Time Series: Defining a Search Engine

Philipp Beer

October 14, 2021

1 Purpose

The purpose of this thesis is to explore the possibility of creating a time series series search engine.

2 Introduction

Time series is often described as "anything that is observed sequentially over time" which usually are observed at regular intervals of time [1]. 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_1 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 their work [2] categorized time series research into (1) representation, (2) indexing, (3) similarity measure, (4) segmentation, (5) visualization and (6) mining. Research in these different fields started taking off in the second half of the 20^{th} century. For example in [3] 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 datasets in this domain are large. Through this time series data incorporates the similar obstacles as high dimensional data, namely the "curse of dimensionality" [4] and requiring large computational efforts in order to generate insights. And as will be discussed in 2.1 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 a algorithm allowing the fast and meaningful comparison of an input time series or template against a vast array time series. Within the research many different areas and approaches have been attempted (at list here). However, there is a tendency to apply the simplest methods possible to achieve the desired results. For time series those are mainly Euclidean Distance and Dynamic Time Warping. While those methods will be explored in section 3 it can be said that those methods are simple, easy to understand and produce mostly reliable results in their respective application domains. Good performance is not their strong suit. Therefore, our approach is targeted to achieving comparable capability in identifying similar series, while achieving it with a significant reduction in computational complexity.

2.1 Applications

Time series are encountered 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 mentioned. Or sensors which are utilized in the automotive industry or heavy machinery where they are employed to capture information for predictive maintenance applications.

In the financial industry time series are a very fundamental component of decision making, like 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.

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.

In engineering fields the utilization is often times similar to the above but it also require 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. Musk describes a recording system with 3072 electrodes generating time series data [5] that is used to capture the brain information and visualized in real-time [6].

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.

2.2 Organization of this thesis

The rest of this thesis is organized as follows:

2.3 TODO to be integrated

- refer to previous work on measures of similarity and outcome
- measure of similarity required
- challenges with time series (domains, granularity, length, outliers)
- area of signal processing interesting methods

3 Related work

Related work addressing the idea of time series search engine focuses on the system architecture and the data processing and pipelining aspect of this such an architecture [7]. However, in 2000 Keogh and Pazzani also applied a dimensionality reduction technique (Piecewise Constant Approximation) to execute fast search similarity search in large time series databases. Other papers address domain specific questions like the introduction of a "Time-series Subimage Search Engine for archived astronomical data" [9].

In order to be able to describe the closeness of time series or multiple time series to each 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. 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) [11]. 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 based measures like Longest Common Subsequence (LCSS), Edit Sequence on Real Sequence (EDR), Swale and Edit Distance with Real Penalty. An example for 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 the most common measures into: (1) Euclidean Distance, (2) Dynamic Time Warping (DTW), (3) Least Common Subsequence (LCSS), and (4) K-Shape.

Dynamic Time Warping (DTW) is an elastic measure. It has been introduced by 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. This gives it the capability to warp peaks or valleys between different time steps of the two series as the resulting distance metric. As will be shown in section 4.2 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 4.1 - and Dynamic Time Warping (DTW) - more details can be found in section 4.2 - and computationally preferable. 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 resulting algorithms that analyze and compare the time series.

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 [14]. 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.

3.1 Dimensionality Reduction related to Singular Value Decomposition

Singular Value Decomposition is a fundamental matrix factorization technique with a plethora of applications and use cases. 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 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 [16]. 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. The main idea of PCA is to apply the SVD to a dataset centered around zero and subsequently computing the covariance of the centered dataset. 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 dataset. 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 dataset.

The Fast Fourier Transform (FFT) is based upon the Fourier Transform introduced by Joseph Fourier in early 19^{th} century to analyze and analytically represent heat transfer in solid objects [18]. This transform is a fundamental component of modern computing and science in general. It has transformed how technology can be used in the in 20^{th} century in areas such as image and audio compression and transfer. The concept will be introduced in more detail in section 4.3. 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 to the transformed data to few key components is the same as in SVD and PCA.

3.2 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 aggegrate approximation (PAA) and a conversion of the results via a lookup table [19]. 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:

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

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

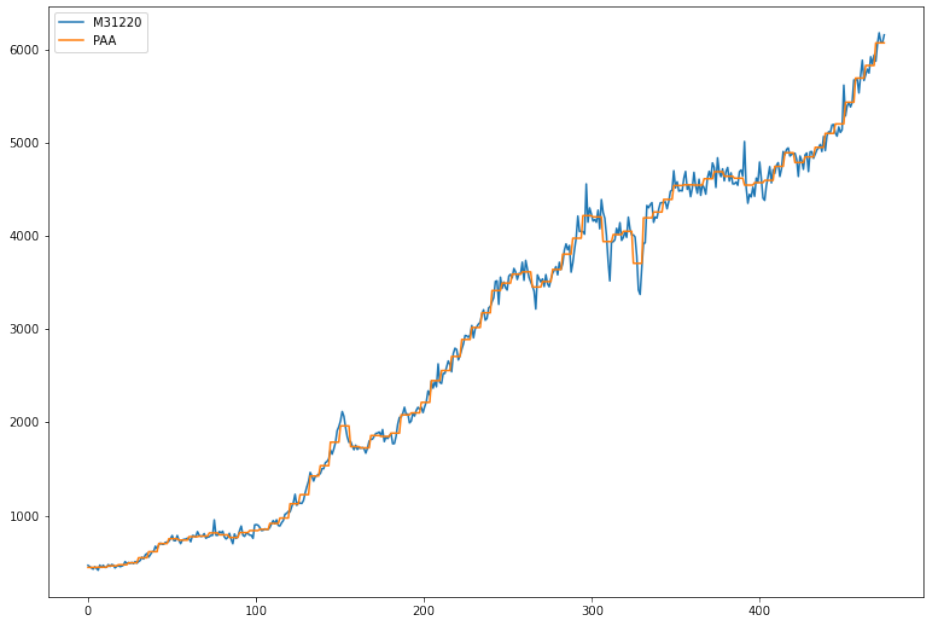

Figure 1: Piecewise Aggregate Approximation - M4 example: M31220 (window size - 6)

example from the M4 dataset can be seen in figure 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 computated via PAA to a symbolic representation of a letter. The used discretization should accomplish equiprobability in the assignments of the symbols [20]. The authors show by example of taking subsequences of length 128 from 8 different time series that the resulting PAA transformation has a Gaussian dstribution. 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) [20]. Table 1 shows the value ranges for values of a from 3 to 10 and has been reproduced from [20].

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

Table 1: Lookup table - reproduced from Lin et al.

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

4 Underlying Concepts

This section gives an overview of the concepts utilized in this thesis to generate the baseline performance of the algorithm against which our

4.1 Euclidean Distance

Euclidean Distance is the most widely used distance metric in the research of time series. It is either used as a metric on its on or a as metric used used inside other methods to compute distances, for example, computation of distance of subsections of the data ([21]) or to compute the distance between various points of two time series (see section 4.2). 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:

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

It is a measure that is easy to compute and comprehend and gives intuitive input for the distance of two time series. From the standpoint of time complexity the algorithm is applicable also to larger datasets with $\mathcal{O}(n)$. Its simplicity also creates some limitations. For example, to compute the euclidean distance between two series their length needs to be the same. Furthermore, it can be easily impacted in its results by the presence of outliers or increased levels of noise. 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 its shortcomings it is a prominent metric and widely used for distance calculations for short comings. Some of its limitations are addressed by more sophisticated metrics that utilize its computation as component.

4.2 Dynamic Time Warping

Berndt and Clifford introduced Dynamic Time Warping in 1994 finding the minimal alignment between two time series computed through a cost matrix and identifying the minimized path through the matrix starting from the final elements of each time series. This warps the points in time between the different series as shown in figure 2.

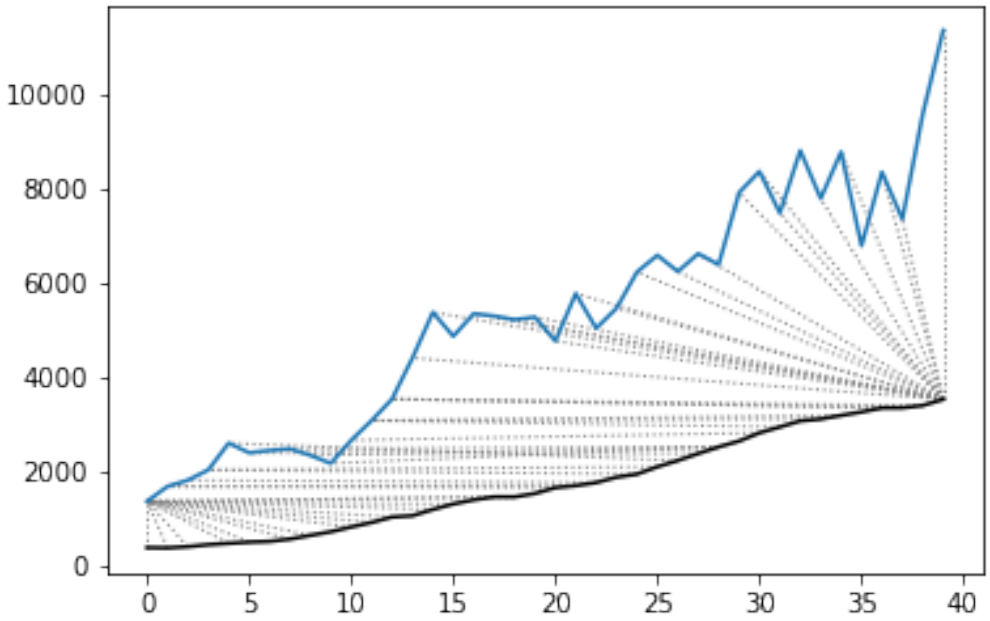

Figure 2: Dynamic Time Warping - M4 Example: Y5683 and Y5376

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 points:

$$m_{ij} = D(s_i, q_j) \tag{3}$$

where 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:

$$\max(m, n) \leq k < m + n - 1 \tag{4}$$

The warping path is constrained with bound with the following condition $p_1 = (1, 1)$ and $p_K = (m, n)$. That means that both 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. This means that 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:

$$\gamma(i, j) = D(s_i, q_j) + \min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\} \tag{5}$$

Therefore, the path can obtained by the following definition:

$$DTW(S, Q) = \min_{P:WarpingPath} \left\{ \sum_{k=1}^{K} \sqrt{p_k} \right\} \tag{6}$$

Figure 3 provides an example for a warping path result.

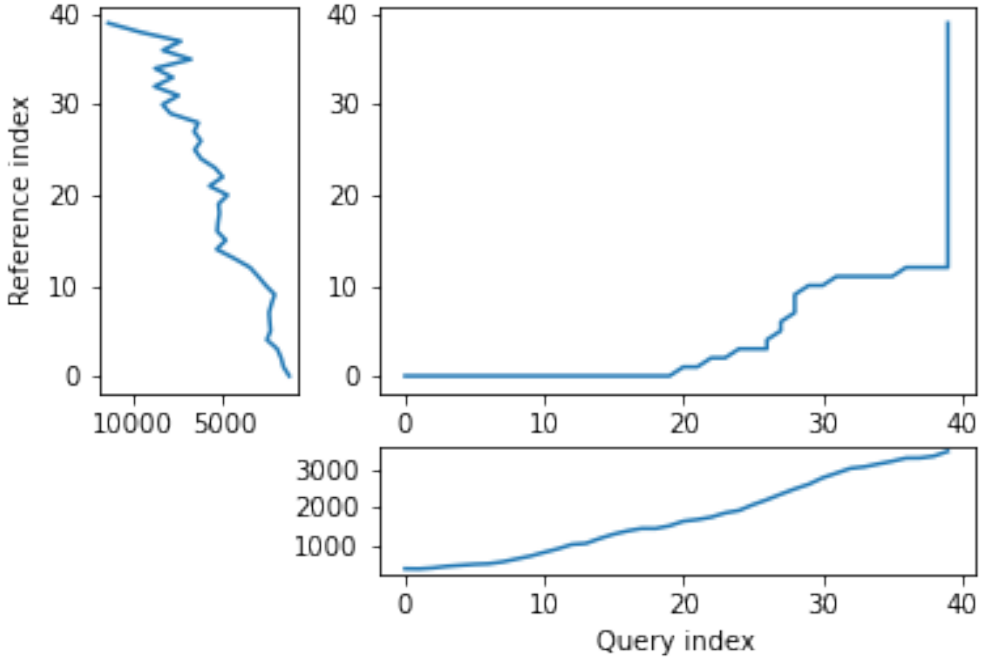

Figure 3: Warping path example - M4 data: Y5683 and Y5376

The challenge with the application of DTW is the time complexity of the algorithm $\mathcal{O}(m * n)$ due to the fact that the distance compuation needs to be executed for each element of each series. Various methods for speed improvements have been introduced. The favorite principle was described by Ratanamahatana and Keogh. They introduced an adjustment window condition that where it is assumed that the optimal path does not drift very far from the diagonal of the cost matrix [22]. However, this does not change the fundamental nature of the algorithm and computing DTW for multiple time series against a database of time series will require days of computation time even on modern computer architectures.

In favor of DTW needs to be stated, that it is flexible with regards to the 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 avaiable 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.

Therefore 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 dataset.

- Similarity through decomposition
  - introduce time series decomposition (reference in [1])
  - trend and seasonality (mention assumptions about period)

4.3 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 properties. The Discrete Fourier transform is based on the Fourier Transform (FT) which concerns itself with the representation of functions which in turn is built upon the Fourier series. We will give a brief introduction to them. However, a thorough introduction can be found in [16]. The principal idea Fourier analysis follows is that it can project functions - i.e. Fourier Transform - and data vectors - i.e. 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.

4.3.1 Inner Product of Functions and their norms

To get to the properties of of data under the Fourier transform we must start with the Hermitian inner product ([23]) of functions in Hilbert spaces, $f(x)$ and $g(x)$ ($\overline{g}$ denotes the complex conjugate of g) in the domain $x \in [a, b]$:

$$\langle f(x), g(x) \rangle = \int_a^b f(x)\, \overline{g}(x) dx \tag{7}$$

This means 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:

$$\langle \vec{f}, \vec{g} \rangle = \sum_{k=1}^{n} f(x_k)\, \overline{g}(x_k). \tag{8}$$

This formula behaves as desired but grows in its value as more and more data points are added. So a normalization is added to counter the effect. The normalization occurs through the domain chosen for the analysis $\Delta x = \frac{b-a}{n-1}$:

$$\frac{b-a}{n-1} \langle \vec{f}, \vec{g} \rangle = \sum_{k=1}^{n} f(x_k)\, \vec{g}(x_k) \Delta x. \tag{9}$$

This corresponds to the Riemann approximation of continuous functions [24]. 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:

$$\|f\|_2 = (\langle f,\, f \rangle)^{\frac{1}{2}} = \sqrt{\langle f,\, f \rangle} = \left(\int_a^b f(x)\, \overline{f}(x) dx \right)^{\frac{1}{2}}. \tag{10}$$

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 [16]. Next we will show how a Fourier series is a projection of a function onto the orthogonal set of sine and cosine functions.

4.3.2 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:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cos(kx) + b_k \sin(kx)). \tag{11}$$

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.

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)dx \tag{12}$$

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(kx)dx \tag{13}$$

$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(kx)dx. \tag{14}$$

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:

$$a_k = \frac{1}{\|\cos(kx)\|^2}\langle f(x),\, \cos(x)\rangle \tag{15}$$

$$b_k = \frac{1}{\|\sin(kx)\|^2}\langle f(x),\, \sin(x)\rangle \tag{16}$$

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:

$$f(x) = \frac{a_0}{2} + \sum\left(a_k \cos\left(\frac{2\pi kx}{L}\right) + b_k \sin\left(\frac{2\pi kx}{L}\right)\right) \tag{17}$$

This modifies the integrals for the coefficients to:

$$a_k = \frac{2}{L}\int_0 Lf(x)\cos\left(\frac{2\pi kx}{L}\right) \tag{18}$$

$$b_k = \frac{2}{L}\int_0 Lf(x)\sin\left(\frac{2\pi kx}{L}\right) \tag{19}$$

One can write the formula utilizing Euler's formula

$$e^{ikx} = \cos(kx) + i\sin(kx), \tag{20}$$

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

$$\begin{aligned} f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx} &= \sum_{k=-\infty}^{\infty}(\alpha_k + i\beta_k)(\cos(kx) + i\sin(kx)) \\ &= (\alpha_0 + i\beta_0) + \sum_{k=1}^{\infty}\left[(a_{-k} + a_k)\cos(kx) + (\beta_{-k} - \beta_k)\sin(kx)\right] + \\ &\quad i\sum_{k=1}^{\infty}\left[(\beta_{-k} + \beta_k)\cos(kx) - (\alpha_{-k} - \alpha_k)\sin(kx)\right]. \end{aligned} \tag{21}$$

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 theb 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:

$$\langle \psi_j, \psi_k \rangle = \int_{-\pi}^{\pi} e^{jkx} e^{-ikx} dx = \begin{cases} \text{if } j \neq k & \int_{-\pi}^{\pi} e^{i0x} = 2\pi \\ \text{if } j = k & \int_{-\pi}^{\pi} e^{i(j-k)x} = 0 \end{cases} \tag{22}$$

When $j = k$ the integral reduces to 1, leaving 2π as the result of the interval to be integrated. In cas $j \neq k$ 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:

$$\langle \psi_j, \psi_k \rangle = 2\pi\delta_{jk}. \tag{23}$$

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 infinitie orthognal-vector space space that is spanned by cosine and sine functions:

$$f(x) = \sum_{k=-\infty}^{\infty} c_k \psi_k(x) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \langle f(x), \psi_k(x) \rangle \psi_k(x). \tag{24}$$

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

4.3.3 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:

$$\begin{aligned} f(x) &= \frac{a_0}{2} + \sum_{k=1}^{\infty} \left[a_k \cos\left(\frac{k\pi x}{L}\right) + b_k \sin\left(\frac{k\pi x}{L}\right) \right] \\ &= \sum_{k=-\infty}^{\infty} c_k e^{\frac{ik\pi x}{L}} \end{aligned} \tag{25}$$

$$c_k = \frac{1}{2L} \langle f(x), \psi_k \rangle = \frac{1}{2L} \int_{-L}^{L} f(x) e^{-\frac{ik\pi x}{L}} dx. \tag{26}$$

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:

$$f(x) = \lim_{\Delta\omega \to 0} \sum_{k=-\infty}^{\infty} \frac{\Delta\omega}{2\pi} \int_{-\frac{\pi}{\Delta\omega}}^{\frac{\pi}{\Delta\omega}} f(\xi) e^{-ik\Delta\omega\xi} d\xi \, e^{ik\Delta\omega x}. \tag{27}$$

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, $\hat{f} \triangleq \mathcal{F}(f(x))$:

$$\hat{f}(\omega) = \mathcal{F}(f(x)) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx \tag{28}$$

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

$$f(x) = \mathcal{F}^{-1}(\hat{f}(\omega)) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{i\omega x} d\omega. \tag{29}$$

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.

4.3.4 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:

$$\hat{f}_k = \sum_{j=0}^{n-1} f_j e^{-2\pi jk/n}, \tag{30}$$

$$f_k = \frac{1}{n} \sum_{j=0}^{n-1} \hat{f}_j e^{i2\pi jk/n}. \tag{31}$$

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 can be (and is) heavily used for physical interpretations for example in engineering questions as well.

4.3.5 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 dataset, which will be introduced in section 5.2.1. The longest series has $n = 9919$ datapoints. 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 to time series it still provides very fast computation times.

To be able to convert the DFT to the FFT a multiple of 2 datapoints is required. For example, take $n = 2^6 = 64$. In this case the DFT matrix can be written as follows:

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

where $\mathbf{I}_{32}$ is the Identity matrix. $\mathbf{f}_{\text{even}}$ 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 excuted 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.

4.3.6 Power Spectrum

One important property of time series transformed 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 from which frequencies 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 [25].

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

$$\begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix} \xrightarrow{FFT} \begin{bmatrix} \hat{f}_0 \\ \hat{f}_1 \\ \vdots \\ \hat{f}_n \end{bmatrix} \tag{33}$$

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

$$\hat{f}_j \triangleq \lambda \tag{34}$$

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

$$\hat{f}_j = \|\lambda\|^2 = \lambda\overline{\lambda} = (a + ib)(a - ib) = a^2 + b^2. \tag{35}$$

This is the magnitude of the particular frequency. In figure 4 an exemplartory time series from the M4 dataset (see section 5.2.1) 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 indidicates the energy contained in the respective frequencies. The x-axis is plotted in log-scale. An important property is the fact that the frequencies in the power spectrum differ depending on the length of the of the time series. A frequency of $k_a = 2$ in a series S_1 length $n_{S_1} = 5$ is equivalent to a frequency $k_b = 4$ in a series S_2 of length $n_{S_2} = 10$.

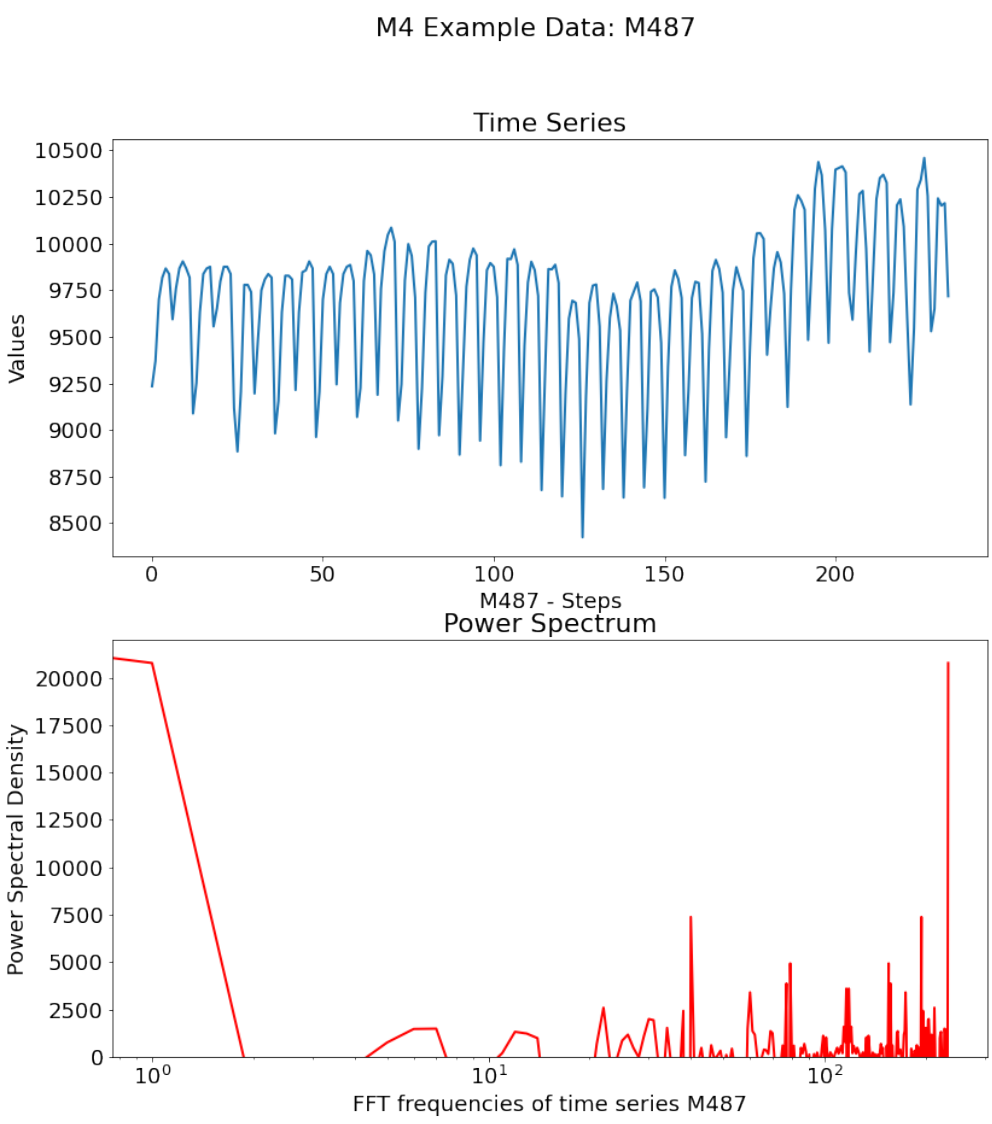

Figure 4: Power Spectrum M4 - Example: M487

4.3.7 Spectral Leakage

The Fast Fourier Transform (FFT) assumes that the signal continues infinitely in time and that there no discontinuities. However, any signal in the real world, including time series has finite data points. If the time domain is an integer multiple of of the frequency k than each records connects smoothly to the next. Real world processes generally do not follow sinusoidal wave forms and can contain significant amounts of noise, as well as phase changes and trends. So if the signal is not an integer multiple of the sampling frequency k this signal leaks into the adjacent frequency bins. See figure 4 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 ranked by energy leve frequencies 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.

5 Time series representation

asdfasdfasf

5.1 Challenges when building a time series

- length of series
- trend
- seasonality
- time complexity -> issue because of data size
- granularity or sampling rates
- noise
- data quality
- similarity is task dependent (level)
- usual need for preprocessing the time series data (denoising, detrending, amplitude scaling) -> any pre-processing does modify the series

5.2 Data Analysis

To develop t

5.2.1 M4 competition data

5.2.2 UCR time series data

5.3 Challenges

- How many frequencies to compare?
- priorities of frequencies (power spectrum)
- different length of time series (leading to different frequencies) - ranges solved with logs

6 Methodology

6.1 Used Data

The research in time series has been numerous and focused on various properties of them as well as finding methods to accurately predict them. Aside of forecasting all researched areas are measures of similarity and retrieval of time series.

- Forecasting In the arena of forecasting the M-competition organized by Prof. Makridakis played a big role in the development of forecasting methods shortly after their inception in 1979.

  One of the aspects that has been correct up until the 5th installment of the M-competition is that statistical methods in forecasting have outperformed more complex machine learning methods. So learning algorithms did not benefit sufficiently from learning from multiple series to generate more accurate point predictions and prediction intervals compared to the statistics-based alternatives.

  One interesting question in this area is whether clustering of time series that have similar properties and training algorithms per cluster of "similar" series can help simplify the learning process for machine learning methods and in consequence improve their performance in future competitions.

  However, expressing similarity for time series is a challenging questions with respect to which metrics to utilize, time complexity as well as limiting assumptions that need to be made for time series.

6.1.1 Data Set Properties

asdfasdfadsf

6.2 Main contribution of the thesis

- transformation into Fourier-space
- transfer frequencies into frequency range band with increasing range width (using log scale)
- computation of frequency energy levels (sort and keep top 5) -> ask Prof. how to name this parameter
- conversion of ordered frequencies into frequency range band
- for each series to compare -> compare whether the frequency matches on the ordered positions -> provide exponential value per position -> match on more powerful frequencies is valued higher

6.3 additional computations

- utilization of FFT utilizes only frequency space (future work should consider comparison of energy levels per frequency)
- additional simple statistics computed (mean, std, quantiles)
- ts decomposition for trend estimation (requires parameter for period) -> then best line fit for slope of the time series

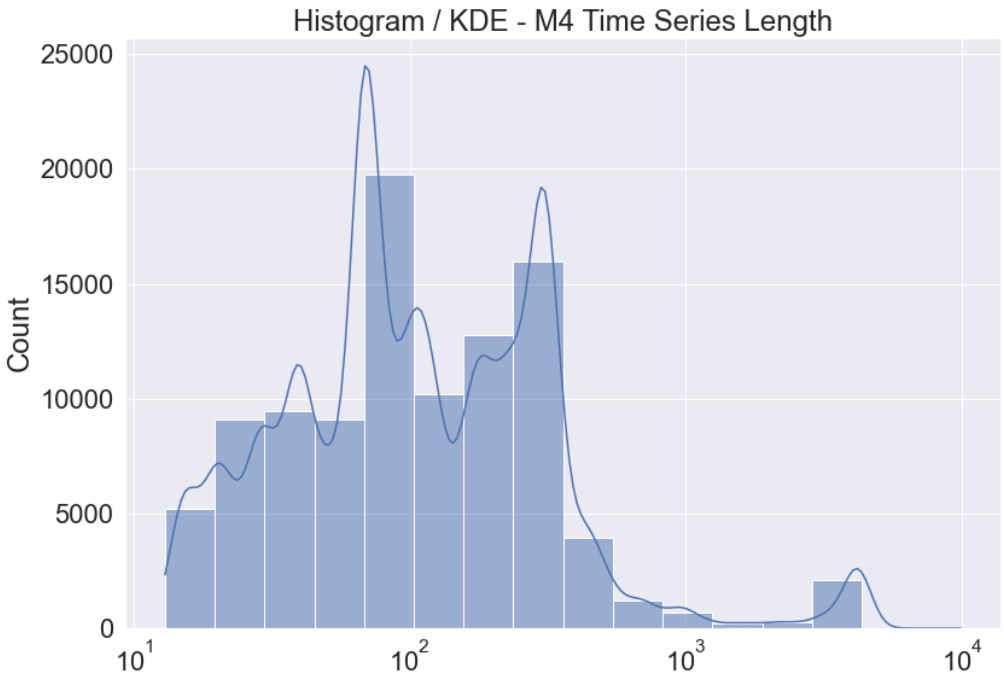

Figure 5: Histogram / KDE - M4 time series length

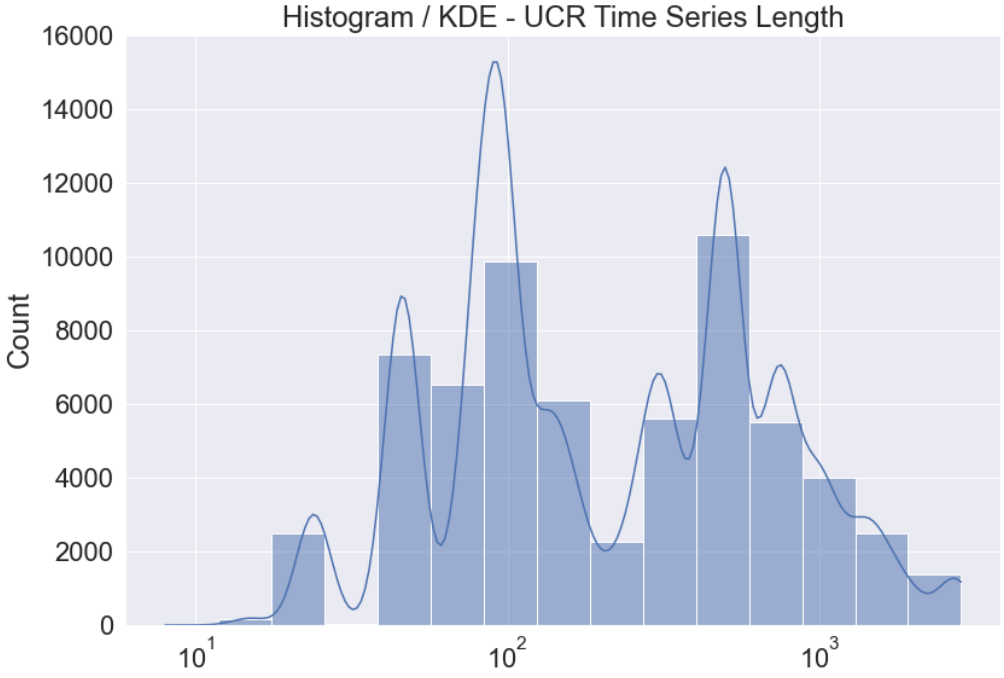

Figure 6: Histogram / KDE - UCR time series length

- computation of deltas for each series to search with statistics and slope of all other time series (review computational complexlity)
- ranking of matching series based highest frequency range match and ONE statistic

6.4 Preprocessing

- M4 data wide format vs. long format

6.5 Parallelization

- computation times
- scalability
- Samples for results only (stratification vs. non-stratification)
- Threads vs. Processes

6.6 Technology (check with Prof. if required)

R vs. Python vs. Mathematica, Matlab

- all languages have FFT either built in or available through common packages

6.7

- load
- transform to FFT vector space
- compare most important frequencies
- compare candidates
- select winner (which criteria)

7 Exploratory Data Study

- what do results look like

8 Formal Evaluation

- (maybe) improvement in forecasting approach
- find dataset with ground truth and compare DTW to this approach
- Distance metrics
- time complexity

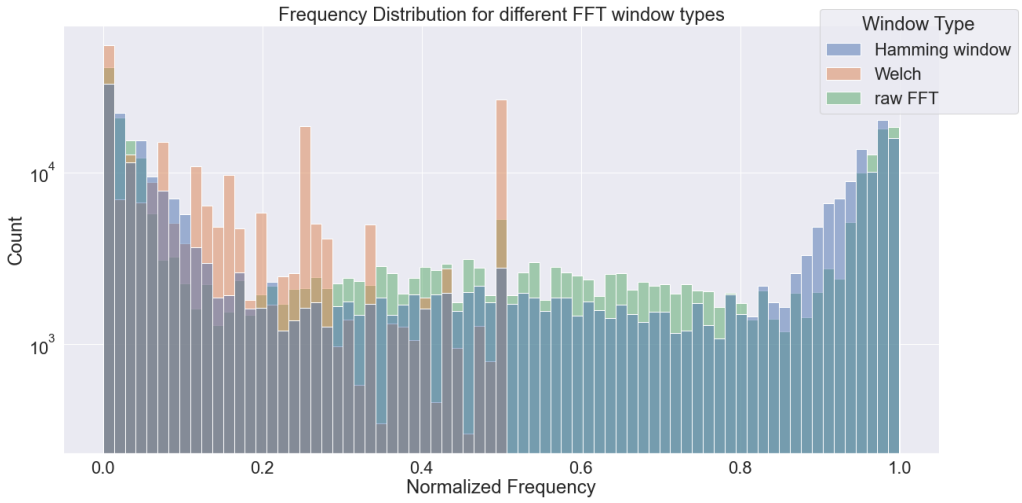

Figure 7: Frequency Distribution for different FFT types

9 Conclusion & future work

9.1 Successes

9.2 Failures

9.3 Flaws

- final computation

9.4 What is missing

- denoising of time series
- adjustment of number of frequencies used
-

10 Results & Discussion

11 References

[1] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2014. ISBN: 9780987507105. URL: https://books.google.de/books?id=gDuRBAAAQBAJ.

[2] Tak-chung Fu. "A review on time series data mining". In: *Engineering Applications of Artificial Intelligence* 24.1 (Feb. 2011), pp. 164–181. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2010.09.007. URL: http://dx.doi.org/10.1016/j.engappai.2010.09.007.

[3] K.J. Åström. "On the choice of sampling rates in parametric identification of time series". In: *Information Sciences* 1.3 (July 1969), pp. 273–278. ISSN: 0020-0255. DOI: 10.1016/s0020-0255(69)80013-7. URL: http://dx.doi.org/10.1016/S0020-0255(69)80013-7.

[4] Yongqiang Tang, Yuan Xie, Xuebing Yang, et al. "Tensor Multi-Elastic Kernel Self-Paced Learning for Time Series Clustering". In: *IEEE Transactions on Knowledge and Data Engineering* (2019), pp. 1–1. ISSN: 2326-3865. DOI: 10.1109/tkde.2019.2937027. URL: http://dx.doi.org/10.1109/TKDE.2019.2937027.

[5] Elon Musk. "An Integrated Brain-Machine Interface Platform With Thousands of Channels". In: *Journal of Medical Internet Research* 21.10 (Oct. 2019), e16194. ISSN: 1438-8871. DOI: 10.2196/16194. URL: http://dx.doi.org/10.2196/16194.

[6] Joshua H Siegle, Aarón Cuevas López, Yogi A Patel, et al. "Open Ephys: an open-source, plugin-based platform for multichannel electrophysiology". In: *Journal of Neural Engineering* 14.4 (June 2017), p. 045003. ISSN: 1741-2552. DOI: 10.1088/1741-2552/aa5eea. URL: http://dx.doi.org/10.1088/1741-2552/aa5eea.

[7] Chuanlei Zhang, Ji'an Luo, Shanwen Zhang, et al. "Introduction to time series search engine systems". In: *2012 International Conference on Systems and Informatics (ICSAI2012)* (May 2012). DOI: 10.1109/icsai.2012.6223532. URL: http://dx.doi.org/10.1109/ICSAI.2012.6223532.

[8] Eamonn J. Keogh and Michael J. Pazzani. "A Simple Dimensionality Reduction Technique for Fast Similarity Search in Large Time Series Databases". In: *Lecture Notes in Computer Science* (2000), pp. 122–133. ISSN: 0302-9743. DOI: 10.1007/3-540-45571-x_14. URL: http://dx.doi.org/10.1007/3-540-45571-X_14.

[9] Q. Kang, C. Yu, Y. Zhang, et al. "Astro-TS3: Time-series Subimage Search Engine for archived astronomical data". In: *Astronomy and Computing* 34 (Jan. 2021), p. 100428. ISSN: 2213-1337. DOI: 10.1016/j.ascom.2020.100428. URL: http://dx.doi.org/10.1016/j.ascom.2020.100428.

[10] Xiaoyue Wang, Abdullah Mueen, Hui Ding, et al. "Experimental comparison of representation methods and distance measures for time series data". In: *Data Mining and Knowledge Discovery* 26.2 (Feb. 2012), pp. 275–309. ISSN: 1573-756X. DOI: 10.1007/s10618-012-0250-5. URL: http://dx.doi.org/10.1007/s10618-012-0250-5.

[11] Zheng Zhang, Ping Tang, and Thomas Corpetti. "Time Adaptive Optimal Transport: A Framework of Time Series Similarity Measure". In: *IEEE Access* 8 (2020), pp. 149764–149774. ISSN: 2169-3536. DOI: 10.1109/access.2020.3016529. URL: http://dx.doi.org/10.1109/ACCESS.2020.3016529.

[12] Shaghayegh Gharghabi, Shima Imani, Anthony Bagnall, et al. "An ultra-fast time series distance measure to allow data mining in more complex real-world deployments". In: *Data Mining and Knowledge Discovery* 34.4 (May 2020), pp. 1104–1135. ISSN: 1573-756X. DOI: 10.1007/s10618-020-00695-8. URL: http://dx.doi.org/10.1007/s10618-020-00695-8.

[13] Donald J. Berndt and James Clifford. "Using Dynamic Time Warping to Find Patterns in Time Series". In: KDD Workshop. 1994, pp. 359–370.

[14] Duo Li, Yifei Zhao, and Yan Li. "Time-Series Representation and Clustering Approaches for Sharing Bike Usage Mining". In: *IEEE Access* 7 (2019), pp. 177856–177863. ISSN: 2169-3536. DOI: 10.1109/access.2019.2958378. URL: http://dx.doi.org/10.1109/ACCESS.2019.2958378.

[15] Jingyue Pang, Datong Liu, Yu Peng, et al. "Intelligent pattern analysis and anomaly detection of satellite telemetry series with improved time series representation". In: *Journal of Intelligent & Fuzzy Systems* 34.6 (June 2018), pp. 3785–3798. ISSN: 10641246, 18758967. DOI: 10.3233/JIFS-169551. URL: https://doi.org/10.3233/JIFS-169551.

[16] S.L. Brunton and J.N. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019. ISBN: 9781108422093. URL: https://books.google.de/books?id=CYaEDwAAQBAJ.

[17] Karl Pearson. "On Lines and Planes of Closest Fit to Systems of Points in Space". In: *Phil. Mag.* 6.2 (1901), pp. 559–572.

[18] J.B.J. Fourier and Firmin père & fils Didot. *Théorie analytique de la chaleur, par M. Fourier.* chez Firmin Didot, pere et fils, 1822.

[19] Jessica Lin, Eamonn Keogh, Stefano Lonardi, et al. "A symbolic representation of time series, with implications for streaming algorithms". In: *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery - DMKD '03* (2003). DOI: 10.1145/882082.882086. URL: http://dx.doi.org/10.1145/882082.882086.

[20] Jessica Lin, Eamonn Keogh, Li Wei, et al. "Experiencing SAX: a novel symbolic representation of time series". In: *Data Mining and Knowledge Discovery* 15.2 (Apr. 2007), pp. 107–144. ISSN: 1573-756X. DOI: 10.1007/s10618-007-0064-z. URL: http://dx.doi.org/10.1007/s10618-007-0064-z.

[21] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. "Fast subsequence matching in time-series databases". In: *Proceedings of the 1994 ACM SIGMOD international conference on Management of data - SIGMOD '94* (1994). DOI: 10.1145/191839.191925. URL: http://dx.doi.org/10.1145/191839.191925.

[22] Chotirat Ann Ratanamahatana and Eamonn Keogh. "Making Time-series Classification More Accurate Using Learned Constraints". In: *Proceedings of the 2004 SIAM International Conference on Data Mining* (Apr. 2004). DOI: 10.1137/1.9781611972740.2. URL: http://dx.doi.org/10.1137/1.9781611972740.2.

[23] J. Ratcliffe. *Foundations of Hyperbolic Manifolds.* Graduate Texts in Mathematics. Springer New York, 2006. ISBN: 9780387473222.

[24] Howard Anton. *Calculus - A New Horizon.* Calculus v. 6. Wiley, 1999, pp. 403–404. ISBN: 0-471-15306-0.

[25] W.H. Press, S.A. Teukolsky, W.T. Vetterling, et al. *Numerical Recipes in FORTRAN 77 Macintosh Diskette Version 2.0: The Art of Scientific Computing.* Fortran numerical recipes. Cambridge University Press, 1992, pp. 542–545. ISBN: 9780521437219.