Tiago Filipe Teixeira dos Santos MSc

Early Classification of Time Series and Deep Learning

MASTER'S THESIS
to achieve the university degree of
Diplom-Ingenieur
Master's degree programme: Computer Science

submitted to
Graz University of Technology

Supervisor
Dipl.-Ing. Dr.techn. Roman Kern

Know-Center
Institut für Wissenstechnologien

Graz, July 2016

This document is set in Palatino, compiled with pdfLaTeX2e and Biber.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, _____ _____

Date Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____ _____

Datum Unterschrift

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Abstract

This thesis aims to shed light on the early classification of time series problem, by deriving the trade-off between classification accuracy and time series length for a number of different time series types and classification algorithms. Previous research on early classification of time series focused on keeping classification accuracy of reduced time series roughly at the level of the complete ones. Furthermore, that research work does not employ cutting-edge approaches like Deep Learning. This work fills that research gap by computing trade-off curves on classification ”earlyness” vs. accuracy and by empirically comparing algorithm performance in that context, with a focus on the comparison of Deep Learning with classical approaches.

Such early classification trade-off curves are calculated for univariate and multivariate time series and the following algorithms: 1-Nearest Neighbor search with both the Euclidean and Frobenius distance, 1-Nearest Neighbor search with forecasts from ARIMA and linear models, and Deep Learning.

The results obtained indicate that early classification is feasible in all types of time series considered. The derived tradeoff curves all share the common trait of slowly decreasing at first, and featuring sharp drops as time series lengths become exceedingly short. Results showed Deep Learning models were able to maintain higher classification accuracies for larger time series length reductions than other algorithms. However, their long run-times, coupled with complexity in parameter configuration, implies that faster, albeit less accurate, baseline algorithms like 1-Nearest Neighbor search may still be a sensible choice on a case-by-case basis.

This thesis draws its motivation from areas like predictive maintenance, where the early classification of multivariate time series data may boost performance of early warning systems, for example in manufacturing processes.

Contents

Abstract v

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Research Questions . . . 3
- 1.3 Outline . . . 3

2 Literature Review 5
- 2.1 Time Series Definitions and Types . . . 5
- 2.2 Time Series Classification . . . 6
- 2.3 Deep Learning for Time Series . . . 9
- 2.4 Early Time Series Classification . . . 12
- 2.5 Summary . . . 15

3 Methods 17
- 3.1 Time Series Concepts and Theory . . . 17
  - 3.1.1 General Definitions . . . 17
  - 3.1.2 Time Series Classification Definitions . . . 21
  - 3.1.3 Classification Algorithms - 1-Nearest Neighbor . . . 21
  - 3.1.4 Classification Algorithms - Deep Learning . . . 23
  - 3.1.5 Early Classification . . . 28
- 3.2 Implementation . . . 31
  - 3.2.1 Algorithm implementation details . . . 32
  - 3.2.2 Frameworks . . . 36
- 3.3 Summary . . . 37

4 Results 39
- 4.1 Datasets . . . 39
  - 4.1.1 UCR . . . 40

4.1.2 Auslan 41
4.1.3 Snackbox 42
4.2 Results per Dataset 44
4.2.1 Univariate Time Series 44
4.2.2 Multivariate Time Series 48
4.3 Summary 55

5 Discussion **57**
5.1 Observations per Dataset 57
5.1.1 Univariate Time Series 57
5.1.2 Multivariate Time Series 61
5.2 Limitations 66
5.3 Summary 67

6 Conclusion **69**
6.1 Review 69
6.2 Future Work 70
6.3 Summary 71

Bibliography **75**

List of Figures

3.1 Diagram of a neuron and its components 23
3.2 Diagram of a neural network 24

4.1 Example of a sign in the Australian sign language (Auslan) . 41
4.2 Screenshot of the snackbox and its sensors 42
4.3 Early Classification Tradeoff Curve - UCR 45
4.4 Early Classification Tradeoff Curve - Auslan 49
4.5 Early Classification Tradeoff Curve - Snackbox 52

1 Introduction

This chapter's purpose is to introduce the reader to this thesis' main topic, early classification of time series. It starts by naming reasons that make this problem an interesting choice for a thesis, in the motivation section. Then, the research questions guiding this work are presented, finally followed by the presentation of the structure of this thesis in the section outline.

1.1 Motivation

Time series are general mathematical constructs, which describe a sequence of data indexed and ordered by time. This general definition, while being not too abstract to grasp, already gives a sense of the vast number of applications such a concept might have in practice. These range from the analysis of financial markets, perhaps modeled as the development of some stock indices over time, forecasting the weather, a setting where a large number of time series of several indicators like temperature, humidity or atmospheric pressure need to be somehow combined, understanding music or sound, where the search for fundamental frequencies in a temporal series of pitches might be of interest, to even computing a rocket's trajectory as it soars through the skies. In short, the use of time series, as a tool to understand and model a very large variety of processes evolving over time, is ubiquitous.

As a concrete motivating example for the more specific task at this thesis' hand, take, for instance, a manufacturing process. In modern manufacturing processes, machines are put to use to carry out many of the assembly operations humans used to perform manually decades (or centuries) ago. Picture therefore a manufacturing process with a high degree of automation,

such that humans are no longer required to be active in the factory floor, but they instead monitor the progress of the machines in the assembly line and ensure that it goes as intended and conforms to certain quality standards. How do factory workers monitor the progress of the machines and are alerted to any issue in the process? The answer is machine sensors, spread throughout the factory's assembly line, which collect data on the machines and on each manufacturing step and report that data. Thus, it is the factory workers' job not only to monitor this data, but to react to it in case some manufacturing step needs immediate attention, for instance in case of an unexpected problem. In such a scenario, where an unexpected problem might occur, it is expected that the maintainer has a timely reaction to sensory data reflecting that issue. A timely, or maybe as-soon-as-possible, reaction in such cases may for example save a production batch, that would otherwise have to be scrapped, or perhaps even economize the often costly maintenance expenses for a given machine, whose sensory data might show that some component in it is malfunctioning and the sooner that flaw gets eradicated the better. A real life example of the benefits of early detection of such events or outliers in sensor data would be that of satellite or probe trajectory corrections. Space missions like the Mars Climate Orbiter by NASA and the Jet Propulsion Laboratory have gone awry in the past, not least due to a failure by the mission monitoring to heed early warning systems (Laboratory, (2010)). Such failures entail not only very heavy financial losses, but also cost years or perhaps decades of work.

The data provided by the sensors in the examples above may be modeled as a time series, or perhaps a (large) number of them. Then, to detect events, such as a malfunctioning component in the time series data, as soon as possible, or to provide early warnings from the sensory data, one wishes to be able to tell if said event is happening in the time series data as early as possible, which is equivalent to just using a certain window of the time series, with as little observations as possible. However, in the same early warning setting, one also wishes to detect that irregularity as accurately as possible. These are in general conflicting objectives, since, intuitively, more observations lead to a better informed, i.e. more accurate, decision basis in a warning system, but, naturally, having more observations of the time series means waiting longer to make a possibly time-critical decision. The question then becomes, what is the trade-off between both aspects?

This thesis' main topic concerns exactly that: to apply machine learning methods to help make early decisions in such cases, for data modeled as time series. In the machine learning context, this is often discussed as time series classification, more specifically early time series classification. The existing literature on this proposes a series of approaches for attacking this problem, but they do not empirically explore the tradeoff between time series length and classification accuracy to derive the so-called early classification tradeoff curve, and they also do not comparatively analyze classic time series classification approaches with more modern techniques like Deep Learning models for that.

Thus, it is the aim of this thesis to tackle this research gap by addressing the following research questions.

1.2 Research Questions

As previously described, it is the objective of this thesis to investigate early classification of time series, and in particular the tradeoff in classification accuracy vs. "earlyness" one incurs when classifying time series before a full sequence is observed. Specifically, this thesis contributes to research in this area by setting out to answer the following research questions:

1. *Which types of series allow for early time series classification?*
2. *What is the trade-off curve between classification accuracy and time series length?*
3. *Which classification algorithms are more appropriate for the early time series classification task? In particular, how do Deep Learning methods perform in this matter and how do they compare with other classification algorithms?*

1.3 Outline

In the next chapter, numbered 2, the current state of the art is presented and the research gap within it, which this thesis aims to bridge, is identified.

Then, in chapter 3, the methods employed by this thesis to tackle the research questions are formally introduced. The chapter starts with a mathematical formalization of the problem setting surrounding time series, time series classification, early time series classification and algorithms therefore. Thereafter, implementation details are provided for the realization of that theory.

The results chapter (4) then ensues, beginning with an exposition of the three different time series datasets examined by this thesis. Graphical and numerical reports on the results achieved with the proposed early time series classification approaches and those datasets round up this chapter.

Chapter 5 is the evaluation chapter, in which interpretations are proposed and conclusions are drawn from those obtained results and the research questions can begin to be answered.

Finally, chapter 6 concludes this thesis with a summary reflection on the progress it achieved and a discussion of promising avenues of future research work in the area of early classification of time series.

2 Literature Review

This literature review is structured as follows. A brief overview on definitions of time series and time series types to be considered by this thesis serves as an introduction to the literature on this topic. Then, literature on general algorithms for time series classification is presented. The application of deep learning algorithms for time series classification is awarded a separate section. Finally, early time series classification literature is reviewed.

2.1 Time Series Definitions and Types

Brockwell and Davis, (2013) define a time series as a series of observations x_t, with each observation x corresponding to a specific time t. They distinguish between *continuous* and *discrete* time series. The latter refers, according to the authors, to time series with a discrete set of observations, and the former to time series of continuously recorded observations over some time interval like $T_0 = [0, 1]$. This thesis focuses on discrete time series.

Another distinguishing feature of time series regards the observations x_t. x_t may be defined as a singular value, in which case the time series is called a "univariate time series". If x_t represents an n-dimensional vector, then the time series is termed "multivariate time series". Both single variable and multivariate time series will be analyzed. The observation values are assumed to be defined in $\mathbb{R}$ and respectively in $\mathbb{R}^n$.

Time series can be seen in a large number of different contexts, such as economic forecasting, stock market analysis or inventory analysis, to name a few of the applications mentioned for example by NIST/SEMATECH, (2013). Many other examples and applications of time series theory will be covered in more detail later.

Time series repositories contain samples of such time series for analysis purposes. These provide the data sources for the practical part of this work. The UCR Time Series Classification Archive by Chen et al., (2015) includes not only a set of labeled univariate time series for classification, but also benchmarks for classifiers. Many papers addressing time series classification used this time series repository and its benchmarks as a data source and as a baseline, as will be seen later. One of the datasets cleaned and summarized in the collection by Chen et al., (2015) is the Auslan dataset, which was originally prepared by M. W. Kadous, (2002). This data set of multivariate time series consists of samples of Auslan (Australian Sign Language) signs, and it is also referenced by a number of time series classification papers. The third and last time series data set analyzed here is also composed of multivariate time series. This dataset, named Snackbox, shows data captured by sensors installed at a room with a box with snacks. The Snackbox dataset represents sensory data captured in a realistic context, where data measurements arrive in real-time. These measurements have been annotated to allow for time series classification on different levels.

2.2 Time Series Classification

Time series classification refers to the process of assigning a label, or class, to a time series. There is a *very* large number of approaches to address the problem of time series classification. Thus, the following review of the literature on time series classification does not aim to be exhaustive, but to give an overview of common as well as cutting edge approaches and theory.

The definition given by Skiena, (1998) of the Nearest Neighbor Search problem in the field of computational geometry is applied here to the machine learning context: In a supervised classification setting, which assumes a train set of labeled samples, the 1-Nearest Neighbor algorithm takes a new, unlabeled instance and assigns to it the label of the nearest neighbor from that train set, according to a certain measure of distance in $\mathbb{R}$. The 1-Nearest Neighbor classifier is used by Chen et al., (2015), Yang and Shahabi, (2007)

and Wei and E. Keogh, (2006), among many others, to classify both univariate as well as multivariate time series. To name a few of the distance measures used with the 1-Nearest Neighbor algorithm, these papers respectively use the Euclidean distance and dynamic time warping for univariate time series, as well as the Frobenius distance for multivariate time series.

Chen et al., (2015) uses the 1-Nearest Neighbor algorithm with the Euclidean distance as a baseline for comparison with dynamic time warping with different warping windows, since the latter generally outperforms the Euclidean distance as concluded by the authors.

On the other hand, Yang and Shahabi, (2007) optimize the k-Nearest Neighbors search (incl. 1-Nearest Neighbor) for multivariate time series with an extension of the Frobenius distance termed "Eros". "Eros" applies the Frobenius distance to the singular value decomposition of the covariance matrices of 2 different multivariate time series in matrix-form. This helps with getting both matrices to have equal dimensions to compute the Frobenius distance, as well to capture the importance of the covariances of columns of the multivariate time series matrices, i.e. of the variables within the time series.

Yang and Shahabi, (2007) and Chen et al., (2015) are not the only papers making use of Nearest Neighbor search as an integral part of a classification process. The paper Wei and E. Keogh, (2006) aims to improve the 1-Nearest Neighbor's performance in a binary classification setting where few labeled data is available. The approach proposed by the authors involves reinforcing the 1-Nearest Neighbor classifier's performance on the unlabeled set by iteratively adding instances classified with high confidence to the train set, until some stopping criterion is achieved. The reason for using 1-Nearest Neighbor as the underlying algorithm, according to Wei and E. Keogh, (2006), is that it is hard to beat its accuracy, its performance converges to that of more complex classifiers for longer series, it is simple to implement, it does not require configuration in the form of parameters and it generalizes to $\mathbb{R}^n$, for example with the Frobenius distance.

Many other approaches have been proposed to improve the strong performance and benefits of the 1-Nearest Neighbor algorithm outlined above.

Take for example Mohammed Waleed Kadous, (1999). That paper describes an architecture that, given raw data as multivariate time series, extracts events, clusters them and combines them back again with globally computed features on the raw data to produce train data for learning classification rules. Its implementation, called "TClass", uses k-means clustering and the naive Bayes learner for the steps previously described. According to the author, this approach leads not only to improved classification accuracy, but also higher understandability of the features used for classification.

Other comprehensible features used in time series classification are so-called "literals": simple statistics, like averages, computed over intervals of the time series. Both Juan J Rodriguez, J. J. R. Guez, and Carlos J Alonso, (2002) as well as Juan Jose Rodriguez, Carlos J Alonso, and Maestro, (2005) apply AdaBoost (Freund, Schapire, and Abe, (1999)) on those literals. AdaBoost is an algorithm which linearly combines many such simple classifiers to one better performing classifier. The paper Juan J Rodriguez, J. J. R. Guez, and Carlos J Alonso, (2002) uses adaboosted literals to cope with variable length time series and perform early classification, as will be seen in more detail later in this literature review. In Juan Jose Rodriguez, Carlos J Alonso, and Maestro, (2005), the same authors aim to improve previous results by considering more complex literal combinations to generate new features. They then apply, on that set of new features, support vector machines, both linear as well as with the Gaussian kernel, to achieve performance gains over the results outlined by other papers using 1- and k-Nearest Neighbors, on the previously mentioned Auslan data set, among others.

Another approach regarding feature extraction concerns shapelets. Shapelets are sub-sequences of the time series that allow for classification basing on local, phase-independent similarity in shape (Hills et al., (2014)). They aim to maximally represent the class of a time series. Hills et al., (2014), Lines et al., (2012) and Ye and E. Keogh, (2009) use shapelets to derive easy-to-interpret features, while also experimentally improving accuracy of the 1-Nearest Neighbor algorithm with the dynamic time warping distance (in some of the datasets of Chen et al., (2015)).

Both the shapelet as well as the time series forest approaches are designed for univariate time series. One further approach that considers multivariate time series, by Banko and Abonyi, (2012), combines the dynamic time

warping distance with principal component analysis to create the ”CBDTW” (correlation based dynamic time warping) measure. That similarity measure is computed by first segmenting an unclassified time series using principal component analysis, mapping the segments to the real numbers using a cost function and then applying the dynamic time warping distance to compare the unclassified time series to the train set of previously segmented time series. This approach thus leverages correlation effects to accurately describe time series classes.

All of the papers mentioned before use either the datasets of Chen et al., (2015) or the multivariate time series data set of M. W. Kadous, (2002) or both. This makes benchmarks and comparisons between them feasible, effectively addressing concerns mentioned by E. Keogh and Kasetty, (2003).

The importance of hidden correlations in time series data, as highlighted by the paper mentioned last, is one of the factors motivating the application of deep learning for time series classification.

2.3 Deep Learning for Time Series

Recently, deep learning has also been applied to the time series classification problem. This section starts with a review of literature on deep learning in general and then addresses state-of-the-art deep learning approaches for time series classification.

To cite Schmidhuber, (2015), quote, ”A standard neural network (NN) consists of many simple, connected processors called neurons, each producing a sequence of real-valued activations”. There are input neurons, which get activated from the environment, and other neurons, for example hidden or output neurons, which, according to the same author, quote, ”get activated through weighted connections from previously active neurons”. The assignment of weights for the connections controls the output of the neural network. In this context, the process of tuning weights to attain certain output is termed learning. The neurons are typically grouped into layers called input, output or hidden (i.e. those between input and output) layers. Each layer transforms, often non-linearly, the aggregate activation of the

previous layer and propagates that to further layers. Deep learning consists of assigning weights (in the context described above) across multiple such layers of often non-linear transformations.

As a part of artificial intelligence, deep learning techniques are currently experiencing both numerous practical applications as well as various research developments. Bengio, (2009) and Deng and Yu, (2014) outline, in their reviews of deep learning methods and architectures, many of the types of deep learning models and their purposes. A few examples of deep neural network architectures used in supervised learning settings include multi-layer neural networks as explained above and convolutional neural networks.

The latter consists of a deep neural network, especially designed for computer vision tasks. As described by LeCun et al., (1998), convolutional neural networks have unique properties like "sparse connectivity", which means that each layer is associated with just one region of an input image, i.e. the so-called "receptive fields", and "shared weights", which refers to each layer having the same set of weights (but with different receptive fields). There are also several deep neural network architectures designed to work in an unsupervised context, such as Restricted Boltzmann Machines, a type of stochastic artificial neural network, Deep Belief Networks, which consist of multiple learning layers (like Restricted Boltzmann Machines) which are trained greedily per layer, or Autoencoders, a type of feed forward network designed to replicate its input.

Deep learning architectures like the ones described above have been applied to successfully address problems like speech recognition (G. Hinton et al., (2012)), image classification (Taigman et al., (2014)) or even beating professional Go players (Silver et al., (2016)). Besides the very promising practical results achieved in those areas, the author Bengio, (2009) mentions the ability deep architectures have to succinctly represent functions as opposed to very large shallow architectures, as the main theoretical advantage of deep learning. Furthermore, the broad availability of open-source software frameworks for deep learning eases the deployment of distributed, performant, complex and state-of-the-art deep neural network models. Notable examples thereof are Theano (Bergstra et al., (2010)), Torch (Ronan Collobert et al., (2016)),

Google's TensorFlow (Martin Abadi et al., (2015)) and Microsoft's CNTK (Amit Agarwal et al., (2014)).

Yet another prominent application of deep learning architectures lies in time series classification. The literature on this topic is reviewed next. One of the first references on multivariate time series classification with neural networks is Chakraborty et al., (1992). That paper proposes a feedforward neural network for predicting flour prices for a number of geographical locations. That neural network's predictions outperform those by an autoregressive moving average model on the root mean squared error measure. However, this paper does not employ deep neural nets since it makes use of only one input layer, one hidden layer and one output layer.

The work by Ahmed et al., (2010) also suggests that neural networks may bring performance improvements to time series forecasting. In their empirical study, the authors directly compare machine learning methods like support vector regression, k-Nearest Neighbor regression and multilayer perceptron to assess their performance as time series forecasting algorithms. The authors conclude that the multilayer perceptron is among the more accurate methods for that task. A multilayer perceptron is a simple neural network consisting of just one hidden layer (and, of course, input and output layers), so, like before, this paper also does not address the use of deep neural networks.

In Busseti, Osband, and Wong, (2012) however, several deep neural networks are trained and analyzed for the task of energy demand load forecasting. A deep recurrent neural network of 2 hidden layers and delivered the best performance in terms of root mean squared error of predicted values. The authors also stress the importance of feature selection and engineering to fully tap neural networks' power to fit highly non-linear models. To that end, both domain knowledge as well as, among other transformations, principal component analysis proved useful in achieving the best possible results.

Feature selection and engineering as part of the application of deep neural networks to time series data is also one of the topics of Langkvist, Karlsson, and Loutfi, (2014). That work addresses the use of deep neural networks to derive, from raw data, relevant features for time series modeling in an unsupervised setting.

Finally, Batres-Estrada, (2015) proposes a complex model, consisting of a deep belief network coupled to a multilayer perceptron, to compose portfolios of stocks. In that work, the deep neural network's input are carefully selected stock value time indexes, which reflects the role domain knowledge plays in the architecture of complex deep neural networks. The deep neural network delivered promising results and performed better in comparison with, among others, a logistic regression network.

After dealing with the literature on time series classification, both with general as well as with deep learning algorithms, work on early time series classification will be reviewed next.

2.4 Early Time Series Classification

Xing, Pei, and Philip, (2009), Parrish et al., (2013) and Dachraoui, Bondu, and Cornuejols, (2015) all agree on the basic early classification definition: It is the problem of trying to come to a classification decision with as little observations of a time series as possible, while sacrificing classification accuracy as little as possible.

A different interpretation of the early classification problem is provided by Petitjean et al., (2014), which tackles the computational performance side of time series classification. This paper improves both the time and resource complexity of 1-Nearest Neighbor with the dynamic time warping distance by creating nearest "centroid" classifiers that are both faster and at least as accurate as nearest neighbor algorithms. This interpretation of early time series classification is not, however, the focus of this thesis.

Again, all of the papers mentioned below use either the datasets of Chen et al., (2015) or the multivariate time series data set of M. W. Kadous, (2002) or both, which is once again helpful for benchmarking and comparing the approaches (E. Keogh and Kasetty, (2003)).

One of the first works on the topic of early classification, as defined over time series length, was written by Juan J Rodriguez, J. J. R. Guez, and Carlos J Alonso, (2002). The authors start with literals, which are, as mentioned before, simple indicators or statistics, for example if a time series is going

up or down over a previously defined interval. These base literals are then combined with an AdaBoost.MH, a version of the prominent ensemble classifier created by Freund, Schapire, and Abe, (1999), which can cope with multivariate time series with variable length. Since their approach can work with time series of variable length, if a partial time series is used as input to their ensemble classifier, some of the literals, though not all, will still be able to output a classification response. Using the fact that the ensemble classifier is just a linear combination of those literals, the authors omit literals with unknown values to still reach a classification decision on a partial sample of the time series, thus performing early classification. Their experimental evaluation of classification performance on an early version of the dataset by Chen et al., (2015) reveals that early classification may be a promising future research avenue, for example by tuning the AdaBoost learning process to address early classification.

After Juan J Rodriguez, J. J. R. Guez, and Carlos J Alonso, (2002), many other authors addressed and further formalized the early classification problem. With Xing, Pei, Dong, et al., (2008), Xing, Pei, and Philip, (2009) and Xing, Pei, Philip, and Wang, (2011), that group of authors advanced research on early classification of time series from a number of different perspectives.

The paper Xing, Pei, Dong, et al., (2008) presents a first approach dedicated to early classification basing on both sequential rules mining and sequential decision trees. The focus of this paper is, however, sequences, which are time series taking values from a finite set (like an alphabet). A more general context, which would be, like in this thesis, time series taking values in $\mathbb{R}$, is considered in the following papers.

Further, more prominent early classification approaches by mostly the same authors include "Early Time Series Classification", developed as early as Xing, Pei, and Philip, (2009) and described in further detail in Xing, Pei, and Philip, (2012). The author stipulates desirable, additional characteristics for early classifiers in general. These are the "seriality" of early classifiers, i.e. the invariance of the classification decision once a certain length of the time series has been observed, and that its accuracy when classifying the reduced time series is retained (or at least does not drop too much) with respect to classification on the time series with full length. These properties guide the authors' derivation of the Early Time Series Classification framework.

That framework builds upon the 1-Nearest Neighbor algorithm and introduces concepts like minimum prediction length and a clustering algorithm. Those concepts together enable smart grouping of time series in a train set according to common prefixes between those time series for early, serial and reliable classification. That framework outperforms not only weaker early classifiers such as 1-Nearest Neighbor fixed, which refers to the application of the 1-Nearest Neighbor on a time series with a fixed reduced length, but also even the 1-Nearest Neighbor algorithm with the full length time series, on some cases.

The problem of finding appropriate features for early time series classification is the main topic of Xing, Pei, Philip, and Wang, (2011), the last paper by those authors on early classification.

More recently, Parrish et al., (2013) proposed a probabilistic framework using quadratic discriminants and support vector machines for performing early classification. That paper's key concept is the reliability of the classification decision, i.e. the degree of confidence with which one can say that the current incomplete data is sufficient to come to the same classification as the complete data (with high probability). That framework allows for classification as soon as a reliability threshold is met. The results achieved by that framework compare favorably against the previously mentioned 1-Nearest Neighbor fixed and Early Classification of Time Series methods, both on classification accuracy and earliness.

The latest development in the theoretical treatment of early time series classification seems to be the paper by Mori et al., (2016). The authors employ a method called "Early classification framework for time series based on class discriminativeness and reliability of predictions", in short "ECDIRE", to classify bird songs early and also beat the results achieved by Xing, Pei, and Philip, (2009) and Parrish et al., (2013) on the same data sets used by those papers. "ECDIRE" is a probabilistic classification framework that is organized into 4 steps. The first step, termed by the authors "Analysis of the discriminativeness of the classes", aims to derive a set of time series timestamps which enable good discrimination of time series classes early. This step begins with the definition of a set of early time series timestamps as a percentage of total time series length. Then, a variation of a Gaussian Process classifier, which in turn is a Bayesian

probabilistic classifier, is trained with cross-folding on each of the time series reduced by the previously defined percentages. Early timestamps are thus identified as those which still maintain an accuracy above the reduced time series length (in percentage) times the accuracy (also in percentage) attained with the same classifier on the full length time series. In a second step, thresholds for classification reliability are defined as a, quote, "distance in terms of differences in class probabilities of the winning class and the next most probable class" per time series and class. This improves reliability by avoiding uncertain predictions and thus also classification timestamps, which might be too early to distinguish certain classes from each other. Finally, the actual probabilistic classifiers are trained on the full training set with the early timestamps computed in step 1 (as well as with the full time series length, as a fallback solution if no early timestamp was found in some particular case).

The paper Mori et al., (2016) was not the only one to apply early classification techniques to address a real-world data sets and problems. Examples thereof include, but are not limited to, the following papers. Ghalwash, Ramljak, and Obradovic, (2012) compose a hidden Markov model with a support vector machine to create an early classifier, which performed well on a medical domain dataset containing gene expression values for a number of multiple sclerosis patients, for which not much data was available. The work by Hatami and Chira, (2013) introduces a classifier with a reject option, which allows the classifier to abstain from a classification decision if it is not clear-cut, weighing in that with the cost of making further observations. Their classifier with the reject option is general enough to allow for different classification algorithms, like a support vector machine or k-Nearest Neighbors, to be used. This early classifier was used in the classification of odors, serving as the basis for a performant electronic nose.

2.5 Summary

In this chapter, literature on time series definitions, its classification, in particular with deep neural networks, and early classification techniques

were reviewed. In conclusion, there is a very large number of both theoretical as well as practical work to classify time series, and, in particular, classify time series early. In particular, this body of work presents not only theoretical results and frameworks to use for different types of time series and data sets, but it also provides results to benchmark against.

However, to the best of the knowledge of the author of this thesis, research on the application of deep learning for early time series classification seems to be missing. The next sections will present how this work thus makes use of many of the theoretical constructs presented above to fill that research gap.

3 Methods

The chapter "Methods" will cover all three of the research questions guiding this thesis starting from a theoretical point of view and following it up from a practical one.

The section 3.1.1 explains which types of series are going to be used for early time series classification. Research question number 2, which asks for the derivation of a trade-off curve between time series length and classification accuracy, is formally addressed in 3.1.5 and from a practical standpoint in 3.2.1 too. The third research topic regards a comparison of classification algorithms for the problem of early classification. This is the subject of the sections on classification algorithms, named 3.1.2, 3.1.3 and 3.1.4 in the theory part, and 3.2.1 in the implementation part. The Methods chapter is rounded up with a brief overview of the software technology stacks employed by this thesis.

3.1 Time Series Concepts and Theory

Firstly, formal definitions relevant for both time series and time series classification are introduced.

3.1.1 General Definitions

Time Series Basics

A time series is an ordered sequence of pairs $(t, x(t))$, where t represents a timestamp and $x(t)$ represents the value the time series takes at timestamp

t. This sequence is ordered by the timestamp t ascendingly. Typically, if a time series has length L, then it has L such ordered pairs. Mathematically:

Definition 1 (Time Series). *A univariate time series ts can be defined as:*

$$ts = \{(t, x(t)) : t \in \mathbb{N}, x(t) \in \mathbb{R}\}, \text{ with } x : \mathbb{N} \to \mathbb{R}$$

Multivariate time series ts of dimension n can be defined as

$$ts = \{(t, x(t)) : t \in \mathbb{N}, x(t) \in \mathbb{R}^n\}, \text{ with } x : \mathbb{N} \to \mathbb{R}^n$$

A time series ts has length $L_{ts} \iff |ts| = L_{ts}$.

The shorter notation $ts[1, L_{ts}]$ *is also used to denote the values the time series ts takes in timestamps* 1 *through* L_{ts}.

Thus, this thesis discusses only discrete time series, i.e. those with a countable set of observations.

The theoretical elaboration above is, in general, sufficient for the problem of time series classification. However, in the context of time series modeling and especially forecasting, the property "stationarity" of a time series is relevant. Below, the criteria for weak stationarity of time series are given, up to small changes, as formulated by Shumway and Stoffer, (2013). Note that, like those authors, this thesis henceforth refers to "weakly stationary" time series simply as "stationary" time series.

Definition 2 (Weakly Stationary Time Series). *Consider a probabilistic context, where a time series is described by the marginal probability distribution* F_t *of a collection of random variables indexed by time* $t \in 1, \ldots, n$. *Let* f_t *represent the derivative of* F_t, *i.e. the corresponding marginal density function. In such a context, a weakly stationary time series* x_t *is one that fulfills the following 2 conditions:*

- x_t*'s mean value function,* $\mu_t = E(x_t) = \int_{-\infty}^{+\infty} x f_t(x)\, dx$ *(provided it exists), is constant and does not depend on time t*
- x_t*'s autocovariance function,* $\gamma(s,t) = cov(x_s, x_t) = E[(x_s - \mu_s)(x_t - \mu_t)]$, *depends on s and t only through their difference* $|s - t|$.

This property will be useful in the study of time series models, which address the problem of time series forecasting.

Time Series Models and Forecasting

This section presents two commonly used models for predicting time series behavior. The reason for introducing here them concerns early classification: Whenever reduced time series need to be classified, one idea is to predict the time series' missing tail and to use that prediction, plus the reduced time series, as a comparison basis with a full-length train set. A more elaborated discussion of this idea will be presented later on.

The first class of time series models presented here is "ARIMA", which is short for "Autoregressive Integrated Moving Average" and comprises stationary time series. The following definition and the accompanying explanations can be found in Shumway and Stoffer, (2013), R. J. Hyndman and Athanasopoulos, (2014) and Nau, (2016a).

Definition 3 (Autoregressive Integrated Moving Average Models ARIMA(p, d, q)). *ARIMA(p, d, q)-type models for a general time series x_t are represented by the following general equation:*

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \ldots + \alpha_p x_{t-p} + \epsilon_t + \beta_1 \epsilon_{t-1} + \ldots + \beta_q \epsilon_{t-q},$$
$$\text{with } \alpha_i, \beta_i \in \mathcal{R}$$

These models are thus are auto-regressive. This means that the dependent variable x_t is modeled, on the one hand, as a function of p of its lags.

On the other hand, for the modeling of x_t, the model also considers q lags of its forecast errors ϵ_i. This part of the model is the so-called "moving average" part.

Finally, the d parameter of this time series model references the order of differencing applied. The process of differencing is the discrete analog of derivation. It is, for example for $d = 1$, defined as $x_t = x_t - x_{t-1}$. Differencing up to a certain order is commonly done to make a time series x_t stationary and to then apply the ARIMA model to the differenced time series.

One common extension of the ARIMA(p, d, q) model is the seasonal ARIMA model, denoted as ARIMA(p, d, q)(P, D, Q). The (P, D, Q) tuple of parameters refers to the parameters (p, d, q) as defined above, but for the seasonal component of a time series.

In general, there are a number of rules-of-thumb and guidelines to help choose parameters, configure and validate ARIMA models. This thesis will not discuss these in detail and instead point to the references mentioned prior to this definition, as well as to Nau, (2016b).

Another popular and efficient theoretical construct for building time series models are linear regressions. Again, the work by Shumway and Stoffer, (2013) and R. J. Hyndman and Athanasopoulos, (2014) is quoted below:

Definition 4 (Linear Regression). *Linear regression models for a general time series x_t have the following equation:*

$$x_t = \beta_1 z_{t1} + \beta_2 z_{t2} + \ldots + \beta_q z_{tq} + w_t$$

In the previous equation, the dependent variable x_t is regarded as a linear combination of q predictor, independent, time series. The regression coefficients β_i are unknown and $\{w_t\}$ is a random noise process.

Given n samples of the z_t variables and using matrix notation, where $\boldsymbol{\beta'} = (\beta_1, \ldots, \beta_q)^T$ is a transposed column vector of the regression coefficients and $\boldsymbol{z_t} = (z_{t1}, \ldots, z_{tq})$ the analogous for the independent variables, the estimators for the $beta_i$ can be obtained via least squares minimization, i.e. as a solution to the following equation:

$$\min \sum_{t=1}^{n} w_t^2 = \min \sum_{t=1}^{n} (x_t - \boldsymbol{\beta' z_t})^2$$

This general equation allows one to model linear trends and seasonality effects, among others.

Having established basic time series theory and these two time series models, the next chapter deals with time series classification theory.

3.1.2 Time Series Classification Definitions

Time series classification is the task of assigning a label to a time series.

A supervised classification settings is considered here: A set of time series and their respective labels, which make up the **train set**, is given.

In this setting, the objective becomes to build a classifier which predicts the label of a new instance, i.e. a new time series not included in the train set, as accurately as possible.

Classifier accuracy is the ratio of time series with correctly predicted classes divided by the total number of time series. In general, classifier accuracy is computed from a **test set**, a set of labeled time series not used in the classifier build process.

Mathematically, with some of the following formulae being adapted from Xing, Pei, and Philip, (2009):

Definition 5 (Time Series Classification). *Let* $T' = \{(ts, l_{ts}) : ts \text{ is time series} \wedge l_{ts} \text{ is label of } ts\}$ *be a time series test set,* $\mathcal{L} = \{l(ts) : ts \in T'\}$ *be the set of labels of time series in* T', $l : T' \to \mathcal{L}$ *be a function that outputs the label of a time series, and* TS *any set of time series. Then, the time series classification task is defined as follows:*

$$\textit{Find a classifier } C : TS \to \mathcal{L} \textit{ s.t.}$$

$$\max_{C \in \mathcal{C}} Accuracy(C, T') = \max_{C \in \mathcal{C}} \frac{|\{C(t') = l(t') : t' \in T'\}|}{|T'|}.$$

This thesis employs the classifier's accuracy as a measure of the quality of the classifier's predictions, since it is used in time series classification literature in general, as seen before. Note that this definition does not impose a fixed length for the time series.

3.1.3 Classification Algorithms - 1-Nearest Neighbor

This section begins with the formal definition of 1-Nearest Neighbor search, which is followed by the definition of the distances used with it.

Given a train set of time series and an unclassified time series, the 1-Nearest Neighbor algorithm assigns to it the label of the nearest neighbor from the train set. The nearest neighbor therefore depends on the definition of a metric to measure closeness or distance, as the mathematical formulation shows:

Definition 6 (1-Nearest Neighbor). *Let T be a train set of time series, $l : T' \to \mathcal{L}$ be a function that outputs the label of any time series, $d : \mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}$ a distance metric and t_{new} an unclassified time series. Then, the 1-Nearest Neighbor classifies t_{new} as follows:*

$$1NN(t_{new}) = l_k \iff \min_{ts \in T} d(ts, t_{new}) = ts_k \wedge l(ts_k) = l_k$$

The 1-Nearest Neighbor algorithm has linear time complexity in the size of the train set: $\mathcal{O}(|T|)$.

The following two definitions feature two commonly used distance metrics for the 1-Nearest Neighbor algorithm. The first, used for the classification of univariate time series, is the Euclidean distance (Eric Weisstein, (2016a)):

Definition 7 (Euclidean distance). *Given two univariate time series $ts_x = \{(t, x(t)) : t \in \mathbb{N}, x(t) \in \mathbb{R}\}$ and $ts_y = \{(t, y(t)) : t \in \mathbb{N}, y(t) \in \mathbb{R}\}$, both with length equal to L, the euclidean distance d_{Euc} between them is given by:*

$$d_{Euc}(ts_x, ts_y) = \sqrt{\sum_{i=1}^{L} (x(i) - y(i))^2}$$

The second distance measure, Frobenius norm, is employed in the classification of multivariate time series (Eric Weisstein, (2016b)):

Definition 8 (Frobenius norm). *Let $ts_x = \{(t, x(t)) : t \in \mathbb{N}, x(t) \in \mathbb{R}^n\}$ and $ts_y = \{(t, y(t)) : t \in \mathbb{N}, y(t) \in \mathbb{R}^n\}$ be two multivariate time series, both with length equal to L. Let $X, Y \in \mathcal{R}^{L \times n}$ be matrix representations of, respectively, ts_x and ts_y. The rows of the matrices represent the temporal indexes of the time series, and its columns correspond to the n-dimensional time series values at the timestamp*

of a row's index. For example, $X(i,j) = x(i)[j]$ *stands for* ts_x*'s value at timestamp* i *on the* j*-th dimension. Then,* d_{Frob} *is defined as:*

$$d_{Frob}(ts_x, ts_y) = \sqrt{\sum_{i=1}^{L}\sum_{j=1}^{n} |X(i,j) - Y(i,j)|^2}$$

3.1.4 Classification Algorithms - Deep Learning

This section begins with basic deep learning definitions and then addresses how deep learning can be applied in the context of time series classification.

Deep Learning Definitions

As mentioned in the literature review, deep learning models consist of multi-layered neural networks. This theory section will not dive into every single mathematical derivations and detail of deep learning theory, but rather give an overview of its theoretical constructs found in this thesis.

The most granular unit of a deep learning structure is a neuron. A neuron is fully described by its input x_i, input weights w_i, transfer function $\sum$ and activation function f. The figure 3.1 by Candel A. et al., (2016) depicts a neuron and its components:

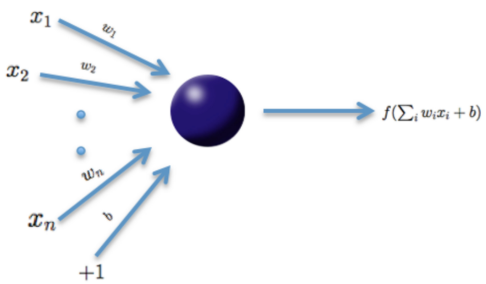


Figure 3.1: Diagram of a neuron and its components

A mathematical formulation for the neuron is given by the following:

Definition 9 (Neuron). *With $i \in \{1, \ldots, n\}$, let $x_i \in \mathcal{R}$ represent a set of inputs. Then, a neuron is a construct equipped with:*

- *$w_i \in \mathcal{R}$, a set of input weights*
- *$b \in \mathcal{R}$, a bias input unit weight associated with a constant input unit with the value 1. The bias represents the neuron's activation threshold.*
- *$\sum$, a transfer function that combines inputs with input weights. It can take many forms, but this thesis assumes it to be a dot product of the inputs with the input weights: $\sum_{i=1}^{n} w_i x_i + b$.*
- *f, an activation function that forms the neuron's output. It is typically a non-linear function, which maps the transfer function's output to $\mathcal{R}$. There are a number of different formulas and variations of this function, with the most common of them being (according to Batres-Estrada, (2015)):*
  - *Sigmoid function: $f : \mathcal{R} \to [0, 1], f(\alpha) = \frac{1}{1+e^{-k\alpha}}$ with $k \in]0, 1]$*
  - *Rectifier function: $f : \mathcal{R} \to \mathcal{R}^+, f(\alpha) = \log(1 + e^{1+\alpha})$, which can also be linear, as in $f(\alpha) = \max(0, \alpha)$*

Neurons are grouped into **layers**. These are formed by a set of neurons, which take the same set of inputs and feed their activations forward to the next layer. It is such a chaining of layers of neurons that composes a neural network. The following diagram, adapted from Candel A. et al., (2016), visualizes this description of a simplified neural network:

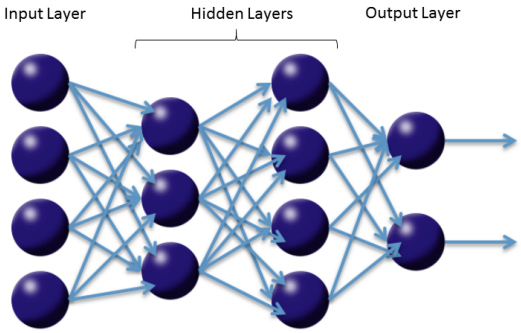


Figure 3.2: Diagram of a neural network

As figure 3.2 shows, neural networks are composed of three types of layers: an input layer, multiple hidden layers and an output layer. The input layer represents the input to the network, which gets transformed in the hidden layers. The end result of those transformations is shown in the output layer. Each neuron of a layer feeds its activation to all neurons of the following layer. Deep neural networks are, in general, those with a large number of hidden layers.

In the context of machine learning, neural networks are trained to transform certain input into desired output, i.e. to get an instance as input and output its class. For a neural network, training means optimally adjusting input weights of each of a neural network's layers, in order to minimize output error, i.e. classification error. Deep learning is the term used for training a deep neural network.

These notions are formalized by the deep learning definitions below:

Definition 10 (Deep Learning). *Let a deep learning construct be described by the following:*

1. *Deep neural network, with layers of neurons given as*
   a) *A matrix of neuron input weights* $W \in \mathcal{R}^M \times \mathcal{R}^N$ *where* M *is the maximum number of neurons over all layers (so the number of neurons per layer does not have to be the same for all layers) and* N *is the number of layers*
   b) *A vector of biases* $B \in \mathcal{R}^N$
   c) *Activation function* f
2. *Train set* T *of classified instances with* n *elements*
3. *Loss function* L*, a measure of the distance of the deep neural network output on an instance, the output* o_j *and its true or target value* t_j*. It can have a number of different formats, such as:*
   - $L(W, B) = \frac{1}{2} \sum_{j=1}^{n} (t_j - o_j)^2$*, the mean squared error*
   - $L(W, B) = -\sum_{j=1}^{n} \sum_{y \in O} (\log(o_{y,j}) * t_{y,j} + \log(1 - o_{y,j}) * (1 - t_{y,j}))$*, the cross entropy function, where* O *is the output layer*

The objective in deep learning is to minimize the loss function $L(W, B)$ *by searching for an optimal set of weights for both* W *and* B*.*

The objective to minimize the loss function guides the deep learning process to train the deep neural network. It can be achieved by a number of different search algorithms, most prominently **stochastic gradient descent** with backpropagation. A brief description of this algorithm follows.

Starting from randomized weight and bias matrices W and B, the algorithm iteratively picks a random instance i of a train set T. Then, for that instance, it updates weights in the direction of the steepest gradient descent as follows:

$$w_{jk} = w_{jk} + \alpha \frac{\partial L(W, B|i)}{\partial w_{jk}}, \forall w_{jk} \in W$$

$$b_{jk} = b_{jk} + \alpha \frac{\partial L(W, B|i)}{\partial b_{jk}}, \forall b_{jk} \in B$$

α is a parameter termed learning rate. It determines the size of the weight update and can be optimized via algorithms such as AdaDelta.

This process is repeated until some pre-defined convergence criterion is reached. It is combined with backpropagation, a process which uses the weights updates obtained via stochastic gradient descent to update the weights of a deep neural network one layer at a time. Each complete iteration through this process and the whole train set T is termed an epoch.

This optimization process may, in general, lead to overfitting train data. To counter this issue, a technique called **regularization** is applied. It consists of slightly modifying the loss function $L(W, B)$ as follows (as mentioned in Candel A. et al., (2016)):

$$L'(W, B) = L(W, B) + \lambda_1 R_1(W, B) + \lambda_2 R_2(W, B)$$

Again citing from Candel A. et al., (2016), $R_1(W, B)$ is defined as the sum of all l_1 norms (i.e. the absolute value of one parameter minus the other) for the weights and biases in the network. $R_2(W, B)$ represents the square root of the sum of squares of all the weights and biases in the network. Both λ_1 and λ_2 are very small constants, in the order of 10^{-5}. According to LISA lab, (2016), the effect of these two regularization functions is the improvement

of generalization: Both the regularization summands smooth the network weights by penalizing large values and thus decreasing non-linearity, over the complete neural network. It is, in some sense, a variation of Occam's razor principle, which allows one to find the "simplest" (least non-linear) solution that still fits the training data.

Deep Learning in Time Series Classification

After having established the deep learning theory for classification, this section addresses the use of deep learning in the time series classification context.

The challenge to the application of deep learning to time series classification is two-fold:

1. The choice of input features to the deep neural network
2. The configuration of the deep neural network

While deep learning approaches in general face the second challenge to varying degrees, the first challenge is specific to the time series domain and, in the literature, it is addressed depending on the context and application.

One example thereof is given by Busseti, Osband, and Wong, (2012). In course of energy demand forecasting, the authors encode periodicity and geographical features into hand-made features into their deep neural networks.

In another example, for the prediction of stock returns with deep learning, Batres-Estrada, (2015) uses, quote, "The input to our model is a matrix with 33 variables or features. These features are the t-2 through t-13 monthly log returns and the 20 daily log-returns for each stock at month t."

Therefore, both feature selection as well as parameter configuration for time series classification with deep learning will be addressed in the Implementation section later on.

3.1.5 Early Classification

Preliminaries

Early time series classification refers to time series classification using a fraction of total time series length.

As seen in the literature review, there are many approaches to reduce time series length while maintaining classification accuracy as high as possible. However, since one of the objectives of this thesis is to derive a trade-off curve for accuracy vs. time series length when performing early time series classification, the requirement on maintaining accuracy is relaxed. In general, time series classification accuracy decreases with decreasing time series length, since less and less time series values are available for classification. Therefore, to derive that accuracy-length trade-off curve, time series are firstly classified using their full length. Then, the time series length is successively reduced by a fixed percentage, until classification accuracy drops below a pre-defined threshold.

Definition

This approach is formalized by the following definitions, which, again, use notation partially borrowed from Xing, Pei, and Philip, (2009):

Definition 11 (Early Time Series Classification). *Consider the same assumptions of Definition 5. Additionally, let L_{ts} denote the length of a time series $ts \in T'$ and T'_p be the set of prefixes of time series in T' obtained by reducing L_{ts} of each $ts \in T$ by $r \in]0,1[$. Mathematically:*

$$ts_p \text{ is the prefix of length } p < L_{ts} \text{ of } ts \iff p = L_{ts} * (1 - r) \wedge ts_p = ts[1, p],$$

$$T'_p = \{ts_p : ts \in T \wedge ts_p = ts[1, p]\}.$$

The early time series classification task is defined as the time series classification task on the set T'_p.

Algorithm

The tradeoff curve mentioned above can be derived with the following algorithm:

Algorithm 1 Early Time Series Classification - Tradeoff Curve Derivation

Input: Train set of time series T
Test set of time series T'
Minimal classification accuracy threshold $mcat \in]0,1]$
Time series reduction stepsize $r \in]0,1]$
Classifier Algorithm CA
Output: Tradeoff Curve $TC_{CA} :]0,1[\rightarrow [mcat, 1]$

1: $currentAccuracy \leftarrow 1$
2: $currentLength \leftarrow 1$
3: $tradeoffCurveSet \leftarrow \{\}$
4: $currentTrainSet \leftarrow T$
5: $currentTestSet \leftarrow T'$
6: **while** $currentAccuracy \geq mcat \wedge currentLength > 0$ **do**
7: $\quad currentTrainSet \leftarrow \{ts[1, ceiling(L_{ts} * currentLength)] : ts \in T\}$
8: $\quad currentTestSet \leftarrow \{ts[1, ceiling(L_{ts} * currentLength)] : ts \in T'\}$
9: $\quad currentClassifier \leftarrow trainClassifier(CA, currentTrainSet)$
10: $\quad currentAccuracy \leftarrow testClassifier(currentClassifier, currentTestSet)$
11: $\quad tradeoffCurveSet \leftarrow$
$\qquad tradeoffCurveSet \cup \{(currentLength, currentAccuracy)\}$
12: $\quad currentLength \leftarrow currentLength - r$
13: **end while**
14: **return** $interpolatePointsOf(tradeoffCurveSet)$

Note that, in steps 7 and 8, it may be the case that $L_{ts} * currentLength \notin \mathcal{N}$. The function "ceiling" rounds the result of that multiplication to the next largest natural number.

This thesis proposes an alternative for step 12 simply called step 12': One can also perform the length reduction step of the algorithm, step number 12, by multiplying "currentLength" with "reductionStep", instead of performing subtractions. For small reduction step sizes, this has the effect of more

rapidly reducing "currentLength" while it is still large (i.e. very close to 1), and gradually reduce it by smaller steps each time. This in turn leads to more sampling points towards the end of the tradeoff curve, making it smoother around critical length values where accuracy may (or may not) drop below the minimal classification accuracy threshold with each step. On the other hand, for large reduction step sizes, this alternative for time series length reduction leads to more sampling points around the beginning of the tradeoff curve and gradually less and less points, as the length decreases. This means that the step 12' variation is also useful in cases where classification accuracy drops very rapidly with time series length, since it would also capture larger critical length values (which are, again, occasions where classification accuracy may drop below the minimal classification accuracy threshold with each step).

Step 14 of the algorithm above refers to any interpolation function, defined over $[0,1] \times [0,1]$, for the accuracy vs. time series length tradeoff points obtained by the algorithm.

This algorithm's complexity entirely depends on the classification algorithm's complexity. The complexity of the former is merely the result of the multiplication of the complexity of the latter by some constant k, which represents the number of time series length reduction steps. It holds that

- $k \leq 1/r$,
- $k = 1/r \iff \forall_{currentLength>0}\ currentAccuracy \geq mcat$.

This, which results from the algorithm also taking time series length as part of its stopping criterion, guarantees it will eventually halt.

This approach is similar to the 1-Nearest Neighbor fixed approach of Xing, Pei, and Philip, (2009), as described in the literature review. The main similarity lies in the repetition of the time series lengths reduction until some classification accuracy threshold is met or undercut. However, both 1-Nearest Neighbor fixed as well as the Early Classification of Time Series framework proposed by those authors do not deal neither with time series of varying lengths, nor with multivariate time series.

This approach may also be compared to Parrish et al., (2013) and Mori et al., (2016). Both feature early time series classification in a probabilistic setting, and Mori et al., (2016) also makes use of a set of previously defined time

series length percentages for early classification, similarly to the reduction step input parameter of the algorithm proposed above. However, none of the above mentioned approaches aims to derive a classification accuracy vs. time series length tradeoff curve. Furthermore, both of them are designed for a specific classification algorithm.

Therefore, this thesis stresses again the fact that the early classification algorithm described above is independent of the classification algorithm.

This thesis proposes, however, a variation for the early classification tradeoff curve derivation algorithm specific to the 1-Nearest Neighbor search on time series with reduced length. Given a reduced time series, this variation consists in:

1. Forecasting the missing tail of the time series with the aforementioned ARIMA or linear models
2. Appending that forecast of the missing values to the time series with unknown label
3. Searching for the 1-Nearest Neighbor of that enhanced time series in the train set of unreduced time series, i.e. those with full length

This strategy is proposed as a means to mitigate the information loss contracted in time series length reduction at each step of early classification.

Having presented the algorithm-agnostic early classification tradeoff curve derivation, its implementation, with the different classification algorithms outlined above, is the topic of the following chapter.

3.2 Implementation

This contents of this section revolve around the practical implementation of the theory outlined previously. It starts with some finer details regarding the algorithm implementations, and continues with a brief description of software frameworks employed.

3.2.1 Algorithm implementation details

Some finer points on the configuration of the time series classification and of the early time series tradeoff curve derivation algorithms are discussed here.

General Notes on the Implementation of Time Series Classification

In a supervised machine learning context, which is the case of every single algorithm presented and used by this thesis, there is a train and a test set. Nevertheless, not all datasets are already made available with separate train and test sets, which implies that either a split or cross-validation of different split sizes must be computed for datasets without such a split.

A choice for a split of the whole data in 60% train data and 40% test data was made for the 2 latter datasets. The reasons behind this decision are two-layered: Firstly, the widely-cited UCR datasets average a 45% train-to-test ratio, so this provides an anchor value and a starting point for benchmarking this thesis' results with those obtained by literature using UCR, which typically does not apply cross-validation (perhaps with the notable exception of Mori et al., (2016)). Secondly, train-to-test ratios like 80/20, 70/30 or 60/40 are very commonly used in literature, when not enough data is available for cross-validation. The choice fell on the train-to-test dataset with the next smallest ratio of train samples, the 60% train-to-test dataset split.

Note that the elements contained in each of split datasets are randomized before each run of the (early) classification algorithms.

In an unrelated topic, which is nevertheless valid for all time series classification algorithms considered in this thesis, the issue of comparing train and test time series of different lengths is also mentioned here. In general, that issue affects both the 1-Nearest Neighbor search and Deep Learning. The issue is that both the Euclidean distance as well as the Frobenius distance respectively assume input that is equally dimensioned. Distances of unequally dimensioned input are otherwise undefined. The analogue goes for Deep Learning.

There are a number of approaches to deal with this issue, and this thesis evaluated two of the most common ones: Either compare only the minimum of the dimensions both elements share, i.e. reduce the longer series to the length of the shortest, or fill up the shorter one with zeroes and compare that modified one with the longer one, i.e. enhance the shorter one with zeroes to match the length of the longest. This thesis employed the latter approach. The main reason behind this is that the tail of zeroes in the shorter series will lead to larger distances in those portions of the time series. Length then becomes, to some extent, also an indicator for time series (dis-)similarity. Thus, the choice of filling the shorter time series with zeroes will tend to penalize time series strongly differing in length. In this case of enhancing the shorter time series with zeroes, this difference in lengths is being implicitly encoded in the time series comparison with whichever algorithm. This implicitness of differing time series lengths gets lost if both time series get reduced to the length of the shorter one, and time series length information is a crucial factor in the context of early classification.

1-Nearest Neighbor for Time Series Classification

The implementation of the 1-Nearest Neighbor search for both univariate and multivariate time series very closely follows the theory presented in 3.1.3 and therefore dispenses further elaboration.

The variation of the early classification tradeoff curve derivation algorithm, with 1-Nearest Neighbor search and time series forecasts, employed the time series forecast models presented in 3.1.1, which are the ARIMA and linear models.

Deep Learning for Time Series Classification

As mentioned previously, the first thing to be covered here will be the selection of time series features to be fed as input for deep learning models.

In short, the time series features used for deep learning are simply the complete time series values.

In the case of an univariate time series $ts = \{(t, x(t)) : t \in \{1, \ldots, n\}, x(t) \in \mathbb{R}\}$, with $x : \mathbb{N} \rightarrow \mathbb{R}$, the deep learning model takes as input nodes the values $\forall_{t=1,\ldots,n} x(t)$.

As far as multivariate time series are concerned, consider $ts = \{(t, x(t)) : t \in \{1, \ldots, n\}, x(t) \in \mathbb{R}^m\}$, with $x : \mathbb{N} \rightarrow \mathbb{R}^m$. Then, the input nodes of the deep learning model become $\forall_{t=1,\ldots,n} \forall_{k=1,\ldots,m} x_k(t)$, i.e., a given input node represents the value of the time series at time t, for dimension k. Therefore, the cardinality of the input nodes is equal to $n \times m$.

Note that the deep neural network, in the context of classification, outputs instance classes. In particular, this means that the deep neural networks used here always have an output layer of cardinality equal to the number of number of classes in a dataset. As described by Mnih and G. E. Hinton, (2009), this is achieved with the use of a softmax layer as the output layer, a layer which outputs class probabilities for the deep neural network's input. It normalizes the previous layer's vector output to real values in the interval $[0, 1]$, by exponentiating each value and dividing that by the sum over all exponentiated values.

The second thing to be addressed here is the configuration and choice of parameters for the deep learning model. The choice of deep learning models involves a lot of degrees of freedom, such as the number of hidden layers, the neurons' activation function or the loss function to be optimized. The parameters available for configuration were all explained in the deep learning theory section, numbered 3.1.4, with the following exception, which is more of a practical nature than specific to deep learning theory, hence it being mentioned here. For the small datasets, a class balancing step was used. Class balancing is a term employed to refer to the process of over-sampling underrepresented classes in the training phase of the deep learning model. This proved to be invaluable to up the performance of the deep learning model in cases where the number of samples with a certain class was simply both absolutely as well as relatively too low for the algorithm. More details on this will follow in the result presentation and discussion sections later on.

To make these parameter choices for the deep learning models, an extensive grid search, which consists of successively testing different parameter config-

urations, was performed. The resulting parameter values will be presented alongside the results.

Early Classification Tradeoff Curve Derivation

The early classification tradeoff curve derivation algorithm, described in 3.1.5, takes train and test sets, a classification algorithm, the minimal classification accuracy threshold and a time series reduction step size as input parameters.

The algorithms considered for the early classification tradeoff curve derivation are all of the above mentioned, which are in summary:

- Univariate time series:
  - 1-Nearest Neighbor search with the Euclidean distance on the reduced time series and on the reduced time series plus the forecasts provided by ARIMA and the linear models as well
  - Deep learning
- Multivariate time series:
  - 1-Nearest Neighbor search with the Frobenius distance
  - Deep learning

The minimal classification accuracy threshold used was 60%.

As far as the reduction step parameter is concerned, the values $\{0.01, 0.05, 0.1, 0.25, 0.5\}$ were used in experimentation. However, both alternatives proposed for step 12 were used, with the reduction step values being set at $\{0.5, 0.9\}$. The exact value was chosen depending on the dataset at hand and will therefore be presented along with the results.

The interpolation of the points obtained with the early classification tradeoff curve derivation algorithm is a simple linear interpolation, where each point is connected with the following one via a line segment.

3.2.2 Frameworks

Some of the considered datasets were processed with the R language and environment for statistical computing by R Core Team, (2015)[1]. Others, on the other hand, were processed with the Java software stack[2].

The R software stack was chosen due to R's powerful primitives for manipulating time series data and also due to its rich package ecosystem. In particular, the deep learning package H_2O, developed by Aiello et al., (2016), was chosen for deep learning-related tasks, due to its tight integration with the R programming environment, and the time series modeling and forecasting package by R. Hyndman, (2016), which provides a simple interface for creating and automatically deriving linear and ARIMA models, and using them for forecasts in early classification as previously described. Finally, the package "class" by Venables and Ripley, (2002) was used for 1-Nearest Neighbor search with the Euclidean distance.

The reason for using Java software for certain other datasets lies in the Java API that came with them. The integration with such datasets is thus simpler, despite the deep learning library H_2O by Candel A. et al., (2016)[3] not having a Java API. That issue was overcome with the use of H_2O's REST API, accessed via simple HTTP requests. For the latter, the library OkHttp of Square, (2016)[4] was used.

All the implementation, experiments and result derivation were conducted using a laptop with an "Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz" processor and 4GB main memory.

[1]The R version used by this thesis is 3.2.3.

[2]The Java version used by this thesis is "openjdk 1.8.0_91".

[3]The h2o version used by this thesis for deep learning with the snackbox dataset was 3.8.0.6. Refer to Aiello et al., (2016) for more information on the h2o R package.

[4]This thesis used the 3.2.0 version of the OkHttp library.

3.3 Summary

In this chapter, the research questions guiding this thesis on the early classification approach were successively addressed, namely which time series it works on, what the tradeoff between length and accuracy is and which algorithms, in particular deep learning, perform best at it.

To that end, first, the theory behind univariate and multivariate time series was covered. Then, different classification algorithms for both types of time series were outlined, and an algorithm for performing early time series classification was presented. This algorithm also focuses on comparing the performance of different time series classification algorithms regarding the early time series classification task, by benchmarking their accuracy at reduced time series lengths. This section concluded with the description of the implementation of all that theory, for time series datasets with varying characteristics to be presented in section 4.1.

After having covered these topics, it is time to assess the algorithms' performance and then evaluate the results.

4 Results

In the search for time series and algorithms adequate for early classification, theory and implementation details around the derivation of a tradeoff curve for time series length vs. accuracy was proposed for different time series classification algorithms. This chapter presents the results of the proposed approach, for three different datasets, namely UCR, Auslan and Snackbox, and the algorithms 1-Nearest Neighbor with the Euclidean distance, its variation with forecasts provided by linear models and ARIMA, 1-Nearest Neighbor with the Frobenius distance and Deep Learning. The datasets are presented first. Then, results on the univariate time series datasets follow, and this section concludes with the results on the multivariate time series datasets Auslan and Snackbox.

4.1 Datasets

To address the research question *which types time series allow for early classification*, this thesis considers three different sets of time series, with each of them having differing characteristics. The datasets' general properties are summarized here:

Dataset	Time series type	Classes	Samples
UCR[1]	Univariate	$[2, 60]$	$[40, 16637]$
Auslan	Multivariate	95	2565
Snackbox	Multivariate	5	60

Table 4.1: Dataset overview

One of the research questions guiding this thesis is to assess the types of time series where early time series classification is feasible. Hence the choice of the datasets listed in table 4.1: These should represent both univariate as well as multivariate time series, synthetical and clean as well as real-world and noisier time series, and, finally, the datasets used here should be representative of those often used in early time series classification literature, to allow for result comparison.

A more detailed presentation of each of the datasets follows.

4.1.1 UCR

The UCR time series classification archive by Chen et al., (2015), referenced throughout this thesis simply as UCR, is a repository of univariate time series. The datasets therein are *very* commonly used for benchmarking machine learning approaches on time series, as evidenced by the works of Xing, Pei, and Philip, (2012), Parrish et al., (2013) or Mori et al., (2016), to cite a few papers in the time series (early) classification research area alone.

This time series repository consists of 85 different time series datasets, each with a varying amount of labeled time series samples, number of classes and time series lengths. The common aspects they all share are that all of them are univariate (in accordance with this thesis' definition of univariate time series), feature separate train and test sets, have equidistant observations, have the same length (within a dataset), are cleaned and z-normalized, i.e. normalized to have mean zero and standard deviation of 1, and are available in a simple comma-delimited file format. The time series themselves are very varied in nature: They consist of synthetic time series generated randomly, of curves consisting of a single cylinder or bell or funnel, of electrocardiography measurements, or even of sensor data from movement generated while pointing a gun, to name a few examples.

In general, these time series encode some natural or random process in a short, cleaned and contrived format, and thus reflect only such types of

[1]Since UCR is actually a repository of time series datasets, the statistics are given as intervals, ranging from UCR's smallest datasets to its largest ones.

series. While the repository does offer a high measure of variety on such time series and is thus used in time series classification benchmarks quite often, the repository's authors and maintainers admittedly recognize its limitations as models of real-world problems, which is reinforced by Hu, Chen, and E. J. Keogh, (2013).

4.1.2 Auslan

Auslan is short for Australian sign language. The next figure, taken from M. W. Kadous, (2002), features an example of such a sign:

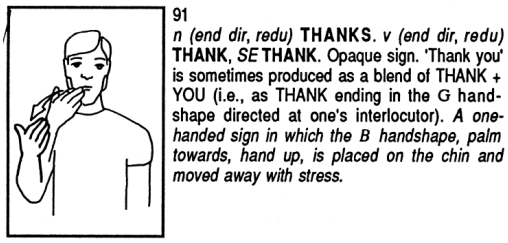


Figure 4.1: Example of a sign in the Australian sign language (Auslan)

The Auslan dataset is comprised of samples of Auslan signs, which, quote, "were captured from a native signer with high precision position trackers and instrumented gloves", as the author of the dataset, M. W. Kadous, (2002), puts it in his PhD thesis.

The dataset contains 27 examples of each of 95 different such signs, for a total of 2565 captured samples. Those samples each contain sign movements as measured by 11 different variables per hand, for a total of 22 measurements per frame. These variables are, per hand, the (x, y, z) position, roll, pitch, yaw and a finger bend measure for each of the fingers. The signs average about 57 frames in length. There are no missing values in the multivariate time series, as each frame or time stamp of the time series features all 22 variable values, and they are zero if no measurement is performed in a given frame. This dataset also features equidistant observations, where the distance between them is a frame.

This dataset is very often used in the early time series classification literature, for example by Juan J Rodriguez, J. J. R. Guez, and Carlos J Alonso, (2002) or Xing, Pei, and E. Keogh, (2010). For the purposes of this thesis' first research question, the one regarding the conditions which need to be in place to allow for early time series classification), this dataset is, firstly, a source of multivariate time series, and secondly, it is regarded as representative of a real-world problem, which includes noisy data.

4.1.3 Snackbox

The Snackbox dataset contains both some sensor measurement raw data, as well as the result of processing and structuring that raw data.

In the offices of Know-Center, (2016), there is a storage room with a box with snacks. Within that snackbox is a coin deposit, where people taking snacks are expected to provide payment. The figure 4.2 depicts the snackbox:

Figure 4.2: Screenshot of the snackbox and its sensors

In general, the money in the coin deposit represents a lower financial value than that of the snacks missing in the snackbox. To understand this discrepancy better, the snackbox room was equipped with sensors like microphones, ambient light sensors, motion detectors and thermometers from TinkerForge, (2016). The raw data of the snackbox dataset thus corresponds to the capture of those sensors' raw data over a period of time.

That raw data was then structured, with the purpose of performing machine learning on it.

Firstly, the raw data, modeled as multivariate time series, was sectioned into windows representing around a second worth of measurements. Then, 300 features in the form of descriptive statistics, like different types of means, standard deviations or skewness and kurtosis coefficients, were computed for each of the sensors, for each of the previously mentioned windows. These windows of features are then called observations.

Via a number of machine learning approaches, whose descriptions are out-of-scope for the purposes of this thesis, a sequence of observations are grouped into so-called phases, and phases are then themselves grouped into sequences called scenarios. Phases describe events related to the acquisition of snackbox snacks. There are 5 different phases:

1. ”default”: Nothing of note is happening
2. ”Entry”: Someone enters the snackbox room
3. ”Rustling”: A snack is taken from the snackbox
4. ”CoinToss”: A coin is deposited
5. ”Exit”: Someone exits the snackbox room

Each of those phases may have a different number of observations.

Scenarios describe a logical sequence of phases such as ”Entry”, followed by ”Rustling”, then ”CoinToss” and finally ”Exit”. There are a total of 4 types of scenarios and they thus distinguish between 4 types of activity measured in the room, including the distinction between phase sequences where a snack was taken and a coin was deposited for it or not.

There are a total of 20 different scenarios, which contained a combined total of 60 phases. The 60 phases, or sequences of observations, which are, again, features from a window of the original raw time series, compose the snackbox dataset’s multivariate time series used in this thesis for early classification. 3 of the 5 different phase types, namely ”Entry”, ”Exit” and ”Rustling”, have 15 samples each. The phases ”CoinToss” and ”default” however, respectively feature only 10 and 5 samples. Therefore, the latter two classes are underrepresented.

This dataset of multivariate time series represents a real-world problem, in a more complex and also realistic sense than the Auslan dataset. The reason for that directly relates with the cleaning and processing required of the raw, uncleaned and often missing sensor data and its grouping and structuring it into observations, phases and scenarios.

4.2 Results per Dataset

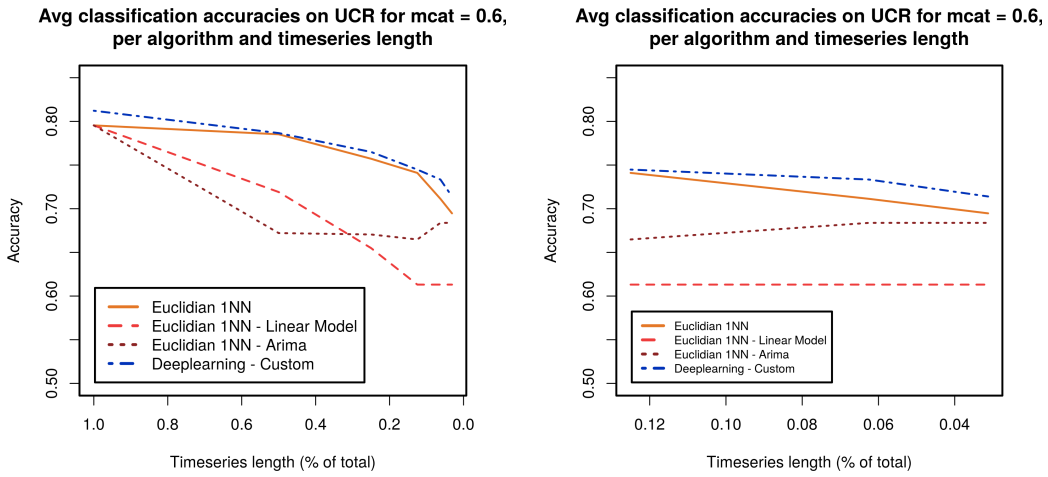
4.2.1 Univariate Time Series

The univariate time series datasets presented here all stem from the UCR time series repository by Chen et al., (2015). As mentioned previously in 2.2 and 4.1, this time series repository builds the most widely used corpus of time series datasets used in the time series classification context. It consists of 85 sets of different time series, each containing a varying number of both training and testing samples.

This thesis processed all 85 time series datasets in UCR. The results presented here were averaged over all datasets, but details on the performance obtained on each individual dataset can be inspected in the code repository annexed to this thesis.

UCR

Early classification tradeoff curves, averaged over all datasets in the UCR time series repository, are depicted, for four different algorithms, in the figures below.



(a) full curves

(b) tail of the curves

Figure 4.3: The plot on the left shows the 'classification accuracy vs. time series length' tradeoff curves for each of the classification algorithms applied to the UCR time series dataset. These curves represent averages of the tradeoff curves, computed over all datasets in the UCR repository per classification algorithm. The curves are the result of interpolating points (time series length, classification accuracy) gathered with 50% reductions of time series lengths, i.e. at 100%, 50%, 25%, ... of the original time series length. These reductions are repeated for a given algorithm until its classification accuracy drops below the minimal classification accuracy threshold (mcat) set beforehand, in this case 0.6. The plot on the right depicts the tail of the curves seen on the left.

Figure 4.3 shows 2 views of the same curves, with the left one showing the full early classification tradeoff curves for a number of algorithms. The right one shows the tail of the same curves.

The reduction step length used to produce these interpolated curves was 0.5, since classification accuracy only really becomes close to the minimal

classification accuracy threshold (mcat) of 60% around time series lengths of about 6% of the original time series full length. In this application of the Early Classification Tradeoff Curve Derivation algorithm, the step 12' variation of the reduction in time series length was used.

The algorithms employed for the derivation of the early classification tradeoff curves shown above were the following:

- 1-Nearest Neighbor search with the Euclidean distance in a number of variations mentioned below. See detailed definitions of the following algorithms in 3.1.3 and 3.2.1.
  - 1-Nearest Neighbor search with Euclidean distance, without any further modifications
  - 1-Nearest Neighbor search with Euclidean distance, with time series forecasts provided by linear models[2] for reduced time series.
  - 1-Nearest Neighbor search with Euclidean distance, with time series forecasts provided by ARIMA[2] for reduced time series.
- Deep learning with the following customized parameters, obtained via experimentation and grid search over the listed parameters, results in the model described below. Refer to section 3.1.4 for details on each parameter and refer to 3.2.1 for details on the input configuration.
  - Activation function: Rectifier
  - Number of hidden layers: 3
  - Neurons per hidden layer: 200
  - Loss function: CrossEntropy
  - Epochs: 100
  - Assumed distribution of the instance classes: Multinomial
  - Number of folds in cross-validation of train set: 5[3]

Note that the Deep Learning model above was the same for all of the 85 UCR time series datasets.

[2]The resulting linear and ARIMA models and their parameters are also available as part of this thesis' annexed code repository.

[3]5 cross-validation folds were used on every UCR dataset except for "OliveOil", which only employed 3 folds due to the dataset's small size.

Table 4.2 shows the exact values depicted in the figure 4.3 and the time it took to train and classify each round of reduced time series in the UCR repository datasets:

Time series length (%)		**Algorithm**			
		1-NN Euc (Euclidean)	1-NN Euc Linear Model	1-NN Euc Arima	Deep Learning Custom
100%	Avg. Accuracy	0.7953	0.7953[4]	0.7953[4]	0.8122
	Time (s)	2.428	2.428[4]	2.428[4]	123.63
50%	Avg. Accuracy	0.7852	0.7191	0.6721	0.7866
	Time (s)	0.60	13.03	523.27	120.80
25%	Avg. Accuracy	0.7572	0.6549	0.6706	0.7650
	Time (s)	0.24	7.05	32.97	96.73
12.5%	Avg. Accuracy	0.7410	0.6131	0.6648	0.7448
	Time (s)	0.08	8.78	34.41	92.41
6.25%	Avg. Accuracy	0.7113	0.6131	0.6838	0.7334
	Time (s)	0.05	8.74	12.44	80.84
3.125%	Avg. Accuracy	0.6947	0.6131	0.6838	0.7140
	Time (s)	0.03	4.28	8.83	74.92

Table 4.2: UCR early classification performance overview

[4]At 100% time series length, there is no reduced portion left to forecast using a linear model or ARIMA, so the listed accuracy and time simply corresponds to that of the 1-Nearest Neighbor search with the Euclidean distance.

Table 4.3 shows the number of datasets where the 1-Nearest Neighbor search with the Euclidean distance outperformed Deep Learning and vice-versa, again for the same minimal classification accuracy threshold of 0.6:

Time series length (%)	**Algorithm with higher accuracy (count)**	
	1-NN Euc (Euclidean)	Deep Learning Custom
100%	20	39
50%	20	34
25%	16	30
12.5%	14	21
6.25%	6	21
3.125%	3	17

Table 4.3: UCR early classification - direct comparison of 1-NN Euclidean with Deep Learning

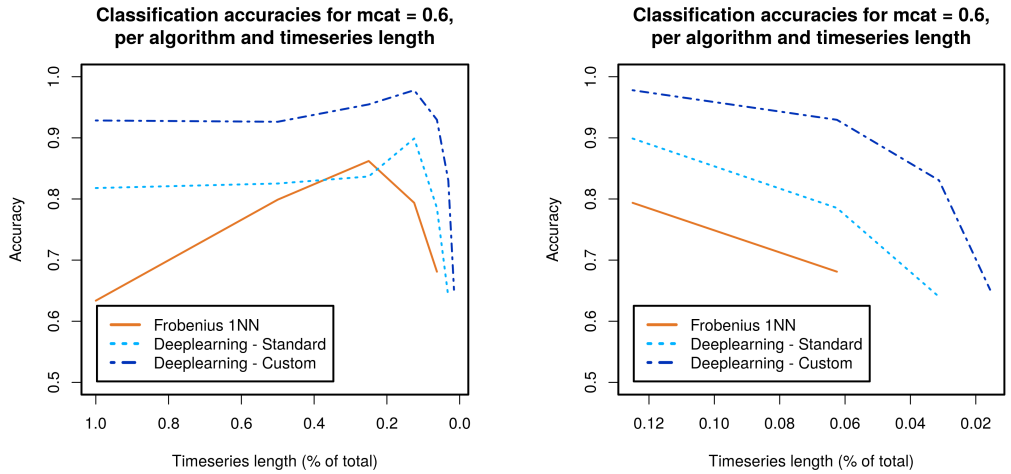
4.2.2 Multivariate Time Series

As mentioned in 4.1, both multivariate datasets considered here are comprised of sensor data type measurements. The first one, Auslan, has less measurements per time series timestamp, but more classes and more samples per class. The Snackbox dataset includes a total of 300 measurements per time series timestamp, but only features 5 classes and an average of 12 samples per class, with two of those 5 classes being significantly underrepresented.

In the following sections showing the obtained results, the effect those dataset characteristics have on early classification can be observed.

Auslan

The early classification tradeoff curve for this dataset is depicted, for three different algorithms, in the figures below.



(a) full curves (b) tail of the curves

Figure 4.4: The plot on the left shows the 'classification accuracy vs. time series length' tradeoff curves for each of the classification algorithms applied to the Auslan time series dataset. The curves are the result of interpolating points (time series length, classification accuracy) gathered with 50% reductions of time series lengths, i.e. at 100%, 50%, 25%, ... of the original time series length. These reductions are repeated for a given algorithm until its classification accuracy drops below the minimal classification accuracy threshold (mcat) set beforehand, in this case 0.6. The plot on the right depicts the tail of the curves seen on the left.

Figure 4.4 shows two views of the same curves, with the left one showing the full early classification tradeoff curves for a number of algorithms on the Auslan dataset. The right one shows the tail of the same curves.

The reduction step length used to produce these interpolated curves was 0.5, since classification accuracy only really becomes close to the minimal classification accuracy threshold (mcat) of 60% around time series lengths of about 6% of the original time series full length. In this application of

the Early Classification Tradeoff Curve Derivation algorithm, the step 12' variation of the reduction in time series length was used.

The algorithms employed for the derivation of the early classification tradeoff curves shown above were the following:

- 1-Nearest Neighbor with the Frobenius distance: See its definition in 3.1.3.
- Deep learning with standard parameters, as derived and computed by Aiello et al., (2016), results in the model described below. Refer to section 3.1.4 for details on each parameter and refer to 3.2.1 for details on the input configuration.
  - Activation function: Rectifier
  - Number of hidden layers: 2
  - Neurons per hidden layer: 200
  - Loss function: CrossEntropy
  - Epochs: 10 (on average)
  - Assumed distribution of the instance classes: Multinomial
- Deep learning with the following customized model, obtained via experimentation and grid search over the listed parameters:
  - Activation function: Rectifier
  - Number of hidden layers: 3
  - Neurons per hidden layer: 200
  - Loss function: CrossEntropy
  - Epochs: 100
  - Assumed distribution of the instance classes: Multinomial
  - Number of folds in cross-validation of train set: 5

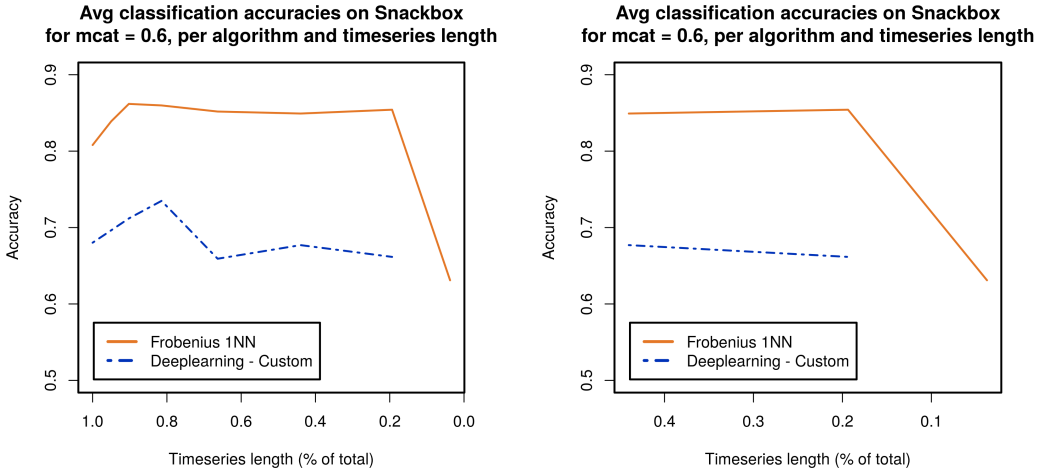
Table 4.4 shows the exact values depicted in the figure 4.4 and the time it took to train and classify each round of reduced time series in the Auslan dataset:

Time series length (%)		**Algorithm**		
		1-NN Frobenius	Deep Learning Standard	Deep Learning Custom
100%	Accuracy	0.6337	0.8179	0.9284
	Time (s)	2034.62	58.74	752.53
50%	Accuracy	0.7989	0.8253	0.9263
	Time (s)	1938.95	47.05	564.09
25%	Accuracy	0.8621	0.8368	0.9547
	Time (s)	1893.74	29.56	631.96
12.5%	Accuracy	0.7937	0.8989	0.9779
	Time (s)	1869.85	19.96	849.64
6.25%	Accuracy	0.6811	0.7853	0.9295
	Time (s)	1871.06	14.61	676.83
3.125%	Accuracy	-	0.64	0.8305
	Time (s)	-	11.03	579.15
1.5625%	Accuracy	-	-	0.6516
	Time (s)	-	-	530.12

Table 4.4: Auslan early classification performance overview

Snackbox

The early classification tradeoff curve for this dataset is depicted, for three different algorithms, in the figures below. Note that these graphics represent average classification accuracies, computed over a set of 50 iterations of the early classification tradeoff curve derivation algorithm on the Snackbox dataset. Individual iteration results are available on the annex of this thesis.



(a) full curves (b) tail of the curves

Figure 4.5: The plot on the left shows the 'classification accuracy vs. time series length' tradeoff curves for each of the classification algorithms applied to the Snackbox time series dataset. These curves represent averages of the tradeoff curves, computed over 50 iterations of algorithm 1 on the Snackbox dataset, per classification algorithm. The curves are the result of interpolating points (time series length, classification accuracy) gathered with 95% reductions of time series lengths, i.e. at 100%, 95%, 90.25%, ... of the original time series length. These reductions are repeated for a given algorithm until its classification accuracy drops below the minimal classification accuracy threshold (mcat) set beforehand, in this case 0.6. The plot on the right depicts the tail of the curves seen on the left.

Figure 4.5 shows 2 views of the same curves, with the left one showing the full early classification tradeoff curves for a number of algorithms on the Snackbox dataset. The right one shows the tail of the same curves.

The reduction step length used to produce these interpolated curves was 0.95, since, algorithms applied on this dataset suffer from rapid classification accuracy drops with relatively small reductions in time series length. In this dataset, the algorithms reach the minimal classification accuracy threshold of 60% already at around time series lengths of about 40% of the original time series full length. In this application of the Early Classification Tradeoff Curve Derivation algorithm, the step 12′ variation of the reduction in time series length was used.

The algorithms employed for the derivation of the early classification trade-off curves shown above were the following:

- 1-Nearest Neighbor with the Frobenius distance: See its definition in 3.1.3.
- Deep learning with the following customized model, obtained via experimentation and grid search over the listed parameters (refer to 3.1.4 and 3.2.1 for more information on each parameter and input configuration):
  - Activation function: Rectifier
  - Number of hidden layers: 7
  - Neurons per hidden layer: 70
  - Loss function: CrossEntropy
  - Epochs: 10 (on average)
  - Class balancing: True
  - λ_1: $10^{-}4$
  - λ_2: $10^{-}4$

Table 4.5 shows the exact values depicted in the figure 4.5 and the time it took to train and classify each round of reduced time series in the 50 iterations of the early classification tradeoff curve derivation algorithm over the Snackbox dataset:

Time series length (%)		**Algorithm**	
		1-NN Frobenius	Deep Learning Custom
100%	Avg. Accuracy	0.808	0.6801
	Time (s)	1.16	77.24
95%	Avg. Accuracy	0.839	0.6964
	Time (s)	1.12	73.48
90.25%	Avg. Accuracy	0.8618	0.7119
	Time (s)	0.42	79.27
81.45%	Avg. Accuracy	0.8598	0.7349
	Time (s)	0.4	89.94
66.34%	Avg. Accuracy	0.8518	0.6591
	Time (s)	0.37	91.57
44.01%	Avg. Accuracy	0.8492	0.677
	Time (s)	0.32	94.88
19.37%	Avg. Accuracy	0.8542	0.6616
	Time (s)	0.28	108.24
3.752%	Avg. Accuracy	0.631	-
	Time (s)	0.24	-

Table 4.5: Snackbox early classification performance overview

Table 4.6 shows the number of early classification algorithm iterations where the 1-Nearest Neighbor search with the Frobenius distance outperformed Deep Learning and vice-versa, again for the same minimal classification accuracy threshold of 0.6:

Time series length (%)	**Algorithm with higher accuracy (count)**	
	1-NN Frob (Frobenius)	Deep Learning Custom
100%	48	2
95%	42	5
90.25%	35	6
81.45%	30	7
66.34%	26	4
44.01%	21	3
19.37%	17	2
3.752%	1	-

Table 4.6: Snackbox early classification - direct comparison of 1-NN Euclidean with Deep Learning

4.3 Summary

This chapter presented the early time series classification results achieved on 3 different datasets of different types, for a number of time series classification algorithms. The data sets presented in this chapter, namely UCR, Auslan and Snackbox, include both theoretical as well as real world examples. The UCR dataset contains a total of 85 univariate time series, whereas the Auslan and Snackbox both comprise only multivariate time series. The results obtained on those data sets consist of graphical representations of the early classification tradeoff curves of the 1-Nearest Neighbor searches and Deep Learning for each of the datasets considered. Overall performance overview of those algorithms are complemented by a zoomed-in view of the curves' tails, which represent the critical region where classification accuracy drops below the pre-defined mininmal classification accuracy threshold. The data underlying those plots was also shown in tabular form. These results serve

as the empirical basis for the evaluation of the research questions guiding this thesis. The discussion thereof and, in particular, also the limitations of these results, will be addressed and discussed in the next chapter.

5 Discussion

Having presented the overall results in the previous chapter, this chapter aims to make sense of them and draw conclusions to help answer the research questions guiding this thesis. While this thesis claims to have answered those, it does not claim its results do not have its limitations, so those will addressed here as well. This chapter's structure follows that of the results chapter: The discussion starts with the UCR univariate time series datasets and continues with the multivariate ones, with the discussion of Auslan being followed by that of the Snackbox dataset. Finally, a reflection on the limitations of this work rounds up this chapter.

5.1 Observations per Dataset

The reason for discussing the results obtained per dataset separately instead of overarching, general observations covering all datasets directly relates to its limited comparability between datasets. As will be seen below, while there are commonalities between them, the number of unique aspects to each of them justified the separation of observations made drawn per dataset.

5.1.1 Univariate Time Series

UCR

Figure 4.3 plots the classification accuracies for the full and reduced time series lengths, obtained per algorithm and averaged over all time series datasets of the UCR repository. The characteristics of the early classification tradeoff curves derived for the univariate time series of the UCR

dataset seem, graphically, to be quite clear: In general, the curves seem to feature an overall decreasing trend. Classification accuracy, irrespective of the classification algorithm, drops steadily with decreasing time series lengths. The "1-Nearest Neighbor search with forecasts by ARIMA models" appears to be, however, the sole exception to this downwards trend, but this apparent exception will be explained later. The Deep Learning model edges out 1-Nearest Neighbor search in classification accuracy throughout all time series lengths. The performance gap seems to be larger along both the beginning and the tail of the early classification tradeoff curves. The algorithms 1-Nearest Neighbor with forecasts by linear and ARIMA models are completely dominated by the other two classification approaches, with the linear models being more accurate for the larger time series lengths and the ARIMA models outperforming the linear ones as time series lengths get shorter and shorter. Although all algorithms classify, with accuracy above the minimal classification accuracy threshold (mcat) of 60%, some UCR dataset of reduced time series measuring 3.125% in length, no algorithm maintains an accuracy above that mcat for even shorter time series lengths, for all of the UCR time series datasets.

A closer look in table 4.2, which lists the exact accuracies (averaged over the UCR datasets) and timings of each of the classification algorithms, for the full and each of the reduced time series lengths, allows for deeper insights into the results obtained in the UCR datasets. The graphically apparent dominance of the Deep Learning algorithm with respect to 1-Nearest Neighbor search with the Euclidean distance translates to a maximal difference in average accuracy between them of only about 2% for the time series lengths 100%, 6.25% and 3.125%. All other differences in accuracy are around 1% or even less. Given the constant Deep Learning model over all 85 datasets, it is interesting to observe that it still, in general, performs quite well. This thesis attributes that fact to the comparatively large size of the deep neural network used here, which covers the complexity contained in the relatively short time series of the UCR dataset. Table 4.2 also confirms the significant performance gap between those two classification methods and the 1-Nearest Neighbor search with the Euclidean distance and forecasts provided by linear models and ARIMA, since the latter have lower average classification accuracies throughout all time series lengths, and mostly by a margin of at least 2%. The poor performance of the 1-Nearest Neighbor

algorithms enhanced by forecasts by linear models and ARIMA is rooted at the structure of the time series themselves. These two time series models are geared towards time series with trend and seasonality components. ARIMA in particular assumes stationarity of the time series. However, the time series in the UCR dataset are, in general, not stationary and do not have seasonality or trend components. The UCR time series are aimed towards classification and not forecasting tasks, hence the poor forecasts by these models, which end up penalizing classification accuracy of the 1-Nearest Neighbor search. Table 4.2 shows, however, one of the weaknesses of the Deep Learning: its slowness in training and classifying the time series. It pales in comparison with the run-times of all the other algorithms, since, in the worst case, it is 3 orders of magnitude larger than that of the 1-Nearest Neighbor search with the Euclidean distance. It takes, on average, still over a minute to train and test a Deep Learning model even on time series with their length reduced to 3.125% of the original length. The 1-Nearest Neighbor search algorithms with the linear model forecast method runs on average in around 10 seconds, while the ARIMA ones take about 30 seconds, with the notable exception of the run-time for the time series length of 50%. This high average value is very strongly skewed by a couple of outlier time series datasets, where the automated model fitting and estimation by R. Hyndman, (2016) take a very long time. This is, again, related to the unexpected time series, from the perspective of both the ARIMA and linear models. For these datasets, the classification accuracy achieved with this method barely tops the minimal classification accuracy threshold, so the other average accuracies and times of this classification algorithm are more representative of its actual performance.

In general, the fact that, for time series lengths shorter than 12.5%, there were at most 3 out of 85 datasets where the classifications algorithms using linear models and ARIMA still perform above the minimal classification accuracy threshold explains a lot. It is the main reason for the rather surprising uptake in average classification accuracy towards the end of the early classification tradeoff curve of the classifier with ARIMA and also the constant tail of the classifier with linear models.

Since these two algorithms never perform better, on average, than the simple 1-Nearest Neighbor search with the Euclidean distance and the Deep Learning models, focus is directed towards the latter algorithms with a table

listing the number of UCR datasets where Deep Learning outperformed the 1-Nearest Neighbor and vice-versa (see table 4.3). This table aims to complement the view on the results provided by table 4.2, whose averages hide some insights into the results obtained. The table shows that, even though Deep Learning did not have a significantly higher average accuracy than the 1-Nearest Neighbor search with the Euclidean distance, the latter achieved higher classification accuracy than the former on a higher amount of UCR datasets for all reduced time series lengths. This is another indicator for the comparatively solid results achieved by Deep Learning, besides the small improvements in average classification accuracy over the simple 1-Nearest Neighbor classifier with the Euclidean distance.

To conclude the discussion of the results on the UCR time series repository, the Deep Learning methods show potential in keeping up with the performances of both the 1-Nearest Neighbor search with the Euclidean distance and perhaps even with 1-Nearest Neighbor search with dynamic time warping, as referenced in Chen et al., (2015). Note, however, that, while 1-Nearest Neighbor search with the Euclidean distance cannot be further improved, different and dataset-specific parameter choices for the Deep Learning models may lead to even better results. This is, however, also a downside of the latter method, since model configuration may become very complex and hyperparameter optimization a never ending task. Furthermore, the very long run-times may also deter one from choosing a Deep Learning model over the very performant (in the speed aspect) 1-Nearest Neighbor search. In general, for these two algorithms, early classification of the UCR time series seems feasible, if a drop in average accuracy of up to 10% is an acceptable loss for the benefit of outputting a time series classification decision with only a fraction of e.g. 3% of total time series length. The results obtained with the variations of the 1-Nearest Neighbor with forecasts provided linear models and ARIMA are, especially in comparison with the other algorithms, rather unsatisfactory, since they are dominated both in terms of average accuracy as well as in the average train and classification run times. This is a consequence of the composition of the time series themselves, which is not appropriate for these forecasting algorithms.

5.1.2 Multivariate Time Series

Auslan

In the dataset of multivariate time series "Auslan", the figure 4.4 plots the classification accuracies for the full and reduced time series lengths, obtained for a typical run of the algorithms 1-Nearest Neighbor search with the Frobenius matrix norm and two different Deep Learning models, with the reason for the latter being addressed later on. The tradeoff curves pictured there exhibit a peculiar behavior, in the sense that classification accuracy seems to increase with reducing time series length, up to a certain point, for all algorithms. This is counter-intuitive to the notion that classification accuracy decreases with shorter time series lengths. This is justified by the fact that many of the Auslan signs time series share a high degree of similarity between them in their end portions. Time series length reductions cut those tails many Auslan signs have in common, improving overall distinction of the Auslan signs time series and thus leading to higher classification accuracy overall. With this dataset, Deep Learning methods quite clearly outperform the Frobenius 1-Nearest Neighbor algorithm, both in classification accuracy as well as in terms of the time series length reductions achieved before dropping below the minimal classification accuracy threshold. As far as the Deep Learning algorithms go, the model specifically tailored to the Auslan dataset performs a larger amount of time series length reductions while staying above the minimal classification accuracy threshold and also features higher classification accuracy at each such step, when compared to the standard, out-of-the-box Deep Learning model estimated by the Aiello et al., (2016) framework.

The table 4.4 corroborates these findings. In terms of both classification performance and time series reductions achieved, the tailored Deep Learning method appears to be the best of the three classification algorithms considered here, followed by the standard Deep Learning model and, last but not least, the Frobenius 1-Nearest Neighbor search. Note, however, the run-times of each of the algorithms. In this aspect, one can observe a reversal of the previous ranking, with the standard Deep Learning method being an order of magnitude faster than the tailored Deep Learning model. The latter took at least around 9.6 minutes to complete a classification task,

which the former solved in about 11 seconds. Although this version of the 1-Nearest Neighbor search was by far the slowest, the truth is not as clear-cut as the results may seem to indicate. This poor performance relates to the concrete implementation of the 1-Nearest Neighbor search using the Frobenius norm. It was done directly in the programming language R, which can be notoriously slow for such tasks. Incidentally, this is the reason why package developers like Venables and Ripley, (2002) resort to implementing algorithms like the 1-Nearest Neighbor with the Euclidean distance in the C programming language (Kernighan and Ritchie, (2006)), which led to its high performance in the UCR datasets. A more efficient implementation of the Frobenius 1-Nearest Neighbor is employed in the Snackbox dataset, where it achieves the type of run-times one observed in the UCR dataset. Therefore, this thesis refrains from drawing further conclusions from the 1-Nearest Neighbor run-time in this case.

Note, also, that the tradeoff curves discussed here base on a single run of the early classification tradeoff curve derivation algorithm. No further iterations were necessary for this dataset, due to the overall stability of its results: It always resulted in the same relation of the algorithms to one other, similar accuracy levels and reduced time series lengths.

To sum the discussion of the results obtained for Auslan, the following conclusions are drawn. Early classification seems to especially practicable on this multivariate time series dataset, since classification accuracies of around 80% can be achieved with as little as 3.125% observations of an average Auslan sign (time series) length. Deep Learning seems to be here, again, a viable alternative to the baseline 1-Nearest Neighbor search with the Frobenius distance. Due to the tailored Deep Learning algorithm's comparatively long training and classifying times, the slightly less accurate, but significantly faster standard Deep Learning model seems to be an attractive alternative, since, on top of that, it also allows for relatively high accuracies of around 80% at about 6.25% of the full time series length.

Snackbox

The multivariate time series dataset "Snackbox", the smallest of the three analyzed in this thesis, is associated with the early classification tradeoff

curves featured in the plot numbered 4.5. In contrast to the Auslan dataset early classification tradeoff curves, the tradeoff curves shown in that figure contain classification accuracy averages, which were computed over a set of 50 iterations of the early classification algorithm of section 3.1.5, for a reason to be explained in detail further down in this section. In a, at least to a certain extent, similar fashion to the Auslan dataset, the early classification tradeoff curves exhibited here display improvements with time series length reduction up until a certain point is reached, then either stay relatively constant on that level or navigate around it, and finally conclude with a sharp drop in classification accuracy with time series lengths under 10%. Again, the initial counter-intuitive increase in classification accuracy can be traced back to a common tail of the Snackbox phases to be classified, which, when cut off, improves overall classification accuracy for both algorithms. The ups and downs for time series lengths in the interval 90% until 40% are direct byproducts of the structure of the Snackbox dataset instances, which consist of time windows of observations of the multivariate time series. Time windows in which some discriminating event occurs are crucial for both classifiers to separate phases from each other, and these are often interspersed by time windows where nothing of note occurs. Therefore, with the successive reductions in time series length, sometimes some of those more relevant time windows remain, but sometimes they are cut off and in the latter cases, classifiers make classification decisions on less clear-cut cases, thus resulting in a decline in the average classification accuracy. Finally, the sharp drop on classification accuracy ensues at around 3% for the 1-Nearest Neighbor search with the Frobenius matrix norm, and already at 19% of total time series length for the Deep Learning algorithm. On that note, in the Snackbox dataset, the roles seem to be reverted: The Deep Learning model is no longer, in contrast to before, either at least on par if not obviously better than the competing algorithms, but it seems to be significantly underperforming the 1-Nearest Neighbor with the Frobenius matrix norm. This dominance is apparent both as far as overall average accuracy, as well as the number of time series lengths reductions possible for the while minimal classification accuracy threshold of 60%, are concerned. One reason for this behavior could, of course, be parameter choice for the Deep Learning model. However, an extensive grid search was performed, so this thesis attributes the poor performance of the Deep Learning model in the Snackbox dataset mainly to the following two factors. The first of them

is the model size itself. Since a phase is composed of several windows of observations, with each observation itself consisting of 300 features, a phase fed as input to the deep learning model can have up to around 10700 input nodes (recall that this thesis' deep learning models for multivariate time series, described in 3.2.1, "flatten" the multivariate time series, represented as an $n \times m$ matrix, to a vector with nm entries). Depending on the size and number of the ensuing hidden layers in the deep neural network, this leads to millions of weights to be optimized. At this point, such a large model suffers from both a software and hardware constraints, since the deep learning framework H_2O cannot handle that many parameters to be optimized and, even if it could, hardware restrictions, particularly in RAM, become an issue. The second factor driving the performance of the Deep Learning model on Snackbox is the low dataset cardinality in terms of its available samples (and not each sample's dimensionality, the topic of the previous sentences). The Snackbox dataset consists of solely 60 phase instances, with each phase class having between 5 and 15 elements. These need to be divided up in a train and test set, thus reducing the sample to train the Deep Learning model with even further. Since Deep Learning models require, in general, a lot of data to really shine, this significantly hampers its results on the Snackbox dataset. The effect the lack of data available has can be observed in the configuration of the Deep Learning model. If one leaves out the "balance classes" parameter, which oversamples underrepresented phases to compensate for the lack of data on those, then average classification plummets, and it is often not higher than the minimal classification accuracy threshold, even at full time series length. Therefore, this leads to the belief that the low average classification accuracy of Deep Learning stems, at least to a certain extent, from the lack of data.

In fact, the Deep Learning model obtained here proved to be quite unstable, showing high volatility in classification accuracy. This volatility depended on the samples, which got randomly assigned to the train set in each iteration of the training process required for each of the reduced time series lengths. Hence, the early classification curve derivation algorithm was looped over 50 times, and each such iteration's results were averaged and then presented here, to ensure that the results are representative. The table 4.5, accompanying the figure 4.5, shows the detailed average classification accuracies and run-times for each of the reduced time series lengths. Although the

table reinforces the conclusions drawn previously, it is remarkable that the 1-Nearest Neighbor search with the Frobenius matrix norm consistently reaches average classification accuracies upwards of 80%, even for time series lengths as short as around 19% of the original length. As far as the run-time is concerned, the Deep Learning train and test times are bested by around an order of magnitude. Interesting to note here is also the fact that the run-time of the Deep Learning early classification seems to increase with decreasing time series length, which is counter-intuitive and was not the case in any of the datasets and iterations before. This is, as far as it could be investigated, a consequence of hardware constraints and the RAM gradually becoming full as previous results did not yet get garbage collected by the Java Virtual Machine.

The summary table 4.6, shows the number of iterations where the Frobenius 1-Nearest Neighbor search dominated Deep Learning in terms of the classification accuracy, per reduced time series length. It represents one final evidence piece that the former really is no match for the latter, as far as the early classification of this dataset is concerned. This is the case despite punctual cases like those of the time series lengths 90.25% and 81.45%, where, due to certain phase observations without high informational value being cut out and them impacting the volatile Deep Learning model "more strongly" than the Frobenius 1-Nearest Neighbor, the ratio of iterations where Deep Learning outperforms Frobenius 1-Nearest Neighbor briefly rises, before dropping again for good.

To conclude, for the Snackbox set of multivariate time series, early classification seems to be possible and interesting with the use of the 1-Nearest Neighbor search with the Frobenius matrix norm, for a reduction of time series length of up to 19% of the original length, is on average even beneficial to the average classification accuracy on this dataset. The application of Deep Learning methods seems to be inadequate for the early classification of this dataset of multivariate time series, due to both its small sample size as well as high cardinality inputs.

5.2 Limitations

Limitations pertaining to each of the algorithm applications to the early classification problem, such as the issue with the 1-Nearest Neighbor with ARIMA or linear model forecasts for non-stationary time series, or the underwhelming performance of the Deep Learning model in the Snackbox multivariate time series dataset with its low cardinality of train data coupled with high dimensionality, were discussed at length in the paragraphs above.

Therefore, the following addresses other, overarching limitations. Firstly, the Deep Learning models for the UCR datasets could be made tailored to each of the UCR time series datasets itself, instead of the current "one-size-fits-all" approach. While it still proves to be effective and accurate in early classification, due to its relatively large size in comparison to the UCR time series themselves, this could always be improved with a deeper grid search or perhaps more tailored feature selection to feed to the Deep Learning model, in particular by using certain windows or shapelets of the UCR time series as input. This approach is followed by Chen et al., (2015) in its computations of optimal dynamic time warping distances per dataset. Dynamic time warping delivers better performances than the Euclidean distance, due to its ability to factor in events in the time series relevant for its classification starting sooner (or later) in a given time window. Since usage of this tailored approach improves classification accuracy overall, the fact an overarching Deep Learning model was used here may hurt it in comparison.

Unfortunately, the results provided here cannot be directly compared to those obtained in the paper Juan J. Rodriguez and Carlos J. Alonso, (2002), due to its usage of an older version of the Auslan dataset, or to the nature of the empirical early classification results presented here per dataset strongly differing from Xing, Pei, and Philip, (2012) or Parrish et al., (2013). The reason for the lack of comparison possibilities with the latter lies in their focus being different than that of this thesis: Those papers present algorithms that smartly reduce time series length, while maintaining a series of properties desirable in a classification setting, such as stability or reliability of the classification decision. This thesis proposes an empirical approach for

assessing the applicability and performance of a given time series classification algorithm in an early classification context, focusing on the comparison of classification algorithms used for that task with each other, and without forcing and engineering them to maintain reliability and accuracy on par with that of classification using full time series length.

Finally, although this thesis employed a wide variety of time series datasets, including both univariate and multivariate time series data, cleaned and raw data, data used in (time series classification) literature and sensory data used for internal purposes, one can always broaden the number and variety of the datasets used in some aspect or the other.

5.3 Summary

In this chapter, the results obtained per dataset were discussed at length. In general, early classification and the derivation of early classification curves, defined as the tradeoff between accuracy and time series length, seems to be feasible for the three different time series datasets and the multiple algorithms considered here. Deep Learning seems to boast higher classification accuracy other algorithms, even if only by a slight margin, but only if used in a context, where certain conditions are met. In short, Deep Learning requires a lot of data and is not the best alternative if run-time is of the essence. Both of these aspects might realistically be relevant for the task of early classification of time series, since it arises, in practice, in contexts where not a lot of data is available (yet) and some classification decision needs to be made fast regardless. In such cases, the use of the 1-Nearest Neighbor can be recommended for use in the early time series classification task of both univariate as well as multivariate time series, due to its overall good performance on both accuracy vs. reduced time series length and low computational resources and run-time required for it. Enhancing reduced time series with classical time series forecasting methods, like linear models or ARIMA, does not seem to help in the early classification task. The results obtained here are limited by the configuration possibilities of the Deep Learning models themselves, which can be literally endless, lack comparison potential with literature on this topic due to it

following significantly different approaches than this thesis, and perhaps also by the choice of the time series datasets themselves, which can always encompass a wider variety of time series datasets and thus become more representative.

6 Conclusion

Basing time series classification decisions on reduced time series is a challenging problem. This thesis helps navigate through issues commonly arising when tackling that problem in a number of ways. Firstly, time series appropriate for early time series classification are identified. Then, an algorithmic recipe for deriving an accuracy vs. time series length trade-off curve is proposed. Finally, that algorithm was applied, in an experiment designed to understand which classification algorithms work best in this context. This chapter sums up the approach employed and results obtained in the search for answers to the research questions.

6.1 Review

A review of state-of-the-art literature on early time series classification revealed a research gap in the topic of early classification of multivariate time series, on the explicit derivation of an accuracy vs. time series length trade-off curve and on the application of deep learning algorithms to this problem. This thesis then set out to establish a theoretical framework to formalize time series mathematical representations and forecast models, classification algorithms, in particular deep learning, and the early time series classification trade-off curve derivation algorithm. Having established that theory and addressed its practical implementation, the results were computed with both univariate as well as multivariate time series datasets. All conclusions drawn on those results were interpreted in the context of the datasets the algorithms were applied to. In general, early classification was possible for all of them, with significantly reduced time series lengths still bringing fairly high accuracies, at least in comparison with the minimal classification accuracy threshold, a lower bound for classification accuracy

on reduced time series. The derived early classification trade-off curves reinforce this idea and allow for comparisons between the classification algorithms themselves used in this thesis. While Deep Learning showed, in general, a lot of potential, having better average performances for univariate time series and notably better performances with one of the multivariate time series datasets. However, it boasted rather low accuracies at even high time series lengths for the other multivariate time series dataset. The latter issues mainly revolved around the scarcity of the data available in that dataset, but Deep Learning also had, in comparison with other algorithms, significantly higher train and test run-times. The 1-Nearest Neighbor search consistently performed well throughout all datasets and was fast too, so it stood the tests this thesis put it through and empirically and comparatively established itself as a very viable algorithm in the context of early time series classification. Its variations with ARIMA and linear models were, on the other hand, quite poor, due to the UCR univariate time series being geared towards classification and not forecasting approaches, and thus not being stationary and not featuring common aspects, like trends or seasonality effects, these forecasting models thrive off of.

6.2 Future Work

This thesis leaves open a number of possible paths for further research work in this area. A few thoughts and ideas on this follow next.

As mentioned before, the classical time series forecasting methods ARIMA and linear models, used to forecast missing time series portions in early classification, assumed the type of characteristics the univariate datasets partially did not have, but other types of models could be applied here, like Kalman filters, which can be robust to non-stationary and, in general, more irregular time series. Entirely different models used in forecasting and even other methods to study time series may be also of interest in this context, such as shapelets or (fast) Fourier transforms, which could provide better indications which critical time series lengths influence the early classification trade-off curve the most.

Seeing as Deep Learning is such a complex approach, which needs a lot of data to perform at acceptable levels, a couple of ideas to improve it would be to engineer and tailor time series features to be fed to it, thus reducing input dimensionality, while capturing as much of the information contained in the reduced time series as possible. This may require specific domain knowledge specific to the time series dataset at hand, but, perhaps, other approaches like Xing, Pei, and Philip, (2012)'s or maybe even Mori et al., (2016)'s could be adapted to help Deep Learning models learn main class distinguishing features at reduced lengths.

One final idea, which might find application in real-world early classification problems might be online learning: Imbuing the general early classification approach presented here with the possibility to learn, adjust and enhance its performance on continuously incoming data seems a very promising idea.

6.3 Summary

This thesis set out to answer three research questions on early time series classification, and obtained some promising answers on those, as empirical results on which time series allow for early classification, trade-off curves therefore and algorithm comparisons were presented. However, constrained by the scope of this work, some avenues of future work on this problem, basing on the approaches presented here, were proposed and left for the reader to further investigate on, such as the testing of more advanced time series classification approaches or feature engineering for Deep Learning.

Appendix

Bibliography

Ahmed, Nesreen K et al. (2010). "An empirical comparison of machine learning models for time series forecasting." In: *Econometric Reviews* 29.5-6, pp. 594–621 (cit. on p. 11).

Aiello, Spencer et al. (2016). *h2o: R Interface for H2O*. R package version 3.8.2.6. URL: https://CRAN.R-project.org/package=h2o (cit. on pp. 36, 50, 61).

Amit Agarwal et al. (2014). *An Introduction to Computational Networks and the Computational Network Toolkit*. URL: http://www.cntk.ai/ (cit. on p. 11).

Banko, Zoltan and Janos Abonyi (2012). "Correlation based dynamic time warping of multivariate time series." In: *Expert Systems with Applications* 39.17, pp. 12814–12823 (cit. on p. 8).

Batres-Estrada, Bilberto (2015). "Deep learning for multivariate financial time series." In: (cit. on pp. 12, 24, 27).

Bengio, Yoshua (2009). "Learning deep architectures for AI." In: *Foundations and trends in Machine Learning* 2.1, pp. 1–127 (cit. on p. 10).

Bergstra, James et al. (2010). "Theano: a CPU and GPU Math Expression Compiler." In: *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Austin, TX (cit. on p. 10).

Brockwell, P.J. and R.A. Davis (2013). *Time Series: Theory and Methods*. Springer Series in Statistics. Springer New York (cit. on p. 5).

Busseti, Enzo, Ian Osband, and Scott Wong (2012). *Deep learning for time series modeling*. Tech. rep. Technical report, Stanford University (cit. on pp. 11, 27).

Candel A. et al. (2016). *Deep Learning with H2O*. URL: http://h2o.ai/resources (cit. on pp. 23, 24, 26, 36).

Chakraborty, Kanad et al. (1992). "Forecasting the behavior of multivariate time series using neural networks." In: *Neural networks* 5.6, pp. 961–970 (cit. on p. 11).

Chen, Yanping et al. (2015). *The UCR Time Series Classification Archive*. URL: www.cs.ucr.edu/~eamonn/time_series_data/ (cit. on pp. 6–9, 12, 13, 40, 44, 60, 66).

Dachraoui, Asma, Alexis Bondu, and Antoine Cornuejols (2015). "Early Classification of Time Series as a Non Myopic Sequential Decision Making Problem." In: *Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 433–447 (cit. on p. 12).

Deng, Li and Dong Yu (2014). "Deep learning: Methods and applications." In: *Foundations and Trends in Signal Processing* 7.3–4, pp. 197–387 (cit. on p. 10).

Eric Weisstein (2016a). *Distance*. URL: http://mathworld.wolfram.com/Distance.html (cit. on p. 22).

Eric Weisstein (2016b). *Frobenius Norm*. URL: Frobenius%20Norm. (cit. on p. 22).

Freund, Yoav, Robert Schapire, and N Abe (1999). "A short introduction to boosting." In: *Journal-Japanese Society For Artificial Intelligence* 14.771-780, p. 1612 (cit. on pp. 8, 13).

Ghalwash, Mohamed F, Dusan Ramljak, and Zoran Obradovic (2012). "Early classification of multivariate time series using a hybrid HMM/SVM model." In: *Bioinformatics and Biomedicine (BIBM), 2012 IEEE International Conference on*. IEEE, pp. 1–6 (cit. on p. 15).

Hatami, Nadereh and Camelia Chira (2013). "Classifiers with a reject option for early time-series classification." In: *Computational Intelligence and Ensemble Learning (CIEL), 2013 IEEE Symposium on*. IEEE, pp. 9–16 (cit. on p. 15).

Hills, Jon et al. (2014). "Classification of time series by shapelet transformation." In: *Data Mining and Knowledge Discovery* 28.4, pp. 851–881 (cit. on p. 8).

Hinton, Geoffrey et al. (2012). "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups." In: *Signal Processing Magazine, IEEE* 29.6, pp. 82–97 (cit. on p. 10).

Hu, Bing, Yanping Chen, and Eamonn J Keogh (2013). "Time Series Classification under More Realistic Assumptions." In: *SDM*. SIAM, pp. 578–586 (cit. on p. 41).

Hyndman, RJ (2016). *forecast: Forecasting functions for time series and linear models*. R package version 7.1. URL: https://CRAN.R-project.org/package=forecast (cit. on pp. 36, 59).

Hyndman, Rob J and George Athanasopoulos (2014). *Forecasting: principles and practice*. OTexts (cit. on pp. 19, 20).

Kadous, M. W. (2002). *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series*. URL: https://sites.google.com/site/waleedkadous/publications (cit. on pp. 6, 9, 12, 41).

Kadous, Mohammed Waleed (1999). "Learning Comprehensible Descriptions of Multivariate Time Series." In: *ICML*, pp. 454–463 (cit. on p. 8).

Keogh, Eamonn and Shruti Kasetty (2003). "On the need for time series data mining benchmarks: a survey and empirical demonstration." In: *Data Mining and knowledge discovery* 7.4, pp. 349–371 (cit. on pp. 9, 12).

Kernighan, Brian W and Dennis M Ritchie (2006). "The C programming language." In: (cit. on p. 62).

Know-Center (2016). *Know-Center: Austria's leading research center for data-driven business and big data analytics*. URL: http://www.know-center.tugraz.at/en/ (cit. on p. 42).

Laboratory, The Johns Hopkins University Applied Physics (2010). *Course Correction Keeps New Horizons (a mission by NASA) on Path to Pluto*. URL: http://pluto.jhuapl.edu/News-Center/News-Article.php?page=20100701 (cit. on p. 2).

Langkvist, Martin, Lars Karlsson, and Amy Loutfi (2014). "A review of unsupervised feature learning and deep learning for time-series modeling." In: *Pattern Recognition Letters* 42, pp. 11–24 (cit. on p. 11).

LeCun, Yann et al. (1998). "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (cit. on p. 10).

Lines, Jason et al. (2012). "A shapelet transform for time series classification." In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 289–297 (cit. on p. 8).

LISA lab (2016). *DeepLearning 0.1 documentation*. URL: http://deeplearning.net/tutorial/gettingstarted.html (cit. on p. 26).

Martin Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: http://tensorflow.org/ (cit. on p. 11).

Mnih, Andriy and Geoffrey E Hinton (2009). "A scalable hierarchical distributed language model." In: *Advances in neural information processing systems*, pp. 1081–1088 (cit. on p. 34).

Mori, Usue et al. (2016). "Reliable early classification of time series based on discriminating the classes over time." In: *Data Mining and Knowledge Discovery*, pp. 1–31 (cit. on pp. 14, 15, 30, 32, 40, 71).

Nau, Robert F. (2016a). *Introduction to ARIMA: nonseasonal models*. URL: http://people.duke.edu/~rnau/411arim.htm (cit. on p. 19).

Nau, Robert F. (2016b). *Introduction to ARIMA: nonseasonal models*. URL: http://people.duke.edu/~rnau/arimrule.htm (cit. on p. 20).

NIST/SEMATECH (2013). *e-Handbook of Statistical Methods - Introduction to Time Series Analysis*. URL: http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc41.htm (cit. on p. 5).

Parrish, Nathan et al. (2013). "Classifying with confidence from incomplete information." In: *The Journal of Machine Learning Research* 14.1, pp. 3561–3589 (cit. on pp. 12, 14, 30, 40, 66).

Petitjean, Francois et al. (2014). "Dynamic time warping averaging of time series allows faster and more accurate classification." In: *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, pp. 470–479 (cit. on p. 12).

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/ (cit. on p. 36).

Rodriguez, Juan J. and Carlos J. Alonso (2002). "Boosting interval-based literals: Variable length and early classification." In: *Knowledge Discovery from Temporal and Spatial Data* (cit. on p. 66).

Rodriguez, Juan J, Juan J Rodr Guez, and Carlos J Alonso (2002). "Boosting interval-based literals: Variable length and early classification." In: (cit. on pp. 8, 12, 13, 42).

Rodriguez, Juan Jose, Carlos J Alonso, and Jose A Maestro (2005). "Support vector machines of interval-based features for time series classification." In: *Knowledge-Based Systems* 18.4, pp. 171–178 (cit. on p. 8).

Ronan Collobert et al. (2016). *torch - A scientific computing framework for LuaJIT*. URL: www.torch.ch (cit. on p. 10).

Schmidhuber, Jurgen (2015). "Deep learning in neural networks: An overview." In: *Neural Networks* 61, pp. 85–117 (cit. on p. 9).

Shumway, Robert H and David S Stoffer (2013). *Time series analysis and its applications*. Springer Science & Business Media (cit. on pp. 18–20).

Silver, David et al. (2016). "Mastering the game of Go with deep neural networks and tree search." In: *Nature* 529.7587, pp. 484–489 (cit. on p. 10).

Skiena, Steven S (1998). *The algorithm design manual: Text*. Vol. 1. Springer Science & Business Media, pp. 361–363 (cit. on p. 6).

Square, Inc. (2016). *An HTTP & HTTP/2 client for Android and Java applications*. URL: http://square.github.io/okhttp/ (cit. on p. 36).

Taigman, Yaniv et al. (2014). "Deepface: Closing the gap to human-level performance in face verification." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708 (cit. on p. 10).

TinkerForge (2016). *TinkerForge building blocks*. URL: http://www.tinkerforge.com/en/home/what_is_tinkerforge/ (cit. on p. 42).

Venables, W. N. and B. D. Ripley (2002). *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer. URL: http://www.stats.ox.ac.uk/pub/MASS4 (cit. on pp. 36, 62).

Wei, Li and Eamonn Keogh (2006). "Semi-supervised time series classification." In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 748–753 (cit. on p. 7).

Xing, Zhengzheng, Jian Pei, Guozhu Dong, et al. (2008). "Mining Sequence Classifiers for Early Prediction." In: *SDM*. SIAM, pp. 644–655 (cit. on p. 13).

Xing, Zhengzheng, Jian Pei, and Eamonn Keogh (2010). "A brief survey on sequence classification." In: *ACM SIGKDD Explorations Newsletter* 12.1, pp. 40–48 (cit. on p. 42).

Xing, Zhengzheng, Jian Pei, and S Yu Philip (2009). "Early Prediction on Time Series: A Nearest Neighbor Approach." In: *IJCAI*. Citeseer, pp. 1297–1302 (cit. on pp. 12–14, 21, 28, 30).

Xing, Zhengzheng, Jian Pei, and S Yu Philip (2012). "Early classification on time series." In: *Knowledge and information systems* 31.1, pp. 105–127 (cit. on pp. 13, 40, 66, 71).

Xing, Zhengzheng, Jian Pei, S Yu Philip, and Ke Wang (2011). "Extracting Interpretable Features for Early Classification on Time Series." In: *SDM*. Vol. 11. SIAM, pp. 247–258 (cit. on pp. 13, 14).

Yang, Kiyoung and Cyrus Shahabi (2007). "An efficient k nearest neighbor search for multivariate time series." In: *Information and Computation* 205.1, pp. 65–98 (cit. on pp. 6, 7).

Ye, Lexiang and Eamonn Keogh (2009). "Time series shapelets: a new primitive for data mining." In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 947–956 (cit. on p. 8).