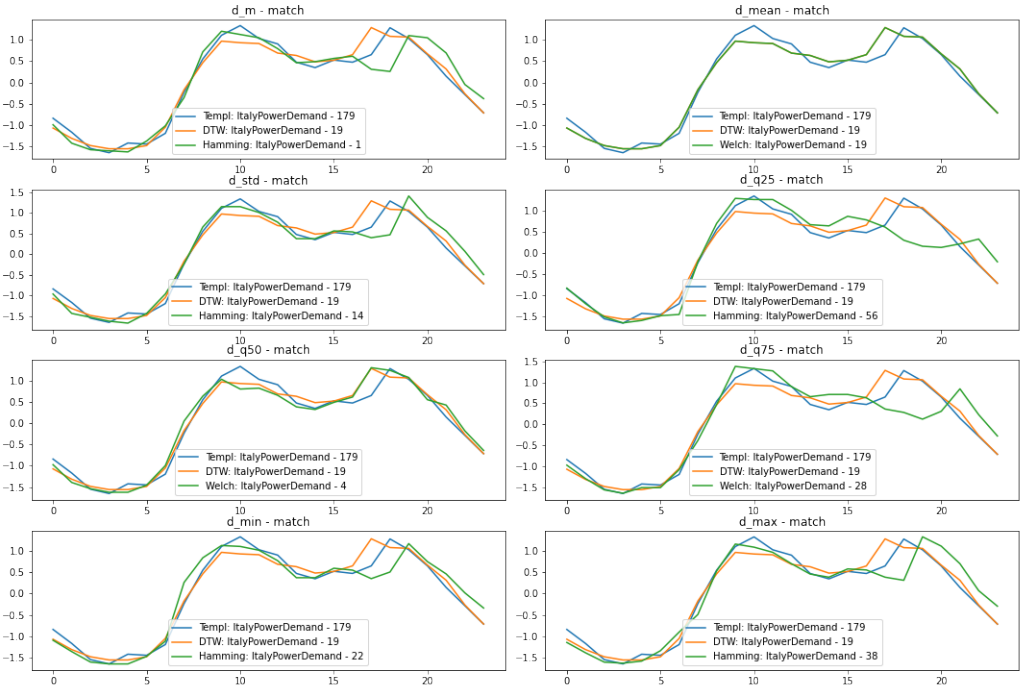

Figure A.4: ItalyPowerDemand - 179

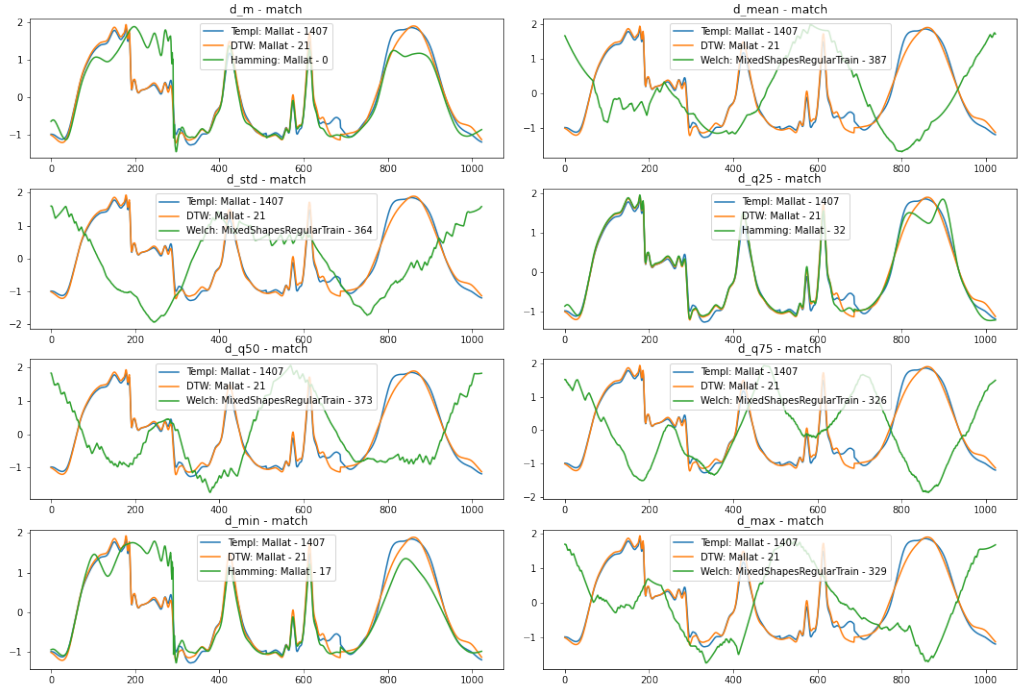

Figure A.5: Mallat - 1407

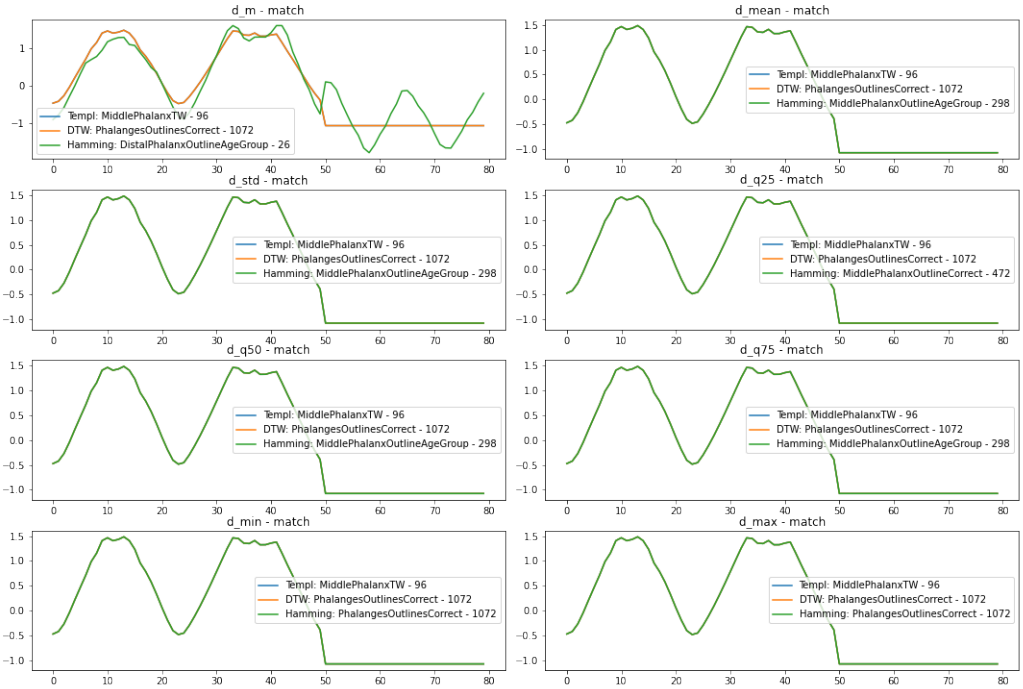

Figure A.6: MiddlePhalanxTW - 96

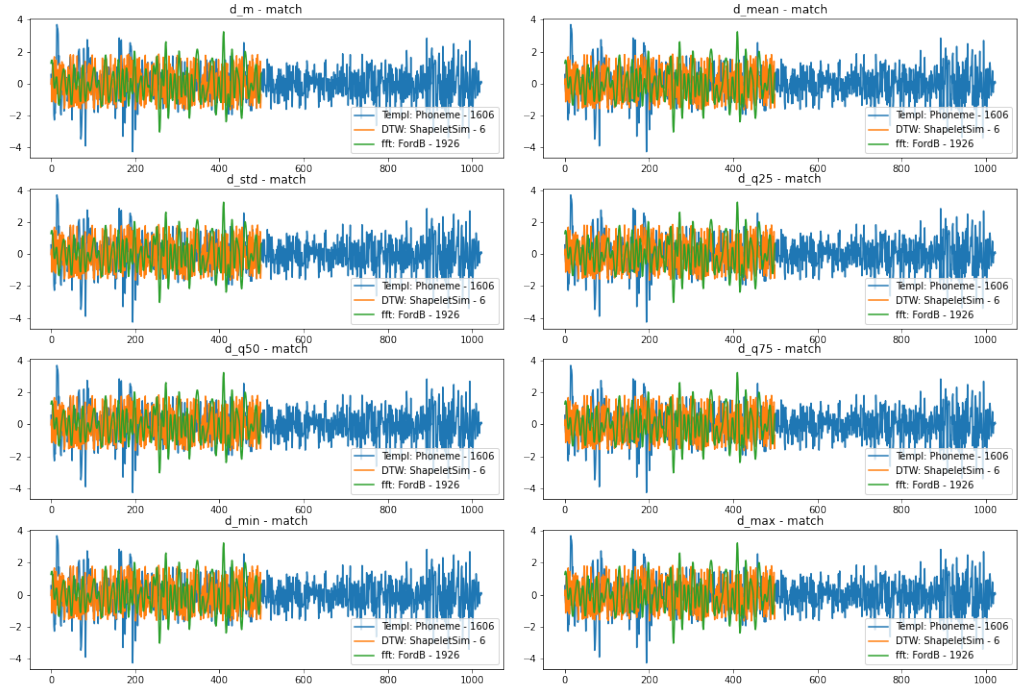

Figure A.7: Phoneme - 1606

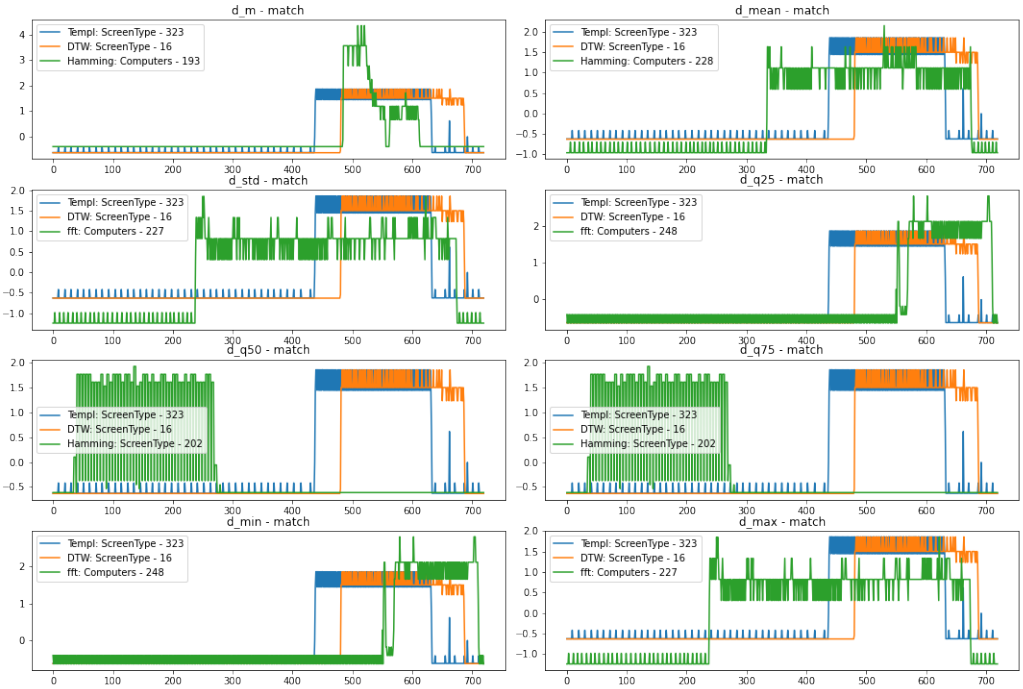

Figure A.8: Screensaver - 323

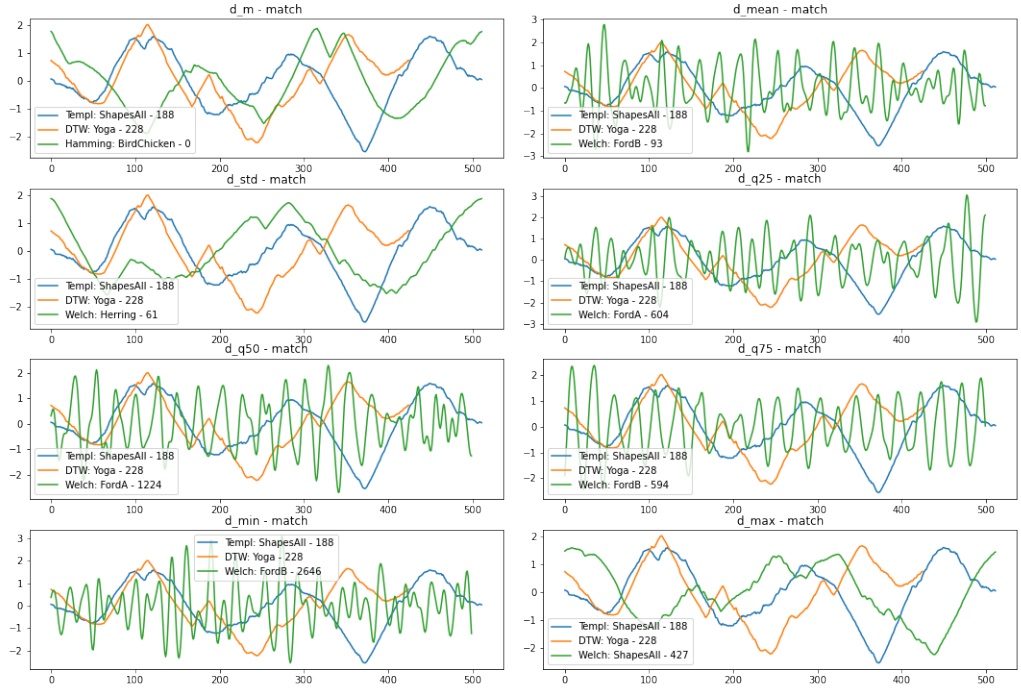

Figure A.9: ShapesAll - 188

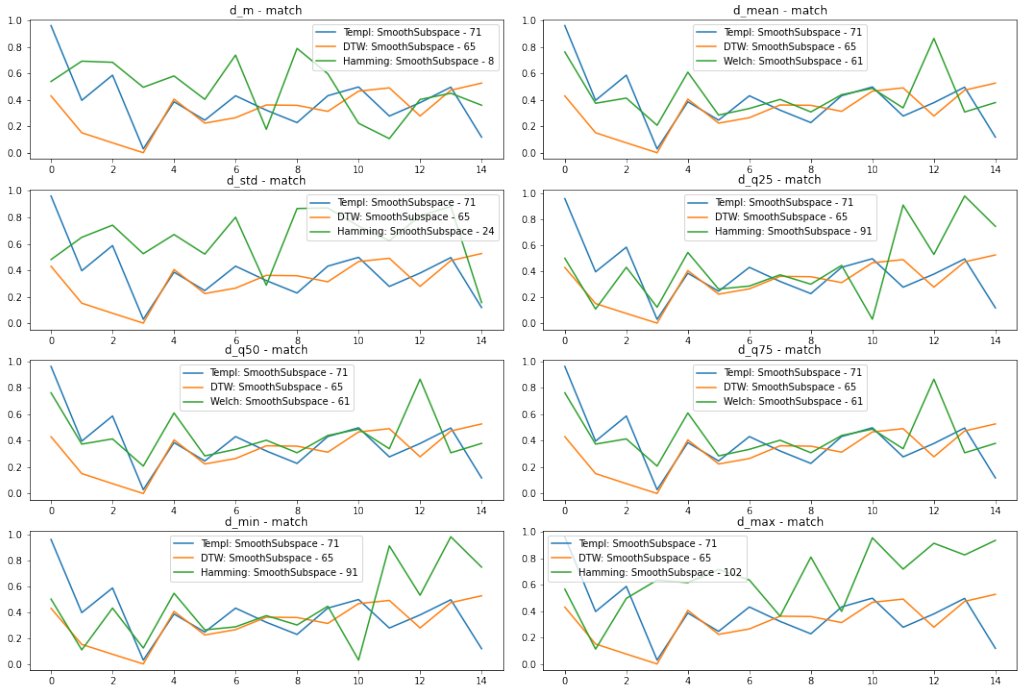

Figure A.10: SmoothSubspace - 71

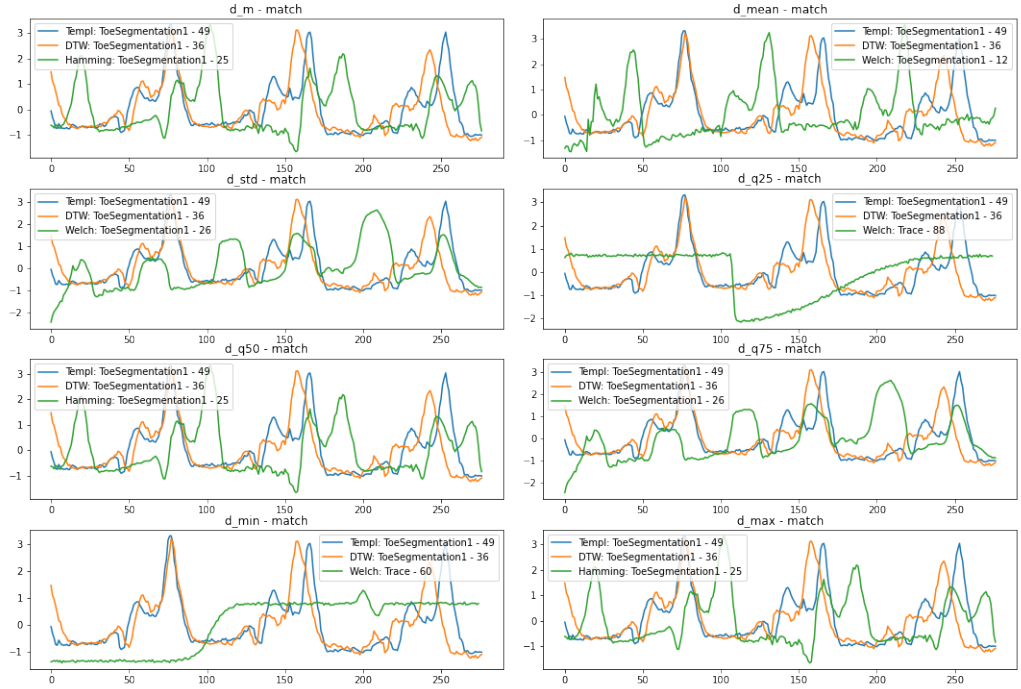

Figure A.11: ToeSegmentation1 - 49

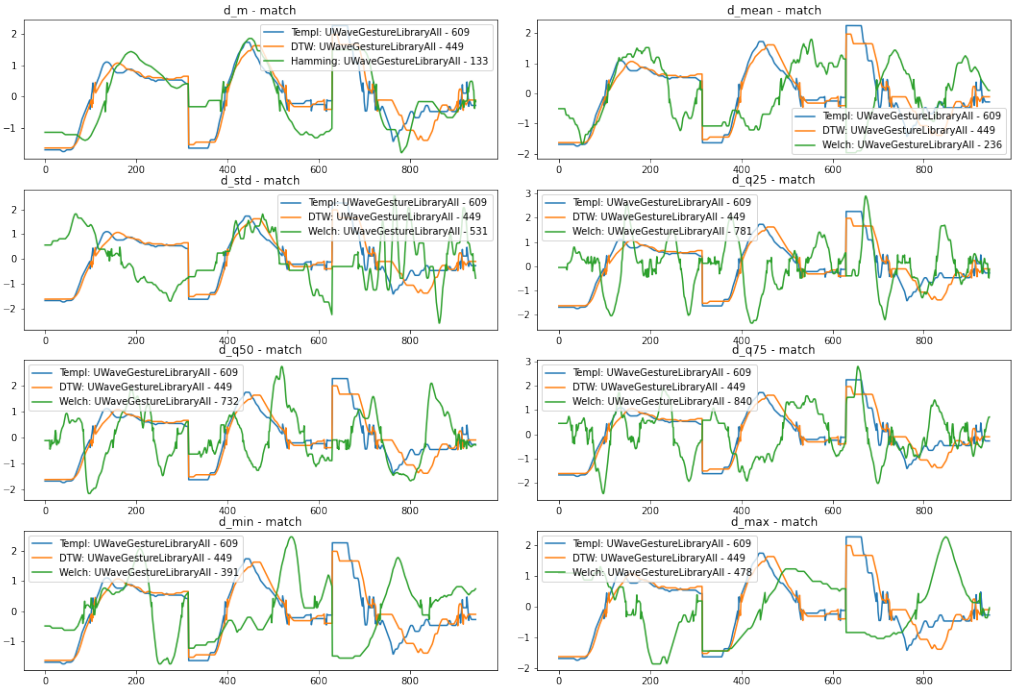

Figure A.12: UWaveGestureLibraryAll - 609

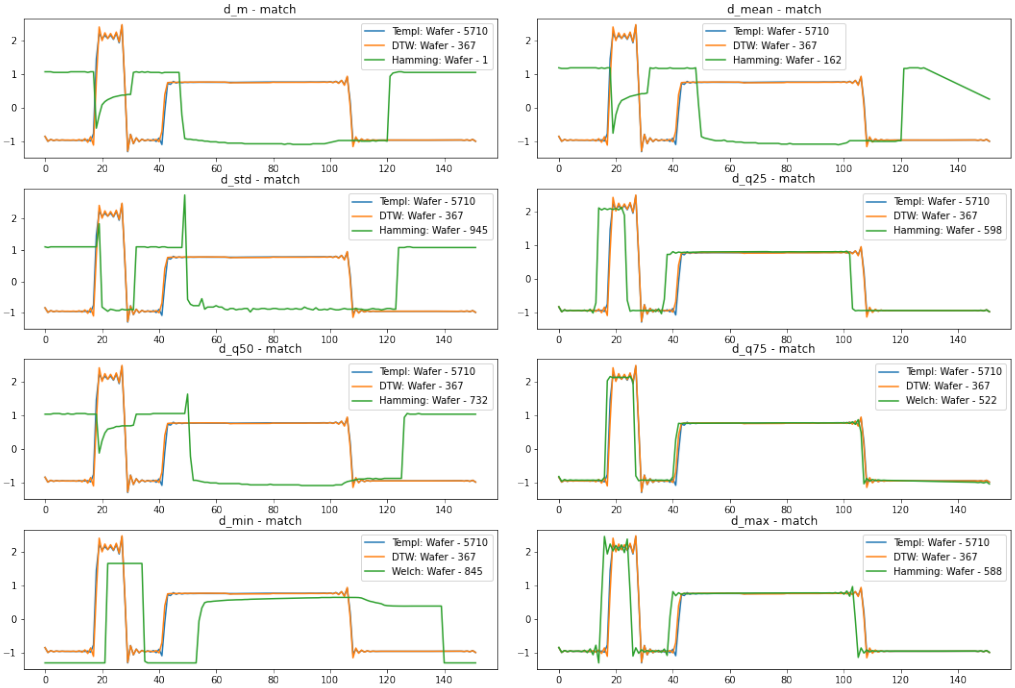

Figure A.13: Wafer - 5710

References

[1] H. Anton. *Calculus - A New Horizon*. Number v. 6 in Calculus. Wiley, 1999.

[2] M. S. Bartlett. Smoothing periodograms from time-series with continuous spectra. *Nature*, 161(4096):686–687, May 1948.

[3] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. KDD Workshop, pages 359–370, 1994.

[4] S. Brunton and J. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[5] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, Nov 2019.

[6] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

[7] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. *Proceedings of the 1994 ACM SIGMOD international conference on Management of data - SIGMOD '94*, 1994.

[8] J. Fourier and F. p. . f. Didot. *Théorie analytique de la chaleur, par M. Fourier*. chez Firmin Didot, pere et fils, 1822.

[9] T.-c. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, Feb 2011.

[10] S. Gharghabi, S. Imani, A. Bagnall, A. Darvishzadeh, and E. Keogh. An ultra-fast time series distance measure to allow data mining in more complex real-world deployments. *Data Mining and Knowledge Discovery*, 34(4):1104–1135, May 2020.

[11] T. Giorgino. Computing and visualizing dynamic time warping alignments inr: Thedtwpackage. *Journal of Statistical Software*, 31(7), 2009.

[12] T. Giorgino. Computing and visualizing dynamic time warping alignments inr: Thedtwpackage. *Journal of Statistical Software*, 31(7), 2009.

[13] F. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

[14] R. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2014.

[15] R. J. Hyndman. A brief history of forecasting competitions. *International Journal of Forecasting*, 36(1):7–14, Jan 2020.

[16] K. Jonasson, S. Sigurdsson, H. F. Yngvason, P. O. Ragnarsson, and P. Melsted. Algorithm 1005. *ACM Transactions on Mathematical Software*, 46(1):1–20, Apr 2020.

[17] Q. Kang, C. Yu, Y. Zhang, C. Cui, C. Sun, J. Xiao, and S. Tang. Astro-ts3: Time-series subimage search engine for archived astronomical data. *Astronomy and Computing*, 34:100428, Jan 2021.

[18] E. Keogh and S. Kasetty. *Data Mining and Knowledge Discovery*, 7(4):349–371, 2003.

[19] E. J. Keogh and M. J. Pazzani. A simple dimensionality reduction technique for fast similarity search in large time series databases. *Lecture Notes in Computer Science*, page 122–133, 2000.

[20] S. Kumar, K. Singh, and R. Saxena. Analysis of dirichlet and generalized “hamming” window functions in the fractional fourier transform domains. *Signal Processing*, 91(3):600–606, Mar 2011.

[21] D. Li, Y. Zhao, and Y. Li. Time-series representation and clustering approaches for sharing bike usage mining. *IEEE Access*, 7:177856–177863, 2019.

[22] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery - DMKD '03*, 2003.

[23] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, Apr 2007.

[24] S. Makridakis. M4 competition data archive, January 2018. `https://mofc.unic.ac.cy/m4/`.

[25] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting*, 1(2):111–153, Apr 1982.

[26] S. Makridakis, M. Hibon, and C. Moser. Accuracy of forecasting: An empirical investigation. *Journal of the Royal Statistical Society. Series A (General)*, 142(2):97, 1979.

[27] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, Jan 2020.

[28] E. Musk. An integrated brain-machine interface platform with thousands of channels. *Journal of Medical Internet Research*, 21(10):e16194, Oct 2019.

[29] J. Pang, D. Liu, Y. Peng, and X. Peng. Intelligent pattern analysis and anomaly detection of satellite telemetry series with improved time series representation. *Journal of Intelligent & Fuzzy Systems*, 34(6):3785–3798, Jun 2018.

[30] K. Pearson. On lines and planes of closest fit to systems of points in space. *Phil. Mag.*, 6(2):559–572, 1901.

[31] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in FORTRAN 77 Macintosh Diskette Version 2.0: The Art of Scientific Computing*. Fortran numerical recipes. Cambridge University Press, 1992.

[32] C. A. Ratanamahatana and E. Keogh. Making time-series classification more accurate using learned constraints. *Proceedings of the 2004 SIAM International Conference on Data Mining*, Apr 2004.

[33] J. Ratcliffe. *Foundations of Hyperbolic Manifolds*. Graduate Texts in Mathematics. Springer New York, 2006.

[34] J. H. Siegle, A. C. López, Y. A. Patel, K. Abramov, S. Ohayon, and J. Voigts. Open ephys: an open-source, plugin-based platform for multichannel electrophysiology. *Journal of Neural Engineering*, 14(4):045003, Jun 2017.

[35] E. Spiliotis, S. Makridakis, A. Kaltsounis, and V. Assimakopoulos. Product sales probabilistic forecasting: An empirical evaluation using the m5 competition data. *International Journal of Production Economics*, 240:108237, Oct 2021.

[36] Y. Tang, Y. Xie, X. Yang, J. Niu, and W. Zhang. Tensor multi-elastic kernel self-paced learning for time series clustering. *IEEE Transactions on Knowledge and Data Engineering*, page 1–1, 2019.

[37] 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 and Knowledge Discovery*, 26(2):275–309, Feb 2012.

[38] D. S. Willett, J. George, N. S. Willett, L. L. Stelinski, and S. L. Lapointe. Machine learning for characterization of insect vector feeding. *PLOS Computational Biology*, 12(11):e1005158, Nov 2016.

[39] C. Zhang, J. Luo, S. Zhang, and C. Zhang. Introduction to time series search engine systems. *2012 International Conference on Systems and Informatics (ICSAI2012)*, May 2012.

[40] Z. Zhang, P. Tang, and T. Corpetti. Time adaptive optimal transport: A framework of time series similarity measure. *IEEE Access*, 8:149764–149774, 2020.

[41] K. Åström. On the choice of sampling rates in parametric identification of time series. *Information Sciences*, 1(3):273–278, Jul 1969.