FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of master's thesis

Title:	AutoML for anomaly detection in a semi or unsupervised setting on time series
Student:	Bc. Marek Nevole
Supervisor:	MSc. Jan Bím, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2023/2024

Instructions

In many industries, there has been a very large increase in volumes of sensoric data. Often this kind of data comes unlabeled and obtaining labels is usually very costly and or time consuming, especially in case of anomalies for the task of anomaly detection. Additionally, due to high demand for solving the problem of anomaly detection in many particular businesses, AutoML (Automated Machine Learning) techniques are used to build such a system very frequently. However, it is very difficult in a setting where the system has no or very little ground truth information. Therefore, the student should investigate the field of anomaly detection in time series [1] and its evaluation methods and suggest a viable strategy for application of AutoML to this kind of task [2]. Since choice of AutoML techniques and composition of the whole pipeline is highly dependent on the available computational resources, we set the scope of the thesis to an application within computational budgets commonly available to a small/medium sized company today. Additionally, this thesis is limited to univariate time series.

Instructions:
1) Conduct a survey of the state of the art in anomaly detection on time series data and its evaluation in a setting with no and/or few labels and use of AutoML in anomaly detection on time series.
2) Compose at least 2 AutoML pipelines solving anomaly detection on time series in unsupervised or nearly unsupervised setting based on existing methods found in 1). Optionally suggest a new method that can replace a part (e.g. new evaluation metric,

---

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 27 December 2022 in Prague.

new anomaly detector etc.) or more parts of the AutoML pipeline, based on what will be expected to bring largest benefit, and implement it, creating a third pipeline.
3) Compare these pipelines on at least 10 datasets suggested in [3] and at least one common dataset (e.g. [4]) used in evaluation of AD on time series to achieve general comparability despite it being marked flawed in [3].
4) Analyze and discuss the results in detail.

[1] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: a comprehensive evaluation," Proc. VLDB Endow., vol. 15, no. 9, pp. 1779–1797, Jul. 2022, doi: 10.14778/3538598.3538602.
[2] M. Bahri, F. Salutari, A. Putina, and M. Sozio, "AutoML: state of the art with a focus on anomaly detection, challenges, and research directions," Int J Data Sci Anal, vol. 14, no. 2, pp. 113–126, Aug. 2022, doi: 10.1007/s41060-022-00309-0.
[3] R. Wu and E. Keogh, "Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress," IEEE Trans. Knowl. Data Eng., pp. 1–1, 2021, doi: 10.1109/TKDE.2021.3112126.
[4] N. Laptev, S. Amizadeh and Y. Billawala, "S5 - A Labeled Anomaly Detection Dataset, version 1.0 (16M)," Mar. 2015; https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

AutoML for anomaly detection in a semi or unsupervised setting on time series

Bc. Marek Nevole

Department of Applied Mathematics
Supervisor: MSc. Jan Bím, Ph.D.

May 4, 2023

Acknowledgements

My gratitude goes to my supervisor MSc. Jan Bím, Ph.D. The completion of this thesis would not have been possible without his guidance, constructive feedback, and endless encouragement. Furthermore, I am also grateful to those who in any possible way contributed to the finalization of this thesis.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.
I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 4, 2023 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Nevole, Marek. *AutoML for anomaly detection in a semi or unsupervised setting on time series.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Abstrakt

Úspěšné použití metod automatizovaného strojového učení (AutoML) pro detekci anomálií v časových řadách, v případě, kdy není k dispozici téměř žádná nebo žádná informace vyjadřující anomalitu dat, je náročný problém. Tato práce poskytuje přehled nejnovějších přístupů v oblasti detekce anomálií, AutoML a vyhodnocení modelů pro detekci anomálií. Provedené experimenty se zaměřují na sestavení nových AutoML kombinací z dostupných metod pro detekci anomálií v jednorozměrných časových řadách při částečně supervizovaném a nesupervizovaném učení. Hlavní náplní experimentů bylo vyhodnocení metrik nesupervizovaného učení pro optimalizaci hyperparametrů a meta-learning přístup pro výběr modelů. Výsledky experimentů této práce nabízí nové poznatky k současným metodám a otevírájí směry pro budoucí výzkum.

Klíčová slova Detekce anomálií, Časové řady, Automatizované strojové učení, Nesupervizované učení

Abstract

Successfully deploying automated machine learning (AutoML) for anomaly detection in time series data where little or no ground truth information is available is a challenging problem that is ever more important. This thesis provides an overview of state-of-the-art approaches in the fields of anomaly detection, AutoML, and evaluation of anomaly detection models. The conducted experiments focus on composing new AutoML pipelines from available methods for anomaly detection in univariate time series data in semisupervised and unsupervised settings. The main focus of the experiments was an evaluation of unsupervised metrics for hyperparameter optimization and a meta-learning approach for model selection. The results of this thesis offer new insight into the methods available and several directions for future work.

Keywords Anomaly detection, Time series, Automated machine learning, Unsupervised learning

Contents

List of Figures

Introduction

We as humanity have been subconsciously collecting sequential data ever since our origin. Our predecessors collected and recorded information about the world around them. From monitoring the changing seasons for agricultural yields to tracking the movements of animals, early humans relied on this data to survive. In the medieval ages, records were used for a variety of purposes, such as taxation, trade, and governance. Merchants would keep track of their sales and purchases to manage their inventory and profits. This information was often recorded in books that contained rows and columns of data.

As of today, we have developed more sophisticated approaches to collecting data that enable us to gather more complex and detailed information. Today, we use advanced technologies and techniques to collect and analyze massive amounts of data.

However, as the amount of data collected grows, it becomes increasingly difficult to identify and interpret important patterns and trends. As a result, the need for anomaly detection is more important than ever. Anomaly detection is the process of identifying data points or patterns that deviate significantly from the norm. Detecting anomalies can reveal hidden insights, prevent potential disasters, and improve decision-making processes.

In many industries, there has been a significant increase in the volume of sensor data. Often this kind of data comes unlabeled, and obtaining labels is usually very costly and/or time-consuming, especially in case of anomalies for the task of anomaly detection. The task of anomaly detection has been researched in many different fields, causing many new methodologies and techniques to be developed in that specific field of study. To allow the usage of methods across the field without the need of going deep into the inner workings of the algorithms a process of automated machine learning is on the rise. Automated Machine Learning (AutoML) solutions have been on the rise in recent years,

as they allow a broader spectrum of AI practitioners to train machine learning models that perform well in practice with ease and with very little domain knowledge, which tends to be a limiting factor and is expensive to learn.

The increasing need for the collection of data in recent years has led to an increase in the number of smaller Internet of Things (IoT) devices connected to networks. As a result of development and research, these devices are becoming more accessible to a wider range of users. However, with the rise of machine learning and, specifically, deep learning, models are becoming larger and more expensive to train and infer with. Therefore, there is a growing need for AutoML pipelines that can deliver both high performance and efficiency simultaneously.

However, it is challenging to deploy AutoML in a setting where the system has no or very little ground truth information available. In the current situation, there is no known go-to approach for the problem of unsupervised automated machine learning for anomaly detection in time series data.

In this thesis, a survey across fields of automated machine learning, outlier detection, and anomaly detection in time series is presented is presented. Specifically, emphasis has been placed on hyperparameter optimization and model selection to create automated machine learning pipelines in unsupervised/semisupervised settings, where little or no labels of anomalies are provided. We also limit ourselves to univariate time series where data only come from one sensor and time series consists only of one channel. As many practical use cases use univariate data. Limiting makes the task much more manageable and easier to interpret at no cost to the value brought into the field.

Structure

In the first chapter, the problem of outlier detection and its equivalent in the time series domain called anomaly detection is presented. The second chapter is dedicated to surveying the state-of-the-art models of anomaly detection. As the evaluation of the anomaly detection models is still ongoing research, the entirety of chapter three is given to it. In chapter four, the process of combined algorithm selection and hyperparameter optimization is presented as well as other state-of-the-art techniques of AutoML. In chapter five, an outline and summary of research questions and expected experimental outcomes are provided. Chapter Six is dedicated to experiments and a discussion of the results obtained. Lastly, the content and main insights of this thesis are concluded in the conclusion, and several research directions are presented.

Contributions

The main contributions of the work to the field of anomaly detection and mainly unsupervised anomaly detection are believed by us:

- A survey of three distinct subfields of anomaly detection has been made, namely models of anomaly detection, evaluation, and automated machine learning.
- An open-source Python code is provided as a package of various models and utilities needed for anomaly detection. [1]
- Two metrics for unsupervised outlier detection have been thoroughly tested.
- Experiments were conducted using a meta-learning approach for outlier detection model selection to determine if the observed performance translates to the time series domain. In addition, potential enhancements were explored.

[1] https://github.com/NevoleMarek/automltsad

Chapter 1

Anomaly detection in time series

In this chapter, we introduce the fundamental concepts and definitions related to anomaly detection in time series, along with the notation used throughout this work. The reader is assumed to have a basic understanding of machine learning, statistics, and mathematics relevant to this field. For those seeking further knowledge on the topic, Bishop's "Pattern Recognition and Machine Learning" [1] and Hastie et al.'s "The Elements of Statistical Learning" [2] are recommended resources."

As increasingly more data are collected, especially from sensors being implemented into new devices more frequently than ever before, there arises the problem of accurate anomaly detection of these sensors/time series data. It is predicted that the number of Internet of Things (IoT) connected devices will more than double in the next decade [3]. These devices regularly collect sensor data.

Anomaly detection is being used in various fields. In manufacturing to find defective products, in biology to identify rare viruses, among others, in medicine to diagnose diseases or to monitor patients and call for help if needed, in finance to detect fraud, in cybersecurity to look for any system breaches and suspicious activity, etc.

The latest popular application of anomaly detection in time series on a global level is in smart watches and phones, where so-called crash detection is being implemented to read sensor data to call for help whenever the users end up in danger (e.g., car crash, fall).

1.1 Anomaly

Since the span of anomaly detection across many fields, there arose a variety of definitions of anomalies also called outliers or novelties in the literature. Braei and Wagner in [4] went through some of the definitions chronologically as they appeared in the works. They found that one of the first definitions of anomalies was proposed by Frank E. Grubbs [5] in 1969. The definition is as follows. *"An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs."*

Later in 1980, Hawkins [6] defined them as *"An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism."* What all of the definitions highlighted were two main characteristics of anomalies:

- The distribution of the anomalies deviates considerably from the general distribution of the data.
- The big majority of the dataset consists of normal data points. The anomalies form only a very small part of the dataset.

As the final definition in [4] following has been used *An anomaly is an observation or a sequence of observations which deviates remarkably from the general distribution of data. The set of anomalies forms a very small part of the dataset.*

The previous definitions are mainly concerned with anomalies in some arbitrary vector space. Not taking time series into account. In this work, a new definition of anomalies in time series has been combined from [4, 7] and is defined as follows:

Definition 1.1 *(Anomaly/outlier/novelty) An anomaly in time series is a point (outlier) or point sequence (irregularity) that deviates w.r.t. a measure, model, or embedding from the regular patterns of the sequence. The set of anomalies forms a very small part of the dataset.*

The emphasis on having a very small part of the dataset being anomalies is also related to the risk of losing the meaning of what it means to be anomalous. When anomalies are too frequent, they may no longer be considered anomalous and instead become part of the normal patterns in the data.

1.2 Time series

Before defining time series and taxonomy of anomalies, a notation for scalars, vectors, sequences, and matrices is given as follows. Scalars are denoted as

small letters, vectors as small bold letters, sequences as capital letters, and matrices as capital bold letters. (e.g. x, $\boldsymbol{x}$, X, $\boldsymbol{X}$).

Definition 1.2 *(Univariate time series) A univariate time series is a sequence $X = (x_t)_{t \in T}$ of observations $x_t \in \mathbb{R}$ taken at specific time $t \in T \subseteq \mathbb{N}$.*

Definition 1.3 *(Multivariate time series) A multivariate time series is a sequence $X = (\boldsymbol{x}_t)_{t \in T}$ of observations $\boldsymbol{x}_t \in \mathbb{R}^d$, where d is a dimension of observation or a number of channels of the time series, taken at specific time $t \in T \subseteq \mathbb{N}$.*

Although not specified majority of time series consist of equidistant time indices. For later use definition of time series subsequences is provided here as well.

Definition 1.4 *(Subsequence) Let X be a time series, a subsequence of length $n \leq |T|$ of time series X starting at i-th time index is $S_i = (\boldsymbol{x}_i, \ldots, \boldsymbol{x}_{i+n-1})$, for i, n so that $i + n - 1 < |T|$.*

Subsequences can be thought of as applying a sliding window over the time series with a stride of one. This is particularly useful as many of the algorithms carry over from outlier detection on tabular datasets or the algorithms just work with local substructures of time series.

1.3 Types of anomalies

Creating a taxonomy of types of anomalies is a subject of interest in anomaly detection research, and there are relatively few works on this topic [4, 8, 9, 10]. A taxonomy of anomalies can help in defining and categorizing different types of anomalies, which can lead to the development of more effective anomaly detection algorithms. The works [4, 8, 9, 10] seem to come to the conclusion that anomalies are mostly divided into three types as follows:

- **Point anomaly** - is a data point or a sequence in the time series that deviates from the normal range of values in an extreme way.
- **Contextual anomaly** - "is the individual instance that is anomalous in a specific context, such as the discord points within the same harmonic pattern. Contextual outliers usually have relatively larger/smaller values in their own context but not globally. Identifying contextual outliers is often considered more challenging." [8]
- **Collective anomaly** - "is a type of anomaly that refers to a set of data points that should be considered an anomaly because they gradually show a different pattern from normal data over time. Individual values

> within this type of anomaly may seem trouble-free, but collectively, they raise suspicion." [9]

Less common types of anomaly categories found in the literature are entire outlier time series, which can be found when the time series is multivariate, or subcategories that further divide collective anomalies (e.g. trend anomalies where a trend is suddenly introduced to data).

Despite the fact that the type of anomaly is important for the development of new methods of anomaly detection, they are not essential to know for the field of automated anomaly detection in an unsupervised or semisupervised setting where no or little to no ground truth is provided. Because of the lack of information on anomalies, the taxonomy cannot be leveraged. The inclusion of taxonomy aims to provide the reader with a deeper understanding of the field.

Chapter 2

Models

This chapter provides the reader with a survey of anomaly detection models from basic statistical approaches to deep learning models. These methods are categorized according to common taxonomy used in the field. The methods presented here are not explained in great depth as the focus of the thesis is the automated selection of these methods rather than the methods themselves. However, all needed references are given.

2.1 Anomaly detector

The task of anomaly detection can be thought of as a binary classification task for predicting whether data points are anomalous or not. But in reality, it is much more valuable to obtain information on how anomalous the data points are. Hence why a scoring function s is fitted and the output of models is an anomaly score $s(x_i) \in \mathbb{R}$ for each data point, usually in the range of real values, with the presumption that the higher the anomaly score the more anomalous is the point. To go from this regression task to binary classification, a threshold $t \in \mathbb{R}$ is selected in the fitted scoring function and each data point with an anomaly score greater than $s(x) \geq t$ is found anomalous.

The process of selecting the threshold is different for each model and usually requires some prior information about anomalies. One of the prior information is the contamination rate c of the dataset. The contamination rate simply indicates the rate of anomalies present in the data. One of the possible ways to set the threshold t is as a $1 - c$th quantile in the range of fitted scoring function s on the training samples such that $P(s(x) \geq t) = c$, where P is an empirical probability of anomaly score of the fitted function s on training samples.

2.2 Taxonomy of models

Recent survey papers [4, 7, 8, 9, 10, 11] seem to find common taxonomies as there are practically no discrepancies. For this thesis, the same taxonomy as in the survey from Schmidl et al. [7] on anomaly detection in time series is used, as this work's taxonomy is the most complete out of the others. Models are categorized into 6 distinct groups: Forecasting, reconstruction, encoding, distance, distribution, and isolation tree methods.

2.3 Forecasting methods

Native to the time series domain, forecasting methods play a major role in anomaly detection in time series. These models are trained on historical contextual windows to be as accurate as possible in predicting future values. To obtain an anomaly score, the future value is predicted and compared to the observed one, the anomaly score is then computed as a deviation of these values from each other. Most models use a window of a specific length with a stride of one as a context for predicting usually one future value [7].

One of the most recognized models in the time series forecasting domain are ARIMA [12] models. The abbreviation ARIMA stands for AutoRegressive Integrated Moving Average and can be split into three basic concepts. The AutoRegressive (AR) model models the assumption that future values depend linearly on previous values. The future value x_t is predicted as

$$x_t = \epsilon_t + \varphi_1 \cdot x_{t-1} + \cdots + \varphi_p \cdot x_{t-p}$$

where ϵ_t is white noise, φ_i are learnable parameters, and p denotes the order of the AR model. I stands for integrated and indicates the order of differencing done to the input data before modeling using AR and MA models. Model with $I = 1$ denotes that the input is transformed as $x'_t = x_t - x_{t-1}$. Differencing addresses trend and/or seasonality in time series that is usually preferred to be removed for models to perform better. Lastly, MA predicts future values using white noise error terms as

$$x_t = c + \epsilon_t + \theta_1 \cdot \epsilon_{t-1} + \cdots + \theta_q \cdot \epsilon_{t-q}$$

where c is a $\mathbb{E}(x_t)$, and q is an order of MA. Time series tend to drift in time, i.e. change mean and variance, hence why parameters of ARIMA models tend to be learned with every new observation [7]. This is a considerable disadvantage and makes these methods exceptionally slow.

Decision trees have been used successfully in many classification and regression tasks on tabular data [13]. These traditional and popular machine learning models such as Random forest [14] or Gradient boosted trees [15] can also be

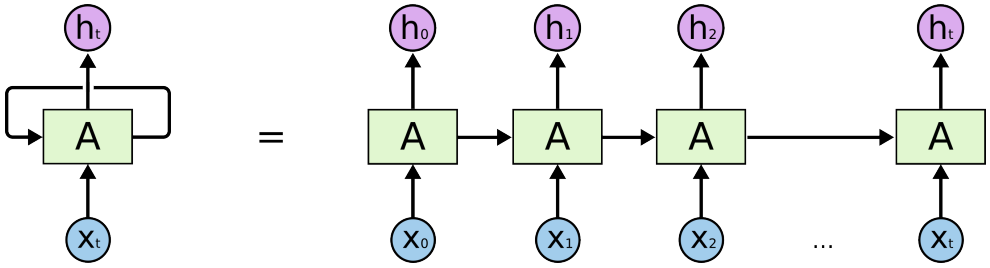


Figure 2.1: (Left) Simple visualization of RNN with input x and output h. (Right) RNNs are commonly visualized as unrolled. Image is from [16].

exploited for time series forecasting. By transforming a dataset via windowing into subsequences, these models can be learned to predict future values from the previous context. However, they are not commonly used and can only be used for univariate time series.

Deep learning is extremely popular and, without a doubt, the most active research area in the field of machine learning. One of the first architectural classes for modeling sequential data, such as time series or language, was recurrent neural networks (RNNs). These networks allow for neural connections to create cycles and the cycles in return allow the information to pass to the subsequent input.

Early RNNs had a problem of vanishing gradient [17], where gradient information is lost (gradients are very small) during backpropagation and weight values are not updated at all. This was a cause for RNNs not being able to reliably learn long-term dependencies. As a reaction to this problem, the same author came up with Long short-term memory (LSTM) [18] neural network architecture. LSTM is a network made up of 4 layers. Them being cell, forget gate f, input gate i, and output gate o. The entire LSTM neural network is usually described in equations of components. Equations for the original LSTM are as follows:

$$
\begin{aligned}
f_t &= \sigma_g(\boldsymbol{W}_f x_t + \boldsymbol{U}_f h_{t-1} + b_f) \\
i_t &= \sigma_g(\boldsymbol{W}_i x_t + \boldsymbol{U}_i h_{t-1} + b_i) \\
o_t &= \sigma_g(\boldsymbol{W}_o x_t + \boldsymbol{U}_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(\boldsymbol{W}_c x_t + \boldsymbol{U}_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \sigma_c(c_t)
\end{aligned}
$$

where σ_g is a sigmoid function, σ_c is a hyperbolic tangent function, t a time index, x_t a input, c_t a cell state, h_t a hidden state, matrices $\boldsymbol{W}_{\{f,i,o,c\}}$ and

$\boldsymbol{U}_{\{f,i,o,c\}}$ weights of the network. The forget gate f_t tells what information can be forgotten in the new cell state c_t, the input gate i_t adds new information to the cell state based on the hidden state h_{t-1} and input state x_t, and the output gate o_t controls what part of new cell state is outputted based on the hidden state h_{t-1} and input state x_t. [18]

The disadvantage of LSTMs lies in the lack of parallelization with their sequential nature. They take longer to train, consume more memory, and are easier to overfit. [19]

Ever since the paper introducing transformer architecture [19] has been published, transformer architecture has been under the spotlight. With recent advancements especially in natural language processing tasks with large language models (LLMs), transformers are state-of-the-art for sequential data modeling. Unlike RNNs, transformers process the entire input at once, allowing for better efficiency and parallelization. The heart of transformer architecture is the concept of attention. Attention is defined by the following formula:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V}$$

where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are matrices computed by matrix product of input embedding $\boldsymbol{X}$ and learnable matrices $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V$, and d_k dimension of $\boldsymbol{K}$ and $\boldsymbol{Q}$. $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ stands for query, keys, and values and serves only as an abstraction of intuition of inner workings. The scaling factor d_k is present to keep the variance at 1 for stability, as the input is usually standardized. The attention mechanism tries to capture what is important in the input sequence for the prediction of the output using the softmax function that assigns to each input term a value from $(0, 1)$ being the importance of the term. The transformer architecture itself does not know the ordering of the sequence passed in. To give a transformer a piece of information about the ordering of the sequence, positional encoding [19] is added to the embedded input. Transformer architecture can be seen in figure 2.2.

Another direction in deep learning is Graph neural networks (GNNs) [20]. GNNs have been recently getting more attention in the anomaly detection field as new methods use them to model multivariate time series channels as nodes and capture relationships between them by assigning edges to the graph. GDN [21] and GTA [22] are the methods that perform well among the state-of-the-art methods using GNN architecture [9]. GTA uses a graph convolution network (GCN) [20] to share information between similar channels and then deploys Informer [23] the transformer used in the prediction of long-sequence time series, which is considered a state-of-the-art.

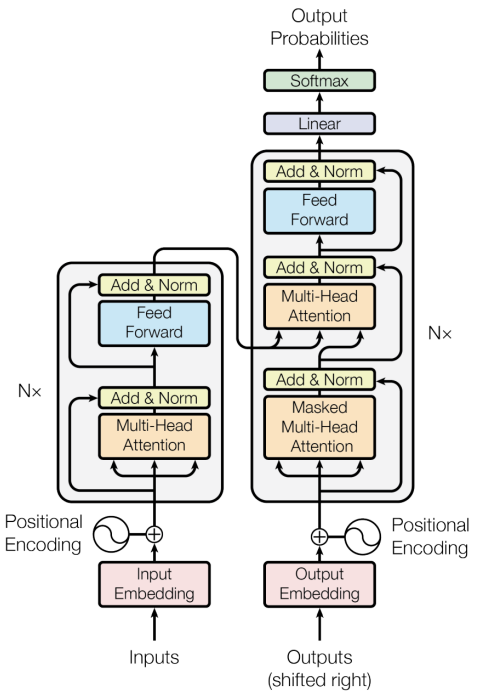


Figure 2.2: Transformer architecture [19].

If possible, these models are preferred to be trained in a semisupervised setting. Where training is done only with nonanomalous data. This would in theory allow for easier recognition of anomalous behavior when deployed as the model has never encountered such an anomalous pattern and would probably mispredict the future values more.

2.4 Reconstruction methods

Reconstruction methods in anomaly detection work by encoding the input, commonly subsequences of X, to a lower dimension latent space and then trying to reconstruct the input by decoding from learned latent space. The error between the original and reconstructed input is used as an anomaly score. Models are expected to not reconstruct anomalous sequences well as their presence in the training dataset is minimal or none [7].

This class of methods is dominated by deep learning models. AutoEncoder (AE) [24] and its variations are the basis of reconstruction models. AutoEncoders usually use the notation of two parametrized function families E_ϕ and D_θ, where ϕ and θ are parameters. Encoding function family $E_\phi : \mathcal{X} \to \mathcal{Z}$ and decoding function family $D_\phi : \mathcal{Z} \to \mathcal{X}$, where $\mathcal{X} \in \mathbb{R}^d$ is original space and $\mathcal{Z} \in \mathbb{R}^k$ a learned latent space with $k < d$, commonly $k \ll d$. The networks

are then built to fulfill these requirements it is common that the decoder is just a mirrored encoder network. To measure the error a generalized metric L_p is used $L_p(x, x') = \sqrt[p]{\sum_{i=1}^{d}(x_i - x'_i)^p}$, where x is the original and x' is the reconstructed sequence and p is usually equal to 1 or 2.

In 2013 Kingma and Welling introduced Variational AutoEncoder (VAE) as part of their paper on the stochastic gradient variational Bayes estimator [25]. Although similar to AE in name, these two are very different approaches to achieving the same goal. VAE models the latent space $\mathcal{Z}$ by a probability distribution Q. In almost all cases, the Gaussian distribution is used [26, 27]. Encoder is then trained to predict a mean and a variance of the distribution. When the mean and variance are predicted in the forward pass, a random sample is drawn from a unit Gaussian distribution. Using the reparametrization trick [28], this sample is transformed with the mean and variance and then decoded in the decoder part of the network. The reparametrization trick is often said to be used because backpropagation can not go through a random sampling node. Technically this is not true, theoretically, there is no guarantee that the estimate of the gradient will converge. The reparametrization trick allows for the correct estimation of derivatives during backpropagation [28]. The loss that is minimized in optimization consists of two terms, Mean squared error (L_2) summed with Kullback-Leibler divergence (KL) between unit Gaussian $\mathcal{N}(0, I)$ and the latent distribution. KL is a measure measuring the distance of distribution Q from probability distribution P. For two continuous random variables, the Kullback-Leibler divergence is defined as:

$$D_{KL}(P\|Q) = \int_{-\infty}^{+\infty} p(x) \log(\frac{p(x)}{q(x)})dx$$

which can be interpreted as an expectation of logarithmic difference between P and Q probabilities under probabilities of P. In the case of $p(x) = 0$ the contribution of the term $0 \log \frac{0}{q(x)}$ is zero for all possible values of $q(x)$. When $p(x) > 0$ and $q(x) = 0$, KL is defined as $+\infty$, $D_{KL}(P\|Q) = +\infty$. The interpretation of this loss is that MSE forces the model to learn how to accurately reconstruct input and KL forces latent space to match unit Gaussian as closely as possible. A much more rigorous explanation is provided in [25] and [26] is a great resource for initial understanding. The effect of KL divergence can be seen in figure 6.7.

Other types of neural networks used for reconstruction in anomaly detection are transformers, TranAD [29] is among the state-of-the-art with its results. This transformer uses adversarial two-term loss where one term focuses on pure reconstruction loss and the latter on accurately predicting errors made by the first decoder. This approach is called self-conditioning and leads to smaller errors. LSTMs can be used for reconstruction as well [30]. EncDecAD

method uses two LSTM networks, one as an encoder and one as a decoder, where the encoder learns the hidden state and passes it as input to the decoder that tries to reconstruct the input sequence in a backward fashion.

2.5 Encoding methods

Encoding methods are similar to reconstruction methods as they try to encode input sequences as well. Assumption close to the reconstruction models is assumed. Anomalous sequences will not be encoded as well as normal sequences. An anomaly score is then obtained by comparing sequences in learned latent space. Encoding methods mostly use statistical approaches [7]. However, the encoder part of a trained AutoEncoder can be used for this purpose as well. In [7], GrammarViz [31] and Series2Graph [32] were two of the encoding methods that performed well among the others.

GrammarViz works by performing discretization on subsequences of time series, turning each subsequence into a word from a finite alphabet using the SAX algorithm [33]. SAX divides a subsequence into further n equal width sub-subsequences and computes the mean in each sub-subsequence. With the means computed one can map these sub-subsequences to an alphabet by a predefined set of breakpoints in the range of mean values. The next step is to generate context-free grammar from the string of SAX-generated words. To compute an anomaly score occurrence of each word in all grammar rules is counted. The words with the least occurrence count are rarely used and could be considered anomalous and the original sequence of time points from which these words came is marked as anomalous.

Series2Graph converts time series into a graph by firstly embedding the subsequences into three-dimensional space using Principal component analysis [34] then rotating the vector space so that two components are close to each other and the last one is further away so that two of the components contain the shape related characteristic, and the last one the average value. By this rotation, a projection is created into a space where subsequences with similar shapes are clustered together. These two components are taken as a two-dimensional embedding. In this two-dimensional vector space, nodes are created in places of higher density of embeddings. Edges for the nodes are assigned for successive nodes in the sequences of time series, and the weight of the edge corresponds to the number of times two nodes are in succession. This allows for distinguishing between normal and anomalous samples by looking at the edge weights. Where paths between two nodes with lower-weighted edges are occurring less. [32]

2.6 Distance methods

Distance methods use various distance metrics and measures to compare between points or subsequences of time series. The assumption here is that anomalous samples are more distant from other samples. Nearest neighbor graph methods rank as one of the most popular in anomaly detection [4, 7, 8, 11, 35]. The simplicity and great intuition allowed many variations of approaches to be created. All the methods rely on some metric in a metric space. The Euclidean metric is usually used. Euclidean distance d between two points a, b in n-dimensional space is then computed as $d(a, b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$. With a defined metric the simplest anomaly detection scoring function can be an average of distances of a test sample to k nearest samples in a training split set. This approach is an unsupervised nearest neighbor model also known as k-NN in the field of anomaly detection. In theory, this works best when no anomalies are present in the training dataset. With an increasing contamination rate, the performance is expected to be lower. Any arbitrary aggregation can be used instead of average, like maximum or median, but the average is preferred as it performed best in [36].

Dynamic time warping (DTW) is a measure created solely for use in the time series domain. DTW aligns two sequences by warping one sequence in the time dimension to match the other sequence, allowing for a more flexible comparison than other methods that require strict matching of each point in the two sequences. The algorithm works by calculating the distance between the corresponding points in the two sequences and finding the optimal alignment with the smallest overall distance. This measure has shown great results on various tasks and has been shown to be the best in many domains [37].

Local outlier factor (LOF) [38] is a method based on the concept of local density. Outliers are points that lie in regions of lower density. To obtain a density of a point A, first a reachability distance d_k to any point B is defined as $d_k(A, B) = \max(\text{core}_k(B), d(A, B))$, where $\text{core}_k(B)$ is the core distance to k-th nearest neighbor of B and d is a distance between A and B. The maximum has the effect of smoothing the distances to reduce the statistical fluctuations between all points A close to B [38]. Objects closer to a point than the core distance are given the reachability distance of the core distance and objects further away are assigned the true distance. With that, the local reachability density lrd_k can be defined as follows:

$$\text{lrd}_k(A) = 1/(\frac{\sum_{B \in N_k(A)} d_k(A, B)}{|N_k(A)|})$$

where $N_k(A)$ is a set of points in $\text{core}_k(A)$ distance of A called neighborhood of A. $\text{lrd}_k(A)$ is the inverse of the average local reachability density of points

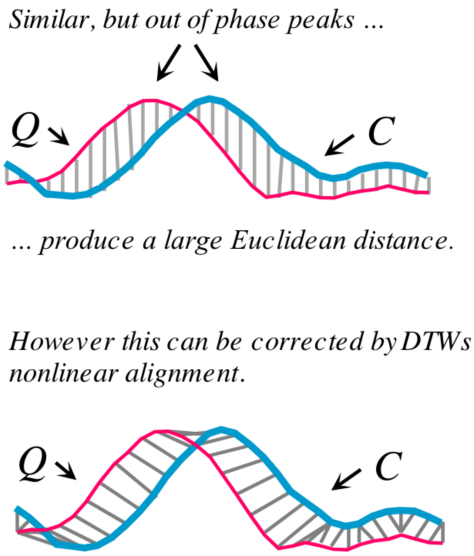


Figure 2.3: (Top) Distance between series Q and C measured using Euclidean distance (Bottom) Distance between series Q and C measured using DTW. Seemingly similar series but out of phase produce large Euclidean distance. [37].

in the neighborhood of A. Lastly, the Local outlier Factor LOF_k of point A is defined as:

$$\mathrm{LOF}_k(A) = \frac{\sum_{B \in N_k(A)} \mathrm{lrd}_k(B)}{|N_k(A)| \, \mathrm{lrd}_k(A)}$$

$\mathrm{LOF}_k(A)$ is an average of local reachability densities of points in the neighborhood of A divided by lrd_k of A. Points with higher LOF_k can be interpreted as points in sparser regions of the space. LOF_k is then used as anomaly score.

Quite a different approach of using distance is used in One-Class Support Vector Machine (OCSVM) [39]. OCSVM is an extension of Support Vector Machines (SVM) [40] to unlabeled data. SVM is a supervised method that tries to find a hyperplane to separate data of different classes. The hyperplane is chosen in such a way that it maximizes the margin between the classes, which means that it maximizes the distance between the nearest points in each class. These nearest points are known as support vectors, and they are used to define the hyperplane. SVMs additionally exploit the so-called kernel trick, where they can learn the hyperplane implicitly in up to infinite dimension space. OCSVM is an unsupervised version of this algorithm that assumes that only one class of data is present in the training dataset. The method learns a function that acts as a tight envelope of the presented data and scores each point in the envelope with a positive real number and points

outside the envelope negatively. The fitted function can be interpreted as a density function or a distance from a hyperplane that can be used as an anomaly score.

2.7 Distribution methods

Estimating the distribution of data is the concept of distribution methods. The anomaly score can be obtained as probability, likelihood, or frequency w.r.t. fitted distribution.

Histogram-based outlier score (HBOS) [41] is an anomaly detection method based on histograms and is suitable even for multivariate time series data. For each channel of the time series, a density is estimated by computing a histogram from dynamic-width bins, where each bin has the same amount of time points. After obtaining the histogram, normalization is applied so that the area of the histogram is equal to 1. An anomaly score for a point x in the multivariate time series is then calculated as:

$$\text{HBOS}(x) = \sum_{i=0}^{d} \log(\frac{1}{\text{hist}_i(x)})$$

where $\text{hist}_i(x)$ is a height of a bin where x is located and d is a dimension of the time series. Anomalous samples are expected to be in the tails of distributions and the height of the bins is expected to be smaller.

In survey [7] DWT-MLEAD [42] is the most efficient algorithm w.r.t. performance metric AUC-ROC. DWT-MLEAD uses multilevel discrete wavelet transformation (DWT) on time series and fits a Gaussian distribution over subsequences of the DWT coefficients using maximum likelihood estimation (MLE). The anomaly score is then computed with the likelihood of a sequence coming from that distribution.

2.8 Isolation Tree methods

Although isolation tree methods inherently use decision trees, they tend to be in a group of their own [7]. IsolationForest [43] is an ensemble of random decision trees. These trees are built recursively by randomly selecting features and splitting values as the tree nodes. Based on the assumption that anomalous samples are easier to separate from normal samples, the average number of splits to isolate a sample is used as a scoring function. On average the number of splits is lower for anomalous points. The detection tends to converge quickly with a growing number of trees. Combined with a small subsampling size of data for training so that normal samples do not interfere

with anomalous samples and the fact that most of the tree for normal samples is not needed, Isolation Forest methods are one of the most efficient methods [43].

Chapter 3

Evaluation

In this chapter, the reader is introduced to the evaluation of anomaly detection models. First, the procedure of unifying scores across models is shown. Next, the chapter covers evaluation under every possible label situation starting with supervised and semi-supervised evaluation, where first common evaluation metrics are presented followed by newer and more specialized evaluation schemes. Lastly, metrics for unsupervised learning are explained in depth as they are crucial for this work.

3.1 Unifying anomaly scores

As presented in chapter 2 there are various types of models that output an anomaly score in different forms. For evaluation, there is a need for each time point $\boldsymbol{x} \in \mathbb{R}^d$ to have an assigned score $s(\boldsymbol{x}) \in \mathbb{R}$. There are types of models that do not come with an implicit scoring function as an integral part of their design. For this reason, this section reports on various procedures for unifying the scores across the models.

The autoregressive and forecasting models evaluate each time point $\boldsymbol{x}$ based on its deviation from the predicted value. This could be a difference between two values or whether an observed point is in the confidence interval of the predicted value. These models require historical data to generate future predictions. Consequently, they cannot determine the score for the first k time points where k represents the required length of historical data. Typically, a score of zero is assigned to these points [7].

Models evaluating subsequences obtained via windowing of time series return anomaly scores for these subsequences. Each point within a particular window is assumed to be scored the same. A time series is split into windows with a stride of 1. Therefore, a time point can be a part of multiple windows. To obtain the final anomaly score of a point x across all the windows it is part

of an aggregation is typically applied. In [7], an average has been used as it performed better than the maximum. Hence why, an average is used in this thesis. The potential downside of this unifying approach could be that time points at the beginning and end of a time series are part of fewer windows.

3.2 Supervised and semi-supervised

Supervised anomaly detection methods require annotations for each time point in the dataset. A dataset with annotation for each element is marked as labeled [4]. In terms of anomaly detection, semi-supervised learning tries to learn from normal behavior, where no anomalies are introduced to the trained model in the fitting phase [7].

Given a fitted model with a scoring function s, that outputs a real number for input, and a threshold $t \in \mathbb{R}$ for the scoring function, so that data points x with score $s(x) \geq t$ are found anomalous, anomaly detection resembles binary classification, where the model decides whether a given time point x in a dataset $\boldsymbol{D}$ is anomalous or not.

In a binary classification setting, a time point $x \in \mathbb{R}^d$ can be classified as either 0 (negative) or 1 (positive). With given ground truth (GT) labels there are 4 possible combinations in total for classification and ground truth labels. Them being:

- **True positive (TP)** - GT label is 1 and classification label is 1.
- **True negative (TN)** - GT label is 0 and classification label is 0.
- **False positive (FP)** - GT label is 0 and classification label is 1.
- **False negative (FN)** - GT label is 1 and classification label is 0.

There are many metrics based on combinations of the previous 4 combinations, that evaluate a performance of a given model [44]. The important ones for anomaly detection are Precision, Recall, and False Positive Rate. The following performance metrics are interpreted in terms of anomaly detection. Precision $P = \frac{\text{TP}}{\text{TP+FP}}$ measures how accurate the model is when raising an alarm for an anomaly. Recall $R = \frac{\text{TP}}{\text{TP+FN}}$ quantifies the fraction of anomalies being correctly found within a dataset. Lastly, False Positive Rate $\text{FPR} = \frac{\text{FP}}{\text{TN+FP}}$ is the chance of a false alarm being raised for a normal sample of data.

When pairing precision and recall together, they capture the occurrence of both false positives and false negatives. The goal of models is to minimize both of these cases. A perfect model would have precision and recall equal to

1, but that is very unlikely to happen, due to large datasets, noisy observations, imperfect labels, etc. In practice, both metrics almost always cannot be prioritized together. Usually, the target is one of the two performance metrics, because the costs of conducting procedures based on alarms triggered differ. For example, if an automotive company would halt production for every anomaly alarm, then they would probably care more about precision as it might be less expensive to actually replace a faulty car rather than stop entire production due to a false alarm. In contrast, when deployed in medicine for monitoring patients, one would rather prioritize recall to capture all potential health-related issues. The severity of false alarms, in this case, is not as high as for example missing a disease in a patient. To combine precision and recall into one metric, with the ability to prioritize one of the two, F-measure $F_\beta = (1 + \beta^2)\frac{PR}{\beta^2 P+R}$, where β is the coefficient to prioritize between the two, has been introduced more than three decades ago. Papers in the field however set the β coefficient to 1, so that the results of the experiments are easier to compare. A F-measure with $\beta = 1$ is called F1-score $F_1 = 2\frac{PR}{P+R}$. F1-score is a harmonic mean between precision and recall.

As mentioned before, the fitted models return a scoring function s. The task of selecting a threshold for a function s is a field of its own. It is not trivial and is usually different for each model [7]. Different models may have different distributions of scores, which can affect the optimal threshold. Also as mentioned previously, the relative costs of false positives and false negatives may vary depending on the specific application. Therefore, selecting an appropriate threshold requires careful consideration of the specific problem at hand. It may require experimentation with different threshold values and evaluation metrics to determine the optimal threshold for a given application. Given labels, this can be learned as a hyperparameter, but that is usually not the case. Since the focus of this work is on unsupervised anomaly detection this task is not a part of this thesis. However, there are performance metrics that evaluate a model without the need for selecting the threshold. In this work, Receiver operating characteristic curve and Precision recall curve are explained. It has been shown that these threshold-agnostic metrics are more suitable for time series anomaly detection evaluation [45].

The receiver operating characteristic curve (ROC) is created by plotting the recall against FPR at different thresholds. To compare across multiple ROC curves, it is common to compute the area under the curve (AUC). AUC-ROC can then be used as an evaluation metric. When evaluating a model using AUC-ROC, one can imagine the performance of the model as how well it can rank negative and positive classes w.r.t. scoring function. Values of AUC-ROC lie in a range of $[0, 1]$. AUC-ROC can be interpreted as a probability of a uniformly taken sample from a positive class being ranked higher than a uniformly taken negative sample [46]. A model with AUC-ROC equal to 0.5

has ranking power as good as a model that would randomly guess the class. Absolutely perfect model has AUC-ROC of 1.

By the nature of anomaly detection, the problem is very imbalanced, the ratio of anomalies present in data should be very low ($\leq 5\%$), or else with a higher percentage of minority class the line between anomaly detection and plain classification blurs and calling an event that occurs so often an anomaly loses the meaning of being anomalous. In general, in anomaly detection, the anomalies are of interest, and because of that it is preferred to use the Precision recall curve instead of ROC [47]. The Precision recall (PR) curve is a plot of recall against precision. Replacing FPR with precision allows the metric to focus on positive (rare) class more as there is no TN term in either precision or recall. Another important difference comes from the interpretation of AUC-ROC as a probability of a uniformly taken sample from a positive class being ranked higher than a uniformly taken negative sample. For example in anomaly detection majority of non-anomalous time points are very easy for a model to rank, hence why they are the majority as they are easy to collect. That is why the probability of a uniformly sampled positive class being scored higher than the uniformly sampled negative class is so high. These easy to rank negatives are not of interest. What is more interesting are the negatives that are hard to rank and are on the edge between being anomalous and not. This is what the PR curve solves. By computing the area under the PR curve (AUC-PR) one can compress the curve into a single number in a range of $[0, 1]$, which can be used as a performance metric.

Algorithms tend to create false alarms when nearing the anomaly or slightly after. The transition from the state of being normal to the state of being anomalous usually does not happen instantaneously and cannot be captured by binary labels. For example when using the nearest neighbor models and scoring by distance to other points. It is expected that the anomaly score could slowly rise when approaching the anomaly and slowly decrease afterward. Furthermore, this is more of a case for anomalies spanning over several time points than for point outliers. This could also be a labeling issue of the dataset or by the nature of the algorithm. Keogh and Wu in [48] suggest allowing for algorithms to detect anomalies up to some particular number of points before and after the anomaly segment and still count it as TP. Paper [45] extends AUC-ROC and AUC-PR metrics dealing with the same issue by allowing detection of these points with linearly decreasing penalty before the anomaly and linearly increasing penalty after. The size of the windows before and after the anomaly can be made into another parameter of these curves making it surfaces in three-dimensional space under which volume can be computed.

3.2.1 Point adjusted evaluation scheme

In recent years papers [22, 49, 50] that evaluate time series anomaly detection models in any shape or form are likely to use the so-called point-adjusted (PA) evaluation scheme. The PA scheme [49] makes it so that is sufficient for a model to detect only one point in a contiguous anomaly segment for a whole segment to count as detected or as true positive. This simulates a real scenario where it is acceptable to trigger an alert for any point in a contiguous anomaly segment if the delay is not too long.

However, it has been shown in [51] that this can overestimate detectors' performance and even baselines can achieve high performances under this scheme. They also point out the lack of baselines used in evaluation comparisons in recent works. Meaning that some of the works could have misleading results. Together with papers [8, 48] evaluation of anomaly detection models is very much an open field and effort is being made to unify the benchmark process.

In [51] Point adjusted @ K (PA@K) protocol has been introduced to support the PA scheme. This protocol fills the gap between traditional and PA evaluation. To count contiguous segments as anomalous or true positive by any model it is required to detect at least $K\%$ of the segment as an anomaly. This scheme can be used by any of the previous metrics and is designed to be used alongside other metrics not instead of them. Nevertheless, not much derivative work with these approaches has been carried out.

3.3 Unsupervised

Having access to labeled data is crucial for training accurate and effective models. However, in reality, it is rare to have all or any labels available. To obtain those labels there is a need for a domain expert with extensive knowledge to evaluate the data. This process can be very expensive and/or time-consuming, which can be impractical. For this reason, the field of unsupervised anomaly detection has been getting more attention. This approach focuses on identifying anomalies or outliers in data without the need for labeled examples.

As stated in [52] no unsupervised evaluation metric for unsupervised model selection exists. However, in the literature, there are two metrics that have been used successfully for unsupervised hyperparameter selection in anomaly detection. Mass-Volume [53] and Excess-Mass [54] metric. One limitation of using these methods for time series data is that they require algorithms to use subsequences, as they were originally designed for anomaly detection in some hyperspace. This means that the methods need to split the time series data into subsequences and analyze each subsequence separately to detect anomalies.

3.3.1 Mass-Volume metric

Mass-Volume (MV) metric [53, 55] builds on an assumption that anomalies lie in sparse regions or tails of probabilistic distributions. From a statistical point of view, the data of n observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d, d \geq 1$ are an i.i.d. realizations of some unknown probability distribution P. Under these assumptions, the problem of anomaly detection can be formulated as estimating high density regions. With anomalies being outside of these regions. Looking for the regions is a problem of estimation of the minimum volume set:

$$\min_{B \in \mathcal{B}(\mathbb{R}^d)} \{\lambda(B), \text{ such that } P(B) \geq \alpha\},$$

where $\mathcal{B}(\mathbb{R}^d)$ is the set of all measurable subsets of $\mathbb{R}^d$ and λ the Lebesgue measure on $\mathbb{R}^d$. To estimate a minimum volume set one first estimates the density $h(\boldsymbol{x})$ and then thresholds it at an offset ρ such that the estimated set has an empirical mass α. In the view of unsupervised anomaly detection density function is substituted by scoring function $s : \mathbb{R}^d \to \mathbb{R} \in \mathcal{S}$. In contrast to the majority of models, this paper assumes that the lower the anomaly score the more anomalous the point is. The scoring function is then thresholded at ρ so that $B_\alpha = \{\boldsymbol{x}, \hat{s}(\boldsymbol{x}) \geq \rho\}$ is an estimation of a minimum volume set. Threshold ρ is obtained such that $P_n(\hat{s}(\boldsymbol{x}) \geq \rho) = \alpha$, where P_n is an empirical probability distribution of the observations. With that Mass-Volume (MV) curve can be defined as follows:

Definition 3.1 (Mass Volume curve[55]) *The Mass Volume curve MV_s of a scoring function $s : \mathbb{R}^d \to \mathbb{R}$ is defined as the parametric curve $t \in \mathbb{R} \to (\alpha_s(t), \lambda_s(t))$ where $\alpha_s(t) = P(s(\boldsymbol{x}) \geq t)$ and $\lambda_s(t) = \lambda(\boldsymbol{x}, s(\boldsymbol{x}) \geq t)$. If α_s has no flat parts, the Mass Volume curve MV_s can also be defined as the function*

$$MV_s : \alpha \in (0, 1) \to \lambda_s(\alpha_s^{-1}(\alpha))$$

where $\alpha_s^{-1}(\alpha) = \inf\{t \in \mathbb{R}, \alpha_s(t) \leq \alpha\}$.

With MV curve defined, the task of selecting optimal parameters $\theta^* \in \Theta$ is to minimize the following distance:

$$\theta^* = \arg\min_{\theta \in \Theta} \|\mathrm{MV}_{\hat{s}_\theta} - \mathrm{MV}^*\|_{L_1} = \arg\min_{\theta \in \Theta} \|\mathrm{MV}_{\hat{s}_\theta}\|_{L_1}$$

where MV^* is an optimal curve, that is known to be smaller or equal at $\forall\alpha$ [55], and L_1 is a distance between the estimated and optimal curve. This is equivalent to minimizing the area under $\mathrm{MV}_{\hat{s}_\theta}$ over interval I noted as $\mathrm{AMV}_I(s)$ and defined over an interval $I = [\alpha_1, \alpha_2]$ as:

$$\mathrm{AMV}_I(s) = \int_{\alpha_1}^{\alpha_2} \mathrm{MV}_s(\alpha) d\alpha$$

In practice, the interval I is usually set to $[0.9, 0.999]$ as there is the assumption that the contamination rate of anomalies does not exceed 10% and that MV diverges in 1 [56]. The only point left uncovered is the estimation of the λ measure. This is covered in the next section as the Excess-Mass metric uses it as well.

3.3.2 Excess-Mass metric

Excess-Mass (EM) [54, 56] metric works with the same assumption as MV and can be defined without any additional notation as:

Definition 3.2 (Excess Mass curve) *The Excess Mass curve EM_s of a scoring function $s : \mathbb{R}^d \to \mathbb{R}$ is defined as the function*

$$EM_s : t \in [0, +\infty] \to \sup_{\rho \in \mathbb{R}} \{\alpha_s(\rho) - t\lambda(\rho)\}$$

where $\alpha_s(\rho) = P_n(s(\boldsymbol{x}) \geq \rho)$ and $\lambda_s(\rho) = \lambda_s(\boldsymbol{x}, s(\boldsymbol{x}) \geq \rho)$ Lebesque measure.

EM_s can be viewed as recovering a collection of subsets $(B_t^*)_{t>0}$ with maximum mass when penalized by their volume in a linear fashion [54]. Similarly to MV, the area under the EM curve is computed in the interval $I = [t_1, t_2]$ with one difference in that for optimal parameters the area is maximized:

$$\mathrm{AEM}_I(s) = \int_{t_1}^{t_2} \mathrm{EM}_s(t)dt.$$

The interval used in [56] was $I = [0, \mathrm{EM}^{-1}(0.9)]$, where $\mathrm{EM}^{-1}(0.9) = \inf\{t \geq 0, \mathrm{EM}_s(t) \leq 0.9\}$. EM curve has several theoretical advantages. One of those being its finite range.

As mentioned in the previous section both of the metrics use λ measure to compute a volume. The volume is estimated by using Monte Carlo integration by generating uniform samples in the hypercube enclosing the data. To uniformly sample with particular granularity it is needed to exponentially increase the number of samples with each additional dimension. This becomes computationally expensive and can only be done in smaller dimension spaces. To scale with dimensions a methodology has been introduced in [56]. In this methodology, a random subsample along the features for both training and testing dataset splits is taken and training and testing are done only on the selected features. This procedure is then run multiple times and the average of the runs is taken as a metric value.

According to paper [56], both the EM and MV metrics can accurately be used to discriminate between methods w.r.t. PR and ROC curves. However, not

many experiments have been carried out in this regard, and this is one of the objectives of our own experiments.

Chapter 4

Automated Machine Learning

Overall, this chapter provides the reader with a comprehensive overview of the various applications of AutoML in unsupervised anomaly detection and demonstrates how these techniques can help automate and streamline the process of model building and optimization. The insights provided can be valuable to practitioners interested in applying AutoML techniques in this domain and are used in experiments.

Automated Machine Learning, or AutoML, simplifies the process of building and deploying machine learning models by automating tasks such as data preprocessing, feature engineering, model selection, and hyperparameter tuning.[52] This makes it easier for both novice and experienced users to build models quickly and accurately.

4.1 Combined Algorithm Selection and Hyper-parameter tuning

Usually, the second form of AutoML (Right after basic hyperparameter optimization) that AI practitioners get in touch with is Combined Algorithm Selection and Hyperparameter tuning (CASH) defined as 4.1. Selecting correct models is as crucial as tuning hyperparameters for many algorithms to give good results. This section reviews various techniques, from basic concepts to state-of-the-art methods using Bayesian statistics, bandit-based optimization, or a combination of both.

Definition 4.1 (CASH) *Given a set of machine learning models* $\mathcal{M} = \{M^{(1)}, M^{(2)}, \dots\}$*, and a dataset* $\boldsymbol{D}$ *divided into training* $\boldsymbol{D}_{train}$ *and validation* $\boldsymbol{D}_{validation}$ *sets. With the goal of selecting a model* $M^{(i)^*}$ *where* $M^{(i)} \in \mathcal{M}$ *and* $M^{(i)^*}$ *is version with tuned hyperparameters of* $M^{(i)}$ *that achieves the highest performance by training on* $\boldsymbol{D}_{train}$ *and evaluating on* $\boldsymbol{D}_{validation}$*. The Combined Algorithm Selection and Hyper-parameter tuning problem can then be*

defined as:

$$M^{(i)^*} \in argmin_{M \in \mathcal{M}} L(M^{(i)}, \boldsymbol{D}_{train}, \boldsymbol{D}_{validation}),$$

where L is a loss function.[57]

The very first AutoML method that is usually introduced to newcomers to the field is a Grid search[58] Grid search evaluates the model on every combination of values of configuration space. This makes it rather expensive in practice and often is used only for introduction to AutoML on toy problems. A simple approach, yet much better for larger problems, is called Random search. The random search[58] evaluates models on randomly sampled configurations until some set budget is depleted. This budget could be a time constraint or a number of evaluations. Both of these approaches suffer from the so-called curse of dimensionality as with each additional hyperparameter the space grows exponentially, despite the problems, their advantage is in easier parallelization.[52]

The validation performance of machine learning algorithms can be modeled as a function $f : \mathcal{X} \to \mathbb{R}$ of their hyperparameters $x \in \mathcal{X}$. The function f is assumed to cannot be observed directly only through noisy observations $y(x) = f(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$. Bayesian optimization uses a probabilistic model $p(f|D)$ to model the objective function f based on the already observed data points $D = \{(x_0, y_0), \ldots, (x_{i-1}, y_{i-1})\}$. Acquisition function $a : \mathcal{X} \to \mathbb{R}$ is then used as a heuristic to select the potentially best next point. With the probabilistic model and acquisition function in place 3 steps are repeated iteratively: Acquire a point x^* that maximizes function a, evaluate with the function f, and augment already observed data points with point x^*.[59]

Tree-Structured Parzen Estimator (TPE) [60] is one of the methods of Bayesian optimization. TPE uses a kernel density estimator to model the densities $l(x) = p(y < \alpha|x, D)$ and $g(x) = p(y > \alpha|x, D)$ over the input configuration space instead of modeling the objective function f directly by $p(f|D)$. In this case, α is a quantile q of observations of y, satisfying $p(\alpha > y) = q$. To select the next point x^* to evaluate, the ratio $\frac{l(x)}{g(x)}$ is maximized.[59] TPE has been introduced to be used to optimize deep belief networks [60] and since then is implemented in many AutoML libraries.

Given that many machine learning algorithms are trained iteratively. Hyperparameter optimization can be modeled as a multi-armed bandit problem [61]. Imagine a bandit coming to a casino to play slot machines, these slot machines have an arm to pull to play. The bandit needs to decide which machines to play to maximize his reward. The analogy of the multi-armed bandit problem to hyperparameter optimization is a problem where a limited and fixed

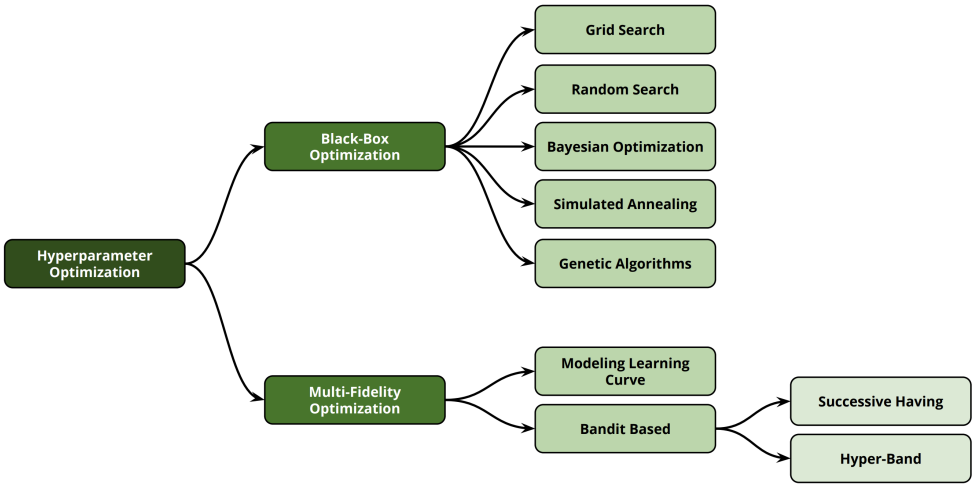


Figure 4.1: Taxonomy of hyperparameters optimization techniques from [57].

amount of resources needs to be allocated between available models or individual hyperparameter settings with the goal of minimizing loss. This can be seen as a pruning of unpromising model configurations that seem likely to not give good results. These methods tend to noticeably speed up the search for optimal configurations.

One of the recent popular algorithms that employs this concept is called Successive Halving (SH) [62]. The idea behind the algorithm is simple: Given an input budget $\mathcal{B}$, uniformly allocate the budget to a set of arms for a predefined number of iterations, evaluate their performance, throw out the worst half, and repeat until just one candidate remains. SH needs to know n the number of configurations in advance. It is not clear how to choose n, whether to take larger n and train shorter on average or take smaller n and train longer. Hyperband (HB)[63] deals with this by performing multiple SH runs over different values of n with the same budget for each run. This approach is particularly effective for optimizing deep learning models with a large number of hyperparameters, as it balances the exploration and exploitation of the hyperparameter space. It has been shown to be magnitudes faster than random search to optimize to the same performing models in the early stages of optimization. But with random search guaranteed to converge to the optimal model, the advantage shrinks for longer optimizations.

BOHB[59] combines Bayesian optimization and Hyperband to bring together a strong final performance of BO with strong anytime performance, scalability, and robustness of HB. In BOHB, HB manages the resources and budget, and BO is left with the selection of configurations using model-based search.

4.2 Meta-learning

Humans learn how to learn across tasks. With every skill learned, learning new skills becomes easier, requiring fewer examples and less trial-and-error. We start from skills learned earlier in related tasks, reuse approaches that worked well before, and focus on what is likely worth trying based on experience. [64] Likewise, when building machine learning models for a specific task, we often build on experience with related tasks or use our (often implicit) understanding of the behavior of machine learning techniques to help make the right decisions. This approach to learning is called meta-learning.

In machine learning, meta-learning can be categorized into 3 groups based on [64]:

- Learning from model evaluations.
- Learning from task properties.
- Learning from prior models.

Having P a set of all scalar evaluations $P_{i,j} = P(\theta_i, t_j)$ of configurations $\theta_i \in \Theta$ in configuration space and tasks $t_j \in T$ of known tasks. Here Θ represents a space of configurations consisting of hyperparameter spaces, pipeline configurations, and/or network architectures. Learning from model evaluations can be used to recommend configurations θ^* for unseen tasks t_{new} by learning a function $f : \Theta \times T \to \theta^*$. Another approach that uses model evaluations uses them to learn better configuration spaces Θ^*. This can drastically save the search time for optimal models[65]. Some of the other concepts of using model evaluations are transferring configuration spaces for new tasks that are similar to some of the known tasks, and surrogate models that learn to predict evaluation performances without the need to evaluate models[66] as that can be very expensive in some cases.

Learning from task properties uses meta-features as meta-data. Each task $t_j \in T$ is described by the vector $m(t_j) = (m_{j,1}, \ldots, m_{j,k})$ of k meta-features. These features can be used to determine a similarity between tasks. When a new task t_{new} is presented, the most similar known task can be used found to transfer information. Meta-features can represent summary statistics, aggregations, statistics of distributions, or even features of simple models learned on the tasks. This group is elaborated on in further sections as the concepts are being used in this thesis.

The last group of concepts is concerned with learning from prior models, learned hyperparameters, or their structure. Transfer learning[67] takes a model that is already pre-trained on previous tasks and uses it as a warm start

for fine tuning on a new task. Few-shot learning is a part of this group as well. Few-shot learning deals with training a model with only a few examples available given a prior experience of training on similar tasks with a large dataset on hand.

4.3 AutoML for unsupervised anomaly detection

A paper from Bahri et al.[52] from February 2022 surveys recent advancements in the field of AutoML for unsupervised anomaly detection. Section 2.1 of the paper lists current challenges that must be tackled to improve the state-of-the-art further. Some of the problems stated are defining search space for hyperparameters being not trivial, an issue with cold starts of models and spending too much time on wrong models, running time, high dimensional data, or scalability. Nevertheless, the most critical problem related to this thesis is evaluation metrics. Evaluation metrics are used primarily to choose between different models generated by an automated framework. The choice of these metrics in machine learning is not trivial and depends on many factors. However, no suitable unsupervised evaluation metric for model selection in anomaly detection exists [52].

The nonexistence of evaluation metrics is a great limitation as all traditional AutoML methods are built on them. A popular approach in recent papers that tries to overcome this issue is meta-learning. MetaOD[68, 69] leverages information about datasets with known labels. The basic concept is a recommendation system that is trained on labeled datasets (can be seen as users) with trained models (can be seen as items) and recommends models using collaborative filtering to a new unseen unlabeled dataset based on meta features extracted from the dataset. The method is described in better detail in section 4.3.1 as it is a part of the experiments of this work.

Another approach, Meta-AAD [70], uses learned meta policy using deep reinforcement learning to query anomalous instances. The meta-learning part is transforming datasets into 6 crafted features that are fed as input in a streaming fashion to meta-policy that decides whether an instance is normal or anomalous. After training the policy it can be used on unseen data without any further tuning.

4.3.1 MetaOD

MetaOD is termed the first data-driven approach to the selection of unsupervised outlier detection models [68]. This data-driven approach leverages a similarity of tasks and a historic performance of models on known labeled datasets to recommend promising models for new tasks on unseen nonlabeled data. Collaborative filtering (CF) is used as a recommendation engine, where

datasets represent users and models stand for items. When an unseen dataset comes best promising model is recommended based on similarity to known data. This similarity is partially captured by meta-features that are then projected into a latent space of dataset latent features created by matrix factorization within CF. Unlike the usual recommendation systems, MetaOD is trained under a Top-1 setting where only the best model matters.

The problem of model selection with a finite pool of models denoted $\mathcal{M} = \{M_1, \ldots,$
$M_m\}$ can be defined as

Definition 4.2 (Unsupervised Outlier Model Selection) *Given a new input dataset (i.e., detection task) $D_{test} = (X_{test})$ without any labels, Select a model $M \in \mathcal{M}$ to employ on X_{test}. [69]*

MetaOD is based on a meta-train database of datasets $\mathcal{D}_{\text{train}} = \{\boldsymbol{D}_1, \ldots, \boldsymbol{D}_n\}$ with ground truth labels $\{\boldsymbol{D}_i = (X_i, y_i)\}_{i=1}^{n}$ and a historical performance of models $\mathcal{M}$ on meta-train datasets $\mathcal{D}_{\text{train}}$. Historical performances are inserted into a performance matrix $\boldsymbol{P} \in \mathbb{R}^{n \times m}$. $P_{i,j}$ corresponds to performance of model M_j on dataset $\boldsymbol{D}_i$. In the paper [69], Average precision has been used as a performance metric. Models can be evaluated on train datasets in a supervised setting as labels are present. The inner workings of the approach can be split into two phases. Offline training phase and online model selection phase.

Offline training

To be able to leverage prior results, MetaOD relies on meta-features of datasets. The extracted features are separated into statistical features and landmark features. The former consists of statistical properties of underlying data distributions, etc. The optimal set of meta-features has been shown to be application-dependent [64]. Labeled as perhaps more important, the landmark features are obtained from easy-to-construct outlier detection models. These extracted features tend to be more problem-specific. A full list of features and elaboration on them can be found in Appendix B in [69]. The process of extracting meta-features is denoted as $\psi(.)$, then the matrix of all meta-features of all meta-train datasets can be written as $\boldsymbol{M} = \psi(X_1, \ldots, X_n) \in \mathbb{R}^{n \times d}$.

At this point, the application of a recommender system to unsupervised model selection is clearer. The performance matrix can be seen as a user-item rating matrix, where datasets are users and models are items. When a new user comes, the recommender recommends items that the user would likely rate higher. This can be done in several ways. MetaOD uses model-based CF using matrix factorization. The performance matrix $\boldsymbol{P}$ can be factorized into $\boldsymbol{U} \in \mathbb{R}^{n \times k}$ user and $\boldsymbol{V} \in \mathbb{R}^{m \times k}$ item matrix consisting of k features in latent

space. These latent features are captured by leveraging properties of matrix multiplication, such that $\boldsymbol{P} \approx \boldsymbol{U}\boldsymbol{V}^T$.

For purposes of model selection, where the ranking of recommended models is more important than the perfect reconstruction of performance, the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are learned by optimizing the discounted cumulative gain (DCG)[71] instead of the usual mean square loss. To use the meta-features $\boldsymbol{M}$ the matrix $\boldsymbol{U}$ is initialized using an embedding function $\phi : \mathbb{R}^d \to \mathbb{R}^k$, where $k < d$, on $\boldsymbol{M}$. The principal component analysis can be substituted as $\phi(.)$, then $\boldsymbol{U}^{(0)} = \phi(\boldsymbol{M})$.

The values of matrix $\boldsymbol{U}$ will inevitably change during optimization. This creates a problem for new testing dataset D_{test} because of a need for initializing $\boldsymbol{U}_{\text{test}}$ using extracted meta-features M_{test}. The multi-output regression function $f : \mathbb{R}^k \to \mathbb{R}^k$ is learned in every optimization step to overcoming this problem. The function f maps the latent space of meta-features to the latent space of optimized $\boldsymbol{U}$.

Online model selection

The output of the training phase are estimated functions $\psi(.)$, $\phi(.)$, $f(.)$ model matrix $\boldsymbol{V} \in \mathbb{R}^{m \times k}$. Given a new unseen dataset D_{test}, prediction consists of a few steps. Meta-features are extracted $M_{\text{test}} = \psi(D_{\text{test}})$, these are then embedded and regressed into latent space of optimized $\boldsymbol{U}$, $U_{\text{test}} = f(\phi(M_{\text{test}}))$. The predicted performances for the set of models are obtained by $P_{\text{test}} = U_{\text{test}}V^T \in \mathbb{R}^m$. The best-performing model with the highest predicted performance from P_{test} is selected as the final model.

4.4 Window size selection

Most state-of-the-art algorithms only look at local structures (windows) in time series [4, 7, 9]. It is crucial for the performance of the algorithms to find the optimal length of the subsequences. With a poor choice of window size algorithms tend not to give good results [72]. In a supervised setting, this length can be optimized as another hyperparameter. However the same cannot be done in an unsupervised environment. Therefore, the need arises for a domain-agnostic method for learning the window size. Part of the paper [72] by Ermshaus et al. surveys these methods specific for anomaly detection. The methods can be roughly separated into two categories, whole-sequence-based and subsequence-based. Some of the basic concepts and ideas are presented in this section. Furthermore, Autoperiod[73] is described in detail as it is later used in the experiments section of this work.

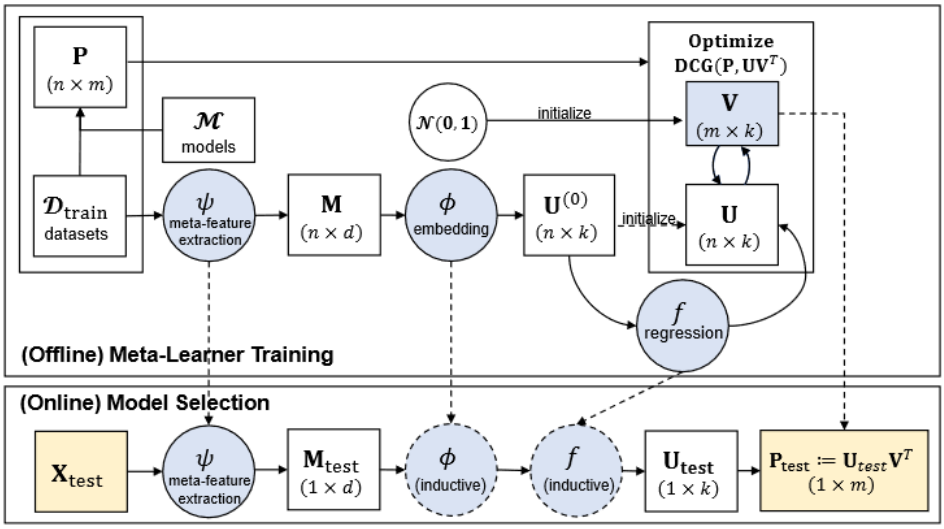


Figure 4.2: Diagram of MetaOD data-driven meta-learning approach for model selection [68, 69].

Whole-sequence based

When it comes to whole-sequence-based methods, the algorithms usually separate further into two categories. Methods working in the time domain look at how time series autocorrelate when shifted by an offset (lag). Methods working in the frequency domain use one of the Fourier transformation algorithms to detect frequencies with a high magnitude.

Dominant Fourier frequency (definition 4.3) converts time series to the frequency domain using Discrete Fourier Transform (DFT) and then computes magnitudes from complex coefficients with the following equation $m_k = \sqrt{\text{Re}(c_k)^2 + \text{Im}(c_k)^2}$, frequency with highest magnitude is chosen as the most dominant.

Definition 4.3 (Discrete Fourier Transform) *The Discrete Fourier Transformation (DFT) of a time series T, $|T| = n$ is a series of complex coefficients $C := (c_0, \ldots, c_{n-1}) \in C_n$, such that $c_k := \sum_{j=0}^{n-1} T_j e^{-2\pi i \frac{jk}{n}}$, for $k = 0, \ldots, (n-1)$ and $i = \sqrt{-1}$. The corresponding series of frequencies is defined as $\mathcal{F} = (f_0, ..., f_{\lceil \frac{n-1}{2} \rceil})$, such that $f_k := k/n$, for $k = 0, \ldots, \lceil \frac{n-1}{2} \rceil$. [72]*

The autocorrelation function (ACF) (definition 4.4) computes the crosscorrelation between the time series and its copy offset by a delay in time. If a period appears in the data, the autocorrelation value is high in that lag. The

dominant window size is selected as local maxima in the autocorrelation of time series, not taking into account lag zero.

Definition 4.4 (Autocorrelation Function) *The Autocorrelation Function (ACF) of a time series T, $|T| = n$ defines the zero normalized cross-correlation*

$$a(l) := \frac{1}{n-l-1} \sum_{j=l+1}^{n} \frac{(T_j - \sigma_T)(T_j - l - \mu_T)}{\sigma_T^2}$$

for a given lag l and with μ_T, σ_T being the mean or, rather, the standard deviation of T. The series of cross-correlations $A := (a(0), ..., a(n-1)) \in \mathbb{R}^n$, is the autocorrelation (AC) of T. [72]

There are many hybrids that combine or modify these methods such as AutoPeriod described in section 4.4.1 or RobustPeriod [74], which firstly detrends the time series and then uses the combination of DFT and ACF to select dominant window size.

Subsequence based

Subsequence-based methods extract local features from subsequences and compare statistics with statistics of the whole time series. The assumption is that windows with statistics that align well with the entire time series statistics capture a temporal pattern that repeats throughout the time series.

Multi Window Finder (MWF) [75] is based on the hypothesis that, given a suitable window size, its variance of moving averages is small. For a given range of candidate sizes, a moving average of a particular size is computed and then a variance is measured as the distance of the moving average to the average of moving averages. Suitable window sizes are found as local minima in variance w.r.t. window size.

Summary Statistics Subsequence (SuSS) [76] compares summary statistics (standard deviation, mean, range of values) computed over subsequences with those computed over the entire time series. Again with the assumption that statistics of windows with suitable size are close to those of the original time series.

4.4.1 AutoPeriod

AutoPeriod is a non-parametric method that combines two techniques, autocorrelation, and periodogram, for periodicity detection. The periodogram identifies potential periods from the sequence, whereas the autocorrelation estimates possible periodicities. Both the periodogram and the autocorrelation can be directly computed using the Fast Fourier Transform in $O(N \log N)$

time.

AutoPeriod is based on a combination of DFT and ACF. In the first part, candidate periodicities are found and are validated in the second. A diagram of the algorithm can be seen in Figure 4.3. To find dominant frequencies, one needs to find a power threshold for the periodogram $\mathcal{P}(f_{k/N}) = \|\mathcal{F}(f_{k/N})\|^2,\ k = 0, 1, \ldots, \lceil \frac{N-1}{2} \rceil$ obtained from DFT. DFT is run 100 times on permutations of the original time series, each time picking a magnitude of the most dominant frequency so that a 99% confidence interval can be provided. The power threshold p_T is then simply selected as the 99th percentile of the magnitudes obtained. All frequencies in the original time series with a magnitude higher than the selected threshold are considered candidates for the next part of the algorithm denoted as $p_{\text{hint}} = \{N/k : \mathcal{P}(f_{k/N}) > p_T\}$. False positives can be created due to potential spectral leakage or just due to coarse estimation by discrete frequency bins in DFT. Hence, there is a need for verification in the second part of the algorithm. Verification is done by computing the ACF for the original time series and looking at where the candidate periods lie on the autocorrelation function. If the candidate's period lies at a peak, it indicates that the period is important. The opposite would be a period lying in a trough, which would indicate a false positive. To decide whether a candidate point lies on a hill or in a valley in an ideal setting it could be determined by the curvature of the ACF around that point. The second derivative would be computed to look at whether a point lies in a concave segment of the ACF. This cannot be done reliably due to the existence of noise that could invalidate the requirements. A much simpler algorithm can be used. This can be done by approximating a two-segment linear regression in a region around the point. The best splitting point t can be found by minimizing the following equation:

$$t_{\text{split}} = arg\min_t \epsilon(\hat{S}_1^t) + \epsilon(\hat{S}_{t+1}^n),$$

where $\hat{S}_a^b$ is a linear approximation between positions a and b and $\epsilon(\hat{S}_a^b)$ is an error introduced by approximation of the segment. Finally, the last remaining issue to be solved is the width around the point where the segments should be fitted. Since the periodicity hint might have leaked from adjacent DFT bins (if it was located near the margin of the bin), half of the adjacent bins are also examined. Therefore, for a hint at period N/k, the range $R_{N/k}$ of the ACF is examined for the existence of a hill. $R_{N/k} = [\frac{1}{2}(\frac{N}{k+1} + \frac{N}{k}) - 1, \ldots, \frac{1}{2}(\frac{N}{k} + \frac{N}{k-1}) + 1]$. In the selected region the closest peaks of those candidate periods can be found in two ways:

- Information about the peak could be derived from previous linear segmentation the peak should be situated either at the end of the first segment or at the start of the second.

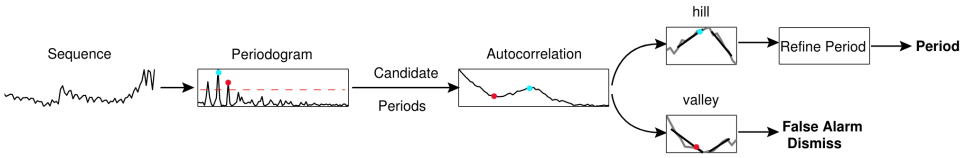


Figure 4.3: Diagram of AutoPeriod [73] method.

- By local search in a manner towards the direction of the positive first derivation of the autocorrelation function.

Out of all the methods presented earlier, AutoPeriod [73] ranked first in anomaly detection benchmark in [72]. Although the difference to other methods has not been statistically significant it performed the same or better than human annotators. The method has been tested on UCR datasets specifically made for anomaly detection introduced in paper [48], which points out flaws in current anomaly detection benchmarking procedures. For this thesis, these datasets are used in the experiments section, together with AutoPeriod used for window size selection.

Chapter 5

Research statement

The motivation for this chapter is to summarize the research questions that are attempted to be answered in the experiments chapter of the work. The experimental chapter contains references to these questions to make navigation throughout the experiments seamless and easier to comprehend.

The current state of unsupervised anomaly detection requires highly specialized data scientists with a significant amount of domain knowledge to correctly assemble a pipeline to achieve good results. Little research effort has been devoted to AutoML in unsupervised anomaly detection [52].

With our experiments, emphasis is placed on two parts of the overall machine learning pipeline. Them being a hyperparameter selection and model selection. The overall objective is to provide procedures for solving unsupervised AutoML for anomaly detection in the time series domain. Currently, there are no known reliable approaches for solving this task as presented in the Introduction and Section 4.3. In this thesis, the new pipelines are made up of existing methods. The main parts are the mass-volume[55], excess-mass[54] metrics, as no other metrics exist, and meta-learning approach for automated model selection for outlier detection from Zhao et al. [68].

Before proceeding with the main experiments, an important step is to determine the optimal size of subsequences for the methods designed to use them. This is a crucial first step, as choosing the right size allows for optimal performance of the subsequent components of the pipeline. For this reason, AutoPeriod mentioned in Section 4.4 has been tested.

Research insights that we would like to gain are:

Q.0 *How does AutoPeriod perform as an unsupervised method for suitable window length selection?*

Q.1 **1** *Can MV and EM metrics be used for reliable hyperparameter selection?*

2 If the previous stands to be true, *Can MV and EM metrics be used for model selection?*

3 *Can MV and EM metrics be extended to allow for some other models to be used?*

Q.2 **1** *Does meta learning approach for unsupervised anomaly detection from [68] carry over to unsupervised anomaly detection in time series?*

2 *Can it be improved specifically for the time series domain?*

Chapter 6

Experiments

This chapter introduces the reader to the process of conducting experiments in this thesis, including selecting datasets, choosing hardware and software setups, designing training and testing procedures, and creating pipelines. The thought process behind each step is also discussed. The chapter ends with presented results and a thorough discussion.

6.1 Datasets

In this section, frequently used datasets for univariate time series anomaly detection are presented and described. Eventually, the selection of the datasets for own experiments is discussed.

The terminology for data used in the field can be confusing. To avoid misunderstandings, it is explicitly stated for this thesis that the terms dataset and time series are used interchangeably and mean the same thing. A collection of data that is used to train a model. It is common that when new data are introduced it is a collection of datasets (time series) and models are trained separately for each dataset and the performance is aggregated or the models are ranked on all of them.

In a paper from Yahoo [77] datasets comprised of a mixture of 50/50 real and synthetic time series are introduced, as well as a synthetic time series generation tool, allowing users to generate data with specific parameters like length, magnitude, anomaly density, anomaly type, etc. The real part of the dataset comes from Yahoo's membership logins and contains roughly 1400 data points representing 3 months' worth of data. However, a limitation of these datasets is that they are provided only for academic purposes and are locked behind a registration process.

Numenta[2] created Numenta Anomaly Benchmark (NAB) [78] with the goal to provide a labeled data stream from real-world applications, scoring methodology, a set of constraints designed for streaming applications, and a controlled open repository for researchers to evaluate and compare anomaly detection algorithms for streaming applications. Part of a NAB are datasets with the purpose of presenting anomaly detection algorithms to real-world scenarios. The NAB datasets contain anomalies covering the taxonomy presented, clean and noisy data, and time series with evolving statistics over time. The first version contains 58 data streams, each with 1000 – 22000 records, for a total of $\sim$ 360000 data points.

Another popular set of datasets comes from NASA where Hundman et al. [79] introduced Soil Moisture Active Passive (SMAP) satellite and the Mars Science Laboratory (MSL) expertly-labeled datasets and successfully presented recurrent neural networks as anomaly detectors in overcoming the issues of the burden placed on operations engineers and operational risk. These datasets are comprised of 55 SMAP and 27 MSL time series. Contamination of the datasets is around 13% and 11% and the data are already split into training and testing sets for better comparability of results. The contamination rate seems to be on the edge of what is still acceptable.

In a paper [80] proposing a new stochastic recurrent neural network for multivariate time series anomaly detection called OmniAnomaly authors used earlier cited NASA datasets in combination with newly introduced Server Machine Datasets (SMD) consisting of 28 time series, for a total of $\sim$ 1400000 data points with anomaly ratio of approximately 4.1%.

Despite the popularity of these datasets, in 2020 Wu and Keogh [48] published a paper discussing the flaws of these datasets and suggested that future research uses the newly introduced UCR Time series anomaly archive. The main flaws pointed out in the paper were as follows:

- **Triviality** - 86% of the datasets are solvable by one line of MATLAB[3] code consisting only of primitive vectorized functions such as *mean*, *max*, *std*, *diff*, etc. With this issue, authors tried to prove a point that if a dataset can be solved with a simple function it strongly suggests no need for much more complex approaches to be tested and developed on those.
- **Unrealistic anomaly density** - Some of the samples have more than half test data consisting of contiguous regions marked as anomalies. There are many regions marked as anomalies. In some datasets, the anomalies are very close to each other. In extreme cases, two anomalous

[2] www.numenta.com/
[3] www.mathworks.com/products/matlab

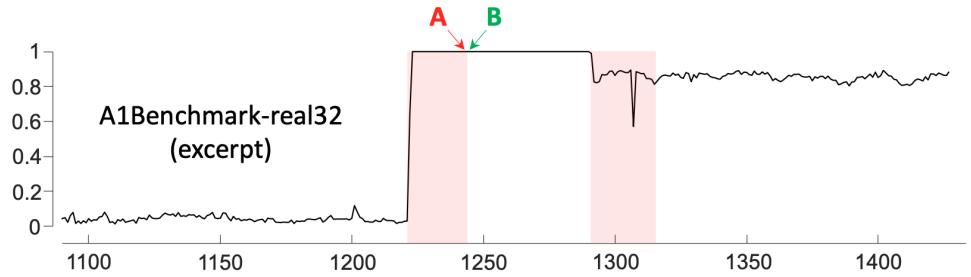


Figure 6.1: A figure from [48] illustrating mislabeled ground truth where an algorithm would be penalized for predicting point B as an anomaly even though it's on the same line as point A.

regions surround one normal data point. With such unrealistic anomaly density, the line between classification and anomaly detection is blurred.

- **Mislabeled ground truth** - All of the benchmark datasets appear to have mislabeled data, both false positives and false negatives. In figure 6.1 detector that points to B will be penalized as having a false positive, but point A is a part of the same constant line. Figure 6.2 illustrates New York Taxi dataset from NAB with anomaly score. For this dataset, only 5 ground truth labels are provided. After careful analysis, more than 5 additional labels being equally worthy as the provided could be found [48]. Additionally, the ground truth label for marathon in figure 6.2 is in reality caused by daylight saving time.

- **Run-to-failure bias** - Many real-world systems are run-to-failure, meaning that in many cases there are no data after the last anomaly. This seems to be the case at least for Yahoo datasets. The issue is that a naive algorithm labeling the last data point as an anomaly could have a good probability of being correct.

With the flaws in mind, the goal of the paper [48] was to create new datasets. The authors created datasets that are synthetic but highly plausible and come from various fields. Only one anomaly region is present in a single dataset to ensure low anomaly density to reflect real-world scenarios. The complexity of the problems is on a spectrum ranging from a small fraction of one-liners to very hard problems. The datasets are already split into training and testing sets so that the results obtained using these data are comparable across different works. UCR archives are widely used and popular for time series classification datasets that are being used mainly to compare state-of-the-art models in the field.

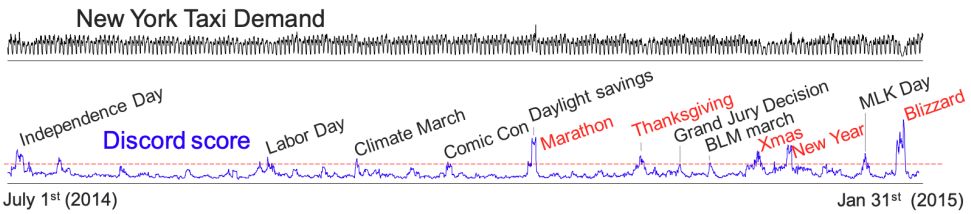


Figure 6.2: NYC Taxi dataset with discord/anomaly score (in blue) of the dataset. Red annotated peaks are the given ground truth labels. After careful analysis, more than 5 additional labels (black annotated peaks) being equally worthy as the provided could be found [48].

6.1.1 Selected datasets

The primary focus of this work regarding the data was on the UCR Time series anomaly archive. The archive is comprised of 250 time series. For experiments, time series were chosen according to their length for computational budget reasons. Some of the datasets were enormous in length compared to the rest as can be seen in figure 6.4. By continuously lowering the length threshold 100 time series were discarded. The final value of the threshold was 50000 data points. Out of those remaining 150 time series 50 were randomly chosen as test time series and the remaining 100 were put aside as they are used for training of MetaOD approach. The statistics of selected UCR datasets can be seen in figure 6.5. For illustration purposes, randomly chosen time series with highlighted anomaly regions are displayed in figure 6.3.
The flaw of these datasets is in their lack of anomalies in the training split. This makes the datasets semi-supervised and some of the detectors work better in this setting. In practice, it often happens that not much information is given on labels and therefore there is a need for additional datasets to be used to thoroughly evaluate the designed pipelines. For this purpose, the experiments were also run separately on the NAB dataset.

NAB datasets do not come with a predefined train test split. For this reason, a split in a 50/50 ratio was done, where the first half was used in training and the second for evaluation. After the split, datasets that had no anomalies either in the training or testing split were discarded. To avoid a semi-supervised setting or no anomalies to be detected. As a result, 33 out of 58 datasets were discarded. The same statistics as for UCR datasets can be seen in figure 6.6.

6.2 Setup

In recent years, the demand for data collection has increased, leading to a rise in the number of devices connected to networks. These devices are becoming more accessible to a wider range of users due to development and

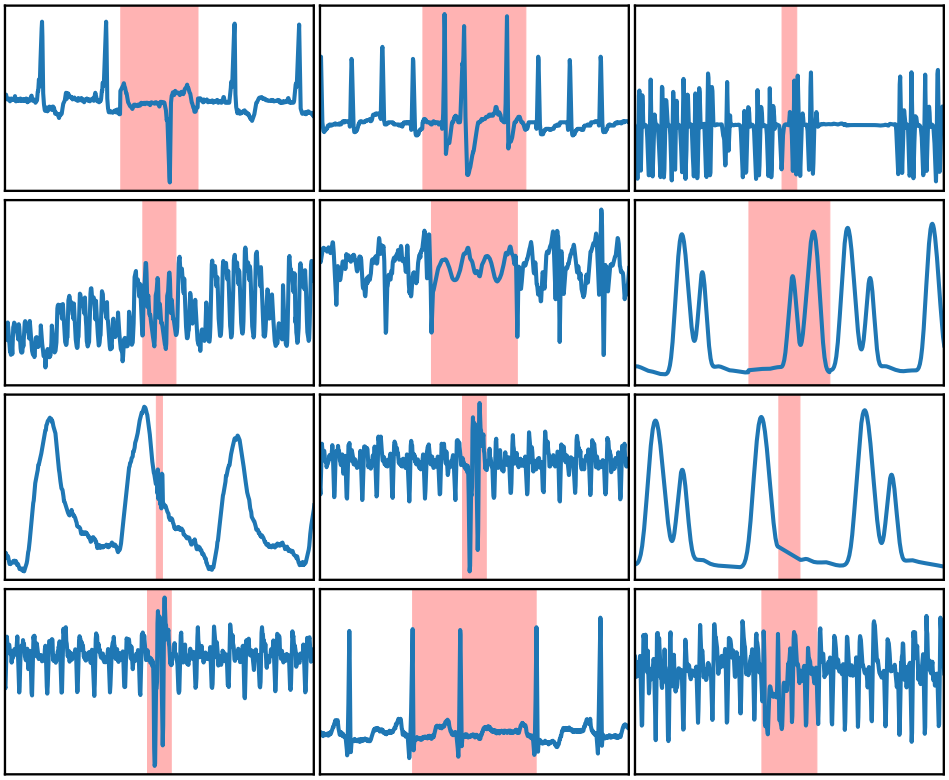


Figure 6.3: Anomaly regions colored in red for randomly sampled time series from UCR Time series anomaly archive.

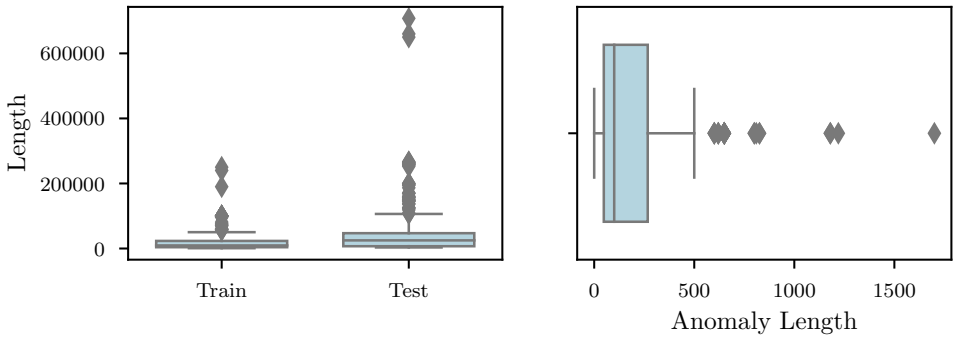


Figure 6.4: Statistics of all UCR anomaly detection datasets. (Left) Boxplot describing the length of the training and testing sets. (Right) Boxplot of anomaly regions in test split.

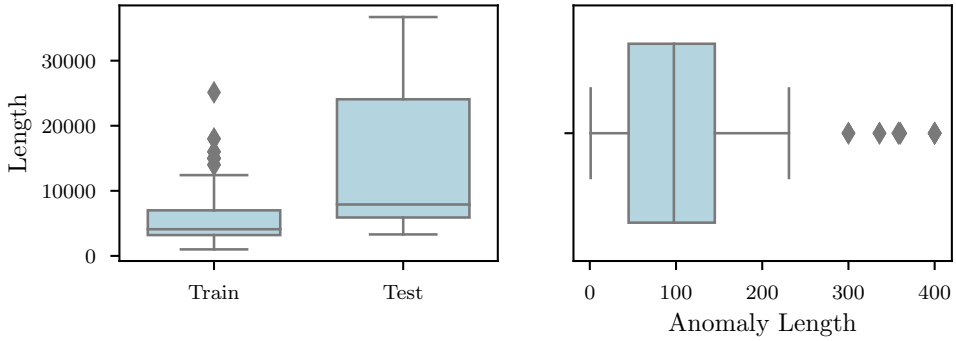


Figure 6.5: Statistics of selected UCR anomaly detection datasets. (Left) Boxplot describing the length of the training and testing sets. (Right) Boxplot of anomaly regions in test split.

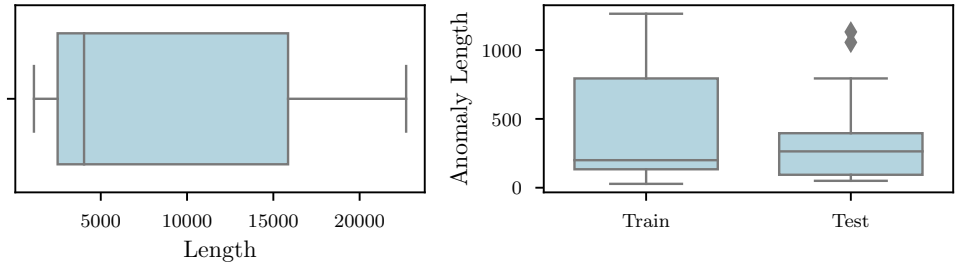


Figure 6.6: Statistics of selected NAB datasets. (Left) Boxplot describing the length of anomaly regions in the training and testing sets. (Right) Boxplot describing the length of datasets.

research. However, as machine learning, particularly deep learning, is on the rise, models are becoming larger and more costly to train and infer with. Consequently, the need for AutoML pipelines that can provide high performance and efficiency simultaneously is increasing. Moreover, to make these pipelines more accessible to small and medium-sized companies, the hardware setup was designed with their scope in mind. For this work, a computational cluster consisting of 4 processors Intel Xeon Processor E5-2650 v4 @ 2.20 GHz, for a total of 48 processor cores and 2 GPU cards Nvidia Tesla V100 16GB were used. All of the code was written in Python[4] 3.10.2.

6.3 Experiments

The structure of the experiments has been foreshadowed in Chapter 5. The main overall objective of the experiments was to thoroughly test the presented

[4]www.python.org

unsupervised metrics and AutoML approaches to create new unsupervised AutoML anomaly detection pipelines. Two main directions have been explored. Namely, AutoML using Mass-Volume and Excess-Mass metrics, and AutoML using a meta-learning approach. Under both directions, various different compositions of pipelines have been used. These are described in the following subsections. The experiments can be divided into 2 segments, where firstly the pipelines were evaluated on UCR datasets, representing semisupervised learning, followed by the same procedures ran on NAB datasets, where anomalies are present in the training data split.

For all following experiments, firstly a motivation is given, followed by a technical description of the experimental design. Finally, Results are presented in a later separate section.

6.3.1 Data preprocessing

To ensure stability and faster convergence in the training of deep learning models and other models such as SVMs, all datasets were normalized using the min-max scaling technique. This technique transforms each time point x_t of the time series X using the following equation:

$$x_t = \frac{x_t - x_{\min}}{x_{\max} - x_{\min}}$$

where $x_{\min}$ and $x_{\max}$ denote the minimum and maximum values, respectively.

6.3.2 Importance of window size

Motivation

As was mentioned before in section 4.4, selecting the appropriate window size is crucial for algorithms to achieve good results. Because of this, an automated window size selection algorithm should be considered the first part of an automated pipeline. In this thesis, AutoPeriod was chosen as an automated window size selection algorithm as it has been shown to perform well in the task of anomaly detection in [72]. One of the collections of datasets used in their experiments has been the UCR archive that is used in this thesis as well. Before using the algorithm for the rest of the experiments observation of how it performs relative to all possible values in a selected range of values.

Design

To obtain the window size, AutoPeriod was used on training splits of 10 random datasets from training UCR datasets. Not all datasets contain a period. Therefore, for datasets that AutoPeriod could not find a periodicity, a default value of 16 has been used. As to not run into computationally intractable

problems, the periods spanning more than 512 time points were clipped to 512, as the time and space complexity of several algorithms is dependent on window size. Models of k-NN, LOF, and Isolation Forest were fitted with default parameters on the training splits using window size from 16 to 512 with a step of 16. Models were evaluated on test splits using F1-score and F1-score under Point adjusted evaluation scheme.

6.3.3 Design of pipelines

The upcoming sections cover the selection of anomaly detection models, the composition of methods that form automated pipelines, and the evaluation metrics employed. Firstly, the motivation and thought process behind the creation of these pipelines are outlined, followed by the technical details of the design and the training and evaluation process.

Models

This subsection introduces models selected for the following experiments. The selected models are divided into four types based on the reason for selection as follows:

- k-NN, LOF, OCSVM, and IF are well-known, simple yet effective methods for detecting anomalies that are commonly used as despite the simplicity they keep good performance in practice.
- VAE and LSTM EncDec could be considered as basic deep learning models for anomaly detection.
- TranAD and GTA are state-of-the-art deep learning models using the latest architecture styles of transformers and graph neural networks.
- DWTMLEAD is a distribution method that was found to be the most effective w.r.t. performance in large evaluation survey [7].

This selection covers models across the taxonomy presented in section 2.2. These are models that performed well in the literature and are reasonably computationally expensive to train and make predictions with. With the selection, it is believed by us that it covers the currently available spectrum of anomaly detection methods applicable to our setting.

Non-changing parts of a pipeline

Few of the parts of a pipeline composition stayed the same for all pipelines. Data were always preprocessed using min-max normalization to a range from 0 to 1. Next, where needed, a size of a window was determined using the AutoPeriod method. In case AutoPeriod did not find any seasonality in the data a default size of 16 was used.

6.3.4 Baseline

Motivation

To not repeat mistakes of some works pointed out in [51], a baseline is presented for the reason of comparison. Having a baseline is essential because it provides a reference point for comparing and evaluating the performance of new methods. Without a baseline, it is challenging to determine whether a new method is actually an improvement.

Design

As a baseline, all selected methods were fitted and evaluated on a training split of testing datasets with default parameters. And then to evaluate the baseline, F1-score, F1-score under point adjusted evaluation scheme, AUC-ROC, AUC-PR metrics were used.

Although four metrics were used for baseline, AUC-ROC and F1-score under PA evaluation scheme are not used in further experiments. For the reason that AUC-PR better suits the task of anomaly detection and F1-score under PA evaluation scheme can severely overestimate performance as shown in [51], where randomly generated anomaly score from uniform distribution achieved better PA F1-score than some of the models that were regarded to as state-of-the-art.

6.3.5 Unsupervised metric

Motivation

Paper [52] states that no unsupervised evaluation metric for anomaly detection exists. Conversely, in [56] Goix et al. conclude they were able to recover which algorithm is better w.r.t. AUC-ROC and AUC-PR than the other using EM and MV metrics in 80% of cases. With this experiment, the main objective is to answer the questions Q 1.1 and Q 1.2.

Design

The MV and EM metrics were computed using a methodology proposed in [56], which only supports methods specifically designed to operate on subsequences. Consequently, only k-NN, LOF, OCSVM, and IF were chosen because the Monte Carlo (MC) method is used for integration within the metrics. MC generates samples to approximate the integral and this is exponentially more expensive with increasing dimensions. Hence why, there is a limitation to using only 5 features. Thus, training models like VAE, LSTM, etc. on only 5 randomly selected features of subsequences would not yield meaningful results.

The selected methods were trained using the previously mentioned approach. Each method was trained 30 times on 5 distinct randomly selected features of subsequences, and the final MV and EM values were obtained by calculating the average MV and EM values across the 30 runs.

The hyperparameters were fine-tuned based on the metric values, using a Tree-structured Parzen estimator to sample hyperparameters. However, since these methods cannot be trained iteratively, a pruning algorithm was not utilized. A computational budget of 20 minutes or 30 trials was allocated for each method on every dataset. Once either of these limits was reached, another training procedure would begin. The evaluation was performed using the F1-score and AUC-PR, and in the experiment, this combination is denoted as **US**.

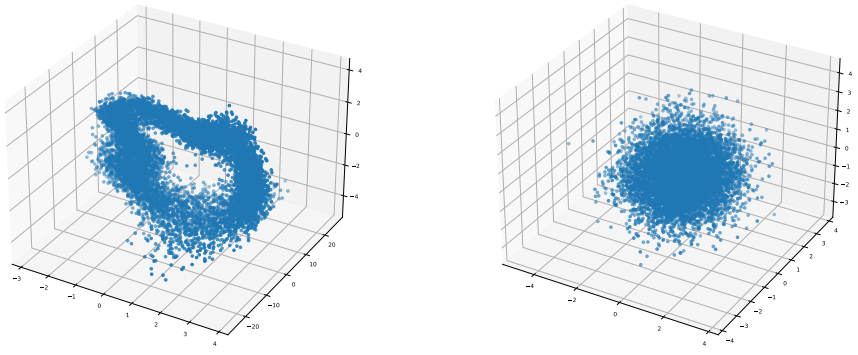
6.3.6 Extended unsupervised metric

Motivation

Given the two main drawbacks of the unsupervised metrics: the randomness associated with feature sampling and the restriction on the types of models that can be employed. Possible solutions are focused on in the following experiments.

First, the focus was shifted to the randomness aspect. By randomly sampling a small number of features, the metrics search for smaller dimension space, in which the anomalous points would still follow the metrics' assumption of an anomaly being present in tails and sparse regions of an underlying probability distribution. As this procedure is repeated multiple times, it is expected that anomalous points are more probable to end up in those regions more frequently than normal points. However, training the methods that many times can be expensive. With this in mind, a dimensionality reduction method could be used in place of randomly selected features. If the method could learn a latent space of the subsequences without losing any information on anomalies, it would be expected that the anomaly would follow the same assumption in the learned latent space.

The usage of a dimensionality reduction method could potentially solve the first limitation. Similarly, the second limitation could also be addressed using a comparable approach. If an AutoEncoder could be trained to accurately reconstruct the subsequences, it is plausible to assume that the anomaly information could be captured in the learned latent space. The proposal is to train the anomaly detection models using complete subsequences, which enables the use of various models, and then measure the metric values in the latent space. Addressing the second issue is question Q 1.3.



(a) Arbitrary latent space generated by an Autoencoder.

(b) Latent space forced to be close to a unit Gaussian.

Figure 6.7: Figure shows the concept of controlling the shape of the learned latent space using a Variational Autoencoder (VAE) by enforcing it to conform to a unit Gaussian distribution, as opposed to an AutoEncoder.

Additionally, it could be advantageous to have control over the shape of the learned latent space. Monte Carlo integration within the unsupervised metrics samples from a multivariate uniform distribution where the bounds are set as a hypercube enclosing the data points. For example, if the learned latent space forms a manifold of a torus many of the drawn samples will not cover the data points and therefore be of no use. One possible solution is to replace an AutoEncoder with a Variational Autoencoder, where the KL term in the VAE loss function enforces the latent space to conform to a unit Gaussian distribution. Uniformly sampling in a hypercube enclosing the hypersphere formed by a multivariate unit Gaussian could be more efficient at covering the data points and therefore to better estimate the metrics. Figure 6.7 illustrates this concept.

Design

Three different pipeline compositions were tested in order to conduct experiments incorporating the previous ideas. These compositions are described in the following list:

- **EUS** - First, an AutoEncoder was trained for each dataset. Next, k-NN, LOF, OCSVM, and IF were fitted on a 5-dimensional latent space learned by AE, and their hyperparameters were optimized using MV and EM metrics. For a sampling of hyperparameters, a Tree-structured parzen estimator was applied. Additionally, a computational budget of 20 minutes or 30 trials was set for each method on each dataset.

Whenever either one was depleted, another training procedure would begin. Finally, F1-score and AUC-PR were used for evaluation.

- **EUS-AE** - Similarly to the previous composition AE was firstly trained for each dataset. Then, all selected models except for DWTMLEAD were trained on full-length subsequences. Next, AE was used to obtain a latent space of a dataset to uniformly sample points in that space. After that, AE was used to decode the samples into the original space, where these samples were scored using the learned anomaly detection model. Having scored samples, the metrics could be computed. For a sampling of hyperparameters, Tree-structured parzen estimator was applied and where possible Successive Halving was utilized to prune unpromising configurations. Additionally, a computational budget of 20 minutes or 30 trials was set for methods trained on CPU and 5 minutes or 50 trials was set for deep learning models that were trained on GPUs on each dataset. As deep learning models can utilize the pruning more trials were allowed. Whenever either one was depleted, another training procedure would begin. Finally, F1-score and AUC-PR were used for evaluation.

- **EUS-VAE** - This composition is identical to EUS-AE with the only difference being a replacement of AE with VAE to control the shape of the learned latent space.

6.3.7 Meta-learning approach

Motivation

Even though previous experiments were done with the hope of finding such metric, we acknowledge the findings of [52], that there is currently no unsupervised metric available for selecting the models. Therefore, new approaches had to be developed. The MetaOD approach, based on meta-learning [68], leverages information about the performance of models trained in a supervised setting, along with meta features that capture the dataset characteristics. This information is used to recommend a model that would likely perform the best on a new, unseen dataset, similar to how a recommendation system works. Question Q 2.1, whether the performance carries over to the time series domain is one of the objectives of this experiment. It is also believed that features better suited for time series could be found and will be the subject of the experiments.

Design

The first step in the approach was to train and optimize all models in a supervised setting using the AUC-PR metric on the 100 training UCR datasets

with the same time constraints as in previous experiments. The resulting performance information was used to create a performance matrix, which was then subjected to matrix factorization into the dataset matrix and the model matrix. The first matrix was initialized with 200 computed dataset meta-features, and matrix factorization was optimized for the discounted cumulative gain using stochastic gradient descent, as the ranking of models is more important than the reconstruction error. A grid search for the learning rate and dimension of both latent spaces was performed. As a result, a latent space of models and a regressor for the latent space of datasets were obtained. The performance of the recommender was then evaluated on testing datasets, using the performance obtained in the unsupervised **EUS-AE** experiment to avoid training the predicted models again. This approach is referred to as **META**, but it should be noted that the meta features computed in the original experiment were not specialized for time series data, as the approach was originally designed for outlier detection. To address this, an additional experiment was conducted using over 700 time series specific features extracted by the tsfresh[5] package. This approach is referred to as **META-ts** and this experiment sought to answer question Q 2.2. To conduct a more thorough comparison, another two pipeline compositions, referred to as **META_base** and **META-ts_base**, were tested, where recommended models obtained performance from the baseline. This simulates a recommendation process followed by training using the default hyperparameters.

6.4 Summary of pipelines

Brief summary of tested pipelines is given in this section. Presented pipelines and their differences can be mostly summarized in four categories: Hyperparameters optimization (HPO), model selection (MS), models used, and other differences.

- Non-changing parts of the pipelines:
  - Data normalization using min-max scaling
  - AutoPeriod used for window size selection
- Baseline
  - HPO and MS
    - Default HPs
  - Models
    - k-NN, LOF, OCSVM, IF, VAE, LSTM EncDec, TranAD, GTA, and DWTMLEAD

[5]https://tsfresh.com/

- **US, EUS, EUS-AE/VAE**
  - HPO and MS
    - Optimization using MV and EM metrics
    - Tree-structured Parzen Estimator used for HP sampling
    - Successive Halving utilized when suitable
  - Models
    - **US**, **EUS** - k-NN, LOF, OCSVM, and IF
    - **EUS-AE/VAE** - k-NN, LOF, OCSVM, IF, LSTM EncDec, VAE, and TranAD
  - Others
    - **US** uses methodology for computing EM/MV metrics from [56]. Five randomly selected features of subsequences are used to fit models and to compute metrics value. This process is repeated 30 times and metric values are averaged to obtain the final value.
    - **EUS** replaces random sampling of features from subsequences used in US by employing AutoEncoder as a dimensionality reduction technique for each dataset
    - **EUS-AE/VAE** models are trained on entire subsequences. Then AE/VAE trained for each dataset is used to obtain the latent space of the given dataset. Uniform samples are drawn from a hypercube enclosing the latent space and are decoded to the original space of the dataset. Next, the samples are scored using the fitted model in the original space. Lastly, the metric value is computed on the samples in the latent space scored in the original space.
- **Meta-learning**
  - HPO
    - **META_base**, **META-ts_base** - None
    - **META**, **META-ts** - using EM/MV metrics under the EUS-AE approach, TPE and SH used when appropriate
  - MS
    - **META_base**, **META** - Recommendation based on original meta features
    - **META-ts_base**, **META-ts** - Recommendation based on time series specific meta features
  - Models - k-NN, LOF, OCSVM, IF, VAE, LSTM EncDec, TranAD, GTA, and DWTMLEAD

6.5 Results

The Results section is divided into three subsections: Results of the importance of window size experiment, semi-supervised experiments conducted on UCR datasets, and unsupervised experiments conducted on NAB datasets. The design of the latter two experiments is the same.

Importance of window size

First, the verification of the statement that selecting an optimal window size is critical for algorithms was attempted. The significance of selecting the correct window size can be seen in figure 6.8, where the performance varies considerably across the range of window sizes. For some datasets, the algorithms only show optimal performance within a small range of tested window sizes. For example, on the *006DISTORTEDCIMIS44AirTemperature2* dataset, k-NN achieved an almost perfect F1-score of 1 with a window size < 100, but the performance dropped by more than 80% when trained on subsequences of length > 150.

The main objective of this experiment was to answer the question Q 0, how does AutoPeriod perform as an unsupervised window size selection algorithm? Before a comparison of the performance is presented, a disclaimer is given that it is not fair to compare the AutoPeriod against the evaluations under supervision as the method was designed to select window sizes with no ground truth information available. Despite the unfairness, the experiment is provided to compare how does AutoPeriod do against the supervised selection of the window. The models with AutoPeriod detected seasonality achieved a mean F1-score of 0.44 which is around 19% less than the mean of the best possible F1-scores equal to 0.54. Relative performance w.r.t. best possible F1-scores across all 3 models and 10 datasets was 75.6% when using AutoPeriod. For a more fair comparison, the experiments were run again with window sizes set to 16, 32, 64, and 128. AutoPeriod achieved better results than these default values by 29%, 11%, 2%, and 14% in the same order. The little gain of 2% when setting the windows to a size of 64 could be explained by a mean of AutoPeriod periods on the datasets being 68.7. The gain over other values supports the need for correct size selection.

AutoPeriod is used to find seasonality in the data by the frequency with the largest magnitude. It is possible that anomalies could stretch across one period. Therefore, giving the methods more context could help discover anomalous patterns. A question immediately arose if the performance of the models could be significantly improved by multiplying the obtained period with some constant. The results for this question can be seen in figure 6.9. When multiplying all the periods by 1.59 a gain of 9% was observed in the mean of

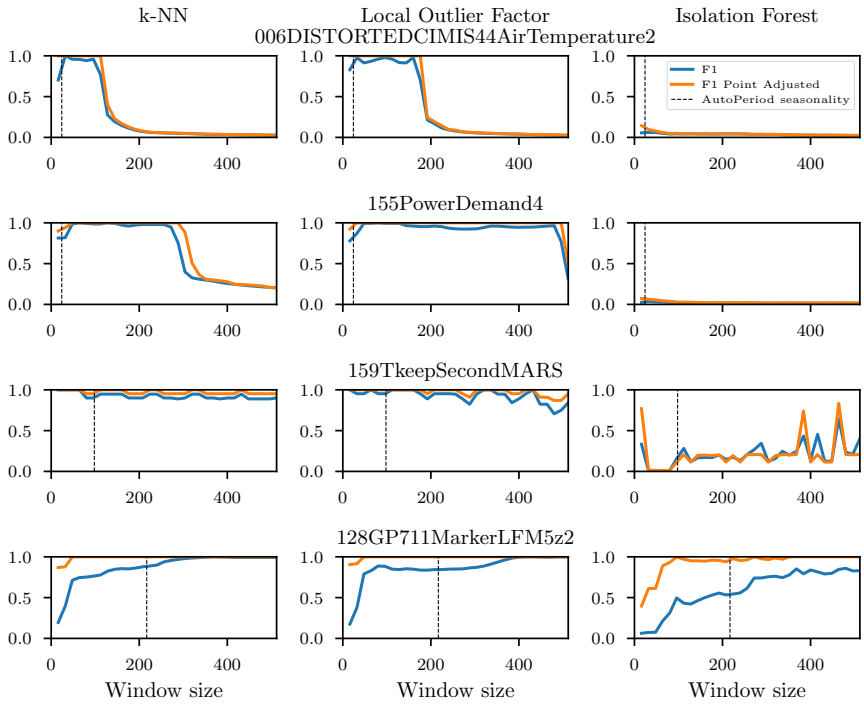


Figure 6.8: The performance of models changes significantly w.r.t. size of windows. The rows represent chosen UCR datasets and the columns represent fitted methods. Performance was measured using F1-score and Point Adjusted F1-Score. From the plots, it can be clearly seen why it is important to dedicate time to correctly choosing window size. The dotted lines represent the window size suggested by the AutoPeriod method. The datasets for the plots were chosen as an example.

F1-scores from 0.44 to 0.48. Nevertheless, no significant conclusion can be drawn from the results, as the number of datasets used in this experiment was small, and further investigation and testing would be needed. For the following experiments based on these results and the literature reviewed, AutoPeriod is used to select the window size, and the multiplying constant is set to 1.

UCR

Firstly, a review of baseline performance is presented. The results can be seen in figure 6.10. There are a few key takeaways from the results. LOF and k-NN both perform significantly better than other methods. Two suggestions for this phenomenon could be the methods not being very sensitive to hyperparameter settings and that the methods benefit from no contamination in the training split. DWTMLEAD performs the best under Point Adjusted

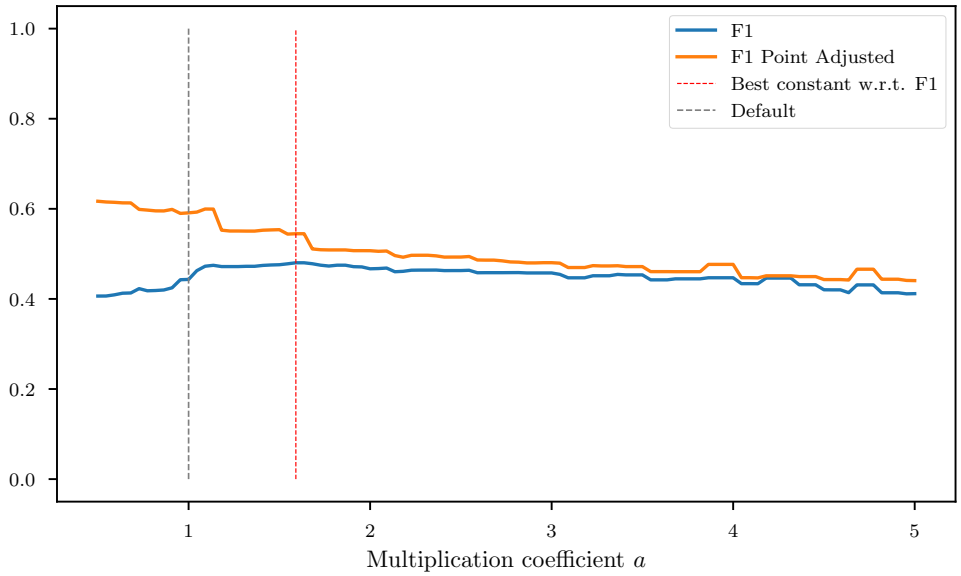


Figure 6.9: Effect of multiplying obtained periods on the performance of models. When multiplying by the constant of 1.59 (red line) the models gain 9% in F1-score. Nevertheless, no significant conclusion can be drawn.

evaluation scheme but falls behind the previously mentioned methods in other metrics. It seems that DWTMLEAD is superior to others in marking only a couple of points in anomalous sequence. It is no surprise deep learning models performed poorly as they were trained only with default hyperparameters.

To answer questions Q 1.1 and Q 1.2, the results of the US pipeline were compared to the baseline. Figure 6.11 shows the comparison between the two. The performance of k-NN and LOF did not show any significant changes. On the other hand, OCSVM showed the largest improvement, with a mean performance increase of around 15% in both F1 and AUC-PR. In contrast, IF showed the highest decrease in performance out of all four methods. However, the results do not allow for any significant general conclusions to be drawn.

The authors of [56] concluded that it was possible to recover the best-performing method in terms of AUC-ROC and AUC-PR in 80% of cases using EM and MV metrics. However, the same results could not be obtained in our experiments. Out of the 50 datasets tested, the best model in terms of AUC-PR could only be recovered 16 times with the EM metric and 17 times with the MV metric, resulting in a recovery rate of 32% and 34%, respectively. Figure 6.12 shows the performance of models selected using these metrics compared to the optimal ones. Based on these results, it can be concluded that the metrics are not reliable and should not be used for model selection.

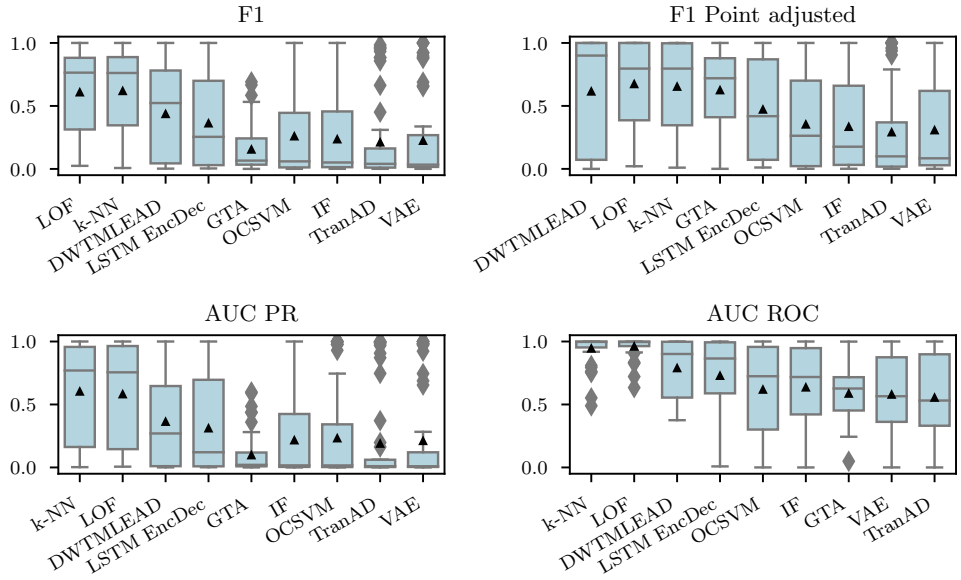


Figure 6.10: UCR: Baseline performance of selected anomaly detection models with default hyperparameters. A horizontal line inside a boxplot marks a median and the mean is marked with a black triangle. The boxplots are sorted by the median. LOF and k-NN perform significantly better across all metrics. DWTMLEAD performs the best under Point Adjusted scheme but not as well in other metrics. GTA being the latest SOTA performs only under the PA scheme. This could be because it was developed and tested using this scheme.

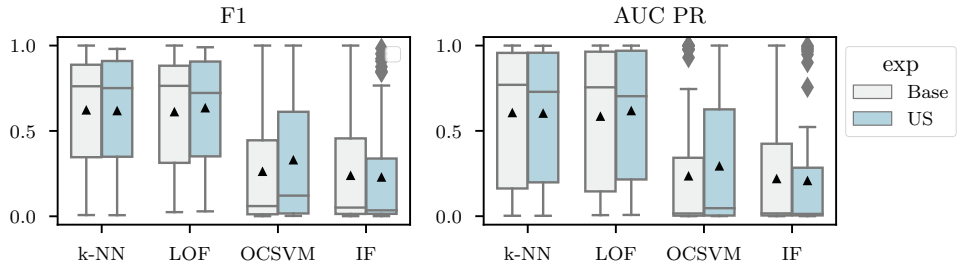


Figure 6.11: UCR: No significant differences were observed comparing baseline against models trained in the US pipeline. The performance of models in the US pipeline was achieved using the MV metric. OCSVM showed the largest improvement, with a mean performance increase of around 15% in both F1 and AUC-PR. In contrast, IF showed the highest decrease in performance out of all four methods.

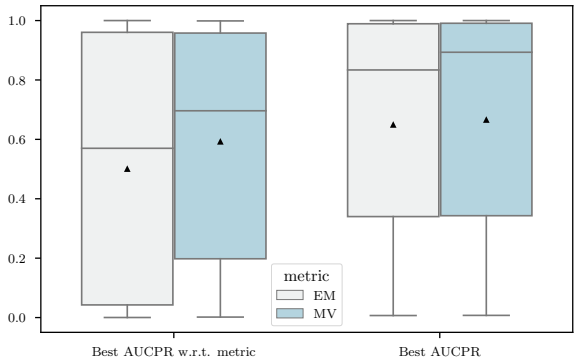


Figure 6.12: UCR: Models cannot be reliably selected using the unsupervised metrics. The performances that could be achieved with perfect model selection are considerably higher than those recovered by the metrics. The performances were achieved by hyperparameter optimization using unsupervised metrics.

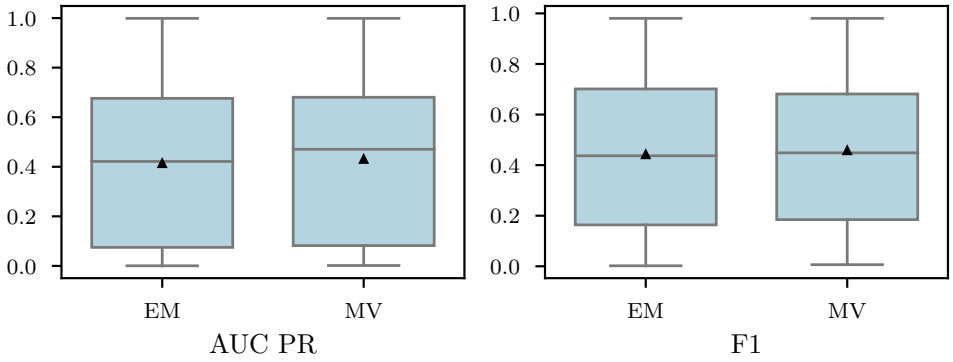


Figure 6.13: UCR: While no significant difference was observed between the EM and MV metrics, an insignificant performance gain was noticed for the MV metric. For simplicity and given the similarity of the results, only the MV metric was used in the subsequent experiments.

To evaluate whether the EM or MV metric performs better for hyperparameter optimization, a comparison between the two was conducted before addressing question Q 1.3. Figure 6.13 shows that no significant conclusion can be drawn from the comparison. However, a small improvement is observed for the MV metric over the EM metric in terms of AUC-PR. For simplicity and given their similarity, only the MV metric is used in the following experiments.

The objective of question Q 1.3 was to see if possible improvements to MV and EM metrics could be made to improve performance and usability. Based on the results presented in figure 6.14, it can be concluded that placing an AutoEn-

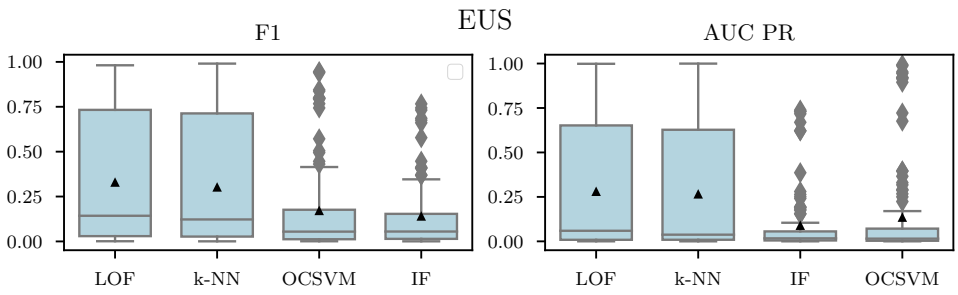


Figure 6.14: UCR: Severe decrease in performance was observed across all models when using AutoEncoder instead of randomly sampling features.

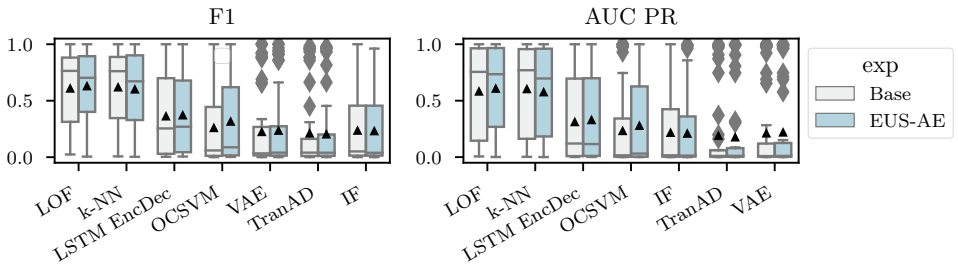


Figure 6.15: UCR: The EUS-AE and EUS-VAE didn't perform better than the baseline but offered advantages over the US pipeline. The gained usability in the ability to allow the use of other models didn't improve performance. EUS-AE is chosen as representative as EUS-VAE performed very similarly.

coder instead of randomly sampling features did not improve the performance of the models. Instead, a severe decrease in performance was observed across all models. The idea of why that could be is an inability to map anomalies to latent space as was expected.

The purpose of the EUS-AE and EUS-VAE pipelines was to improve usability by allowing all models that use subsequences to be optimized using the metrics. Figure 6.15 shows a comparison of EUS-AE and the baseline. The performance does not significantly differ from the performance observed in the US pipeline. The advantage over the US pipeline is that there is no need to compute the metrics under the sampling methodology, but it comes at the cost of training an AutoEncoder over each dataset. The ability to use other types of models could be seen as a benefit, but optimizing them did not bring any relevant performance gain. Consequently, controlling the distribution of the latent space using VAE did not lead to any improvement compared to the use of a AE.

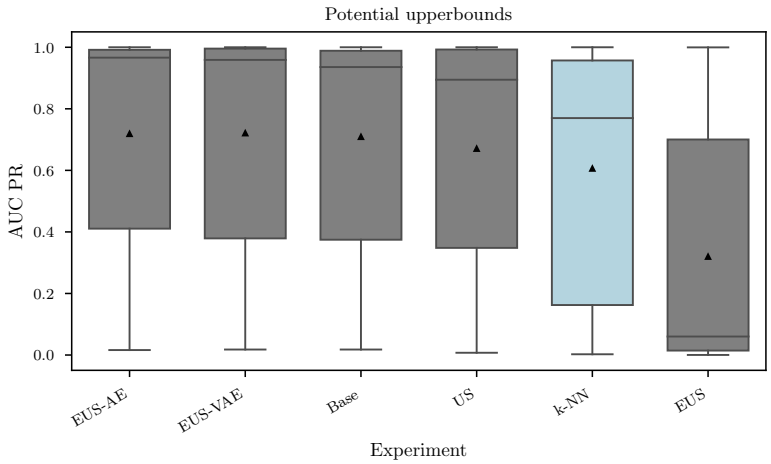


Figure 6.16: UCR: Figure compares the performance upper bounds of different pipelines with perfect model selection (grey colored boxes). k-NN alone could achieve a median AUC-PR of 0.8. The US and EUS pipelines are not recommended. The slight improvements of EUS-AE and EUS-VAE could be explained by a slight improvement in OCSVM performance.

The potential performance upper bound of pipelines that could be achieved if a perfect model selection was available with EM and MV metrics used for hyperparameters optimization was examined before moving to meta-learning experiments, and the results are presented in figure 6.16. The figure shows that even with default parameters, at least half of the datasets could almost achieve AUC-PR close to 1. Among the selected methods, k-NN is the best-performing, and using this method alone with default parameters would produce a median AUC-PR of approximately 0.8. It can be drawn from the figure that the US and EUS pipelines should not be considered, even if a perfect model selection was possible.

In this section, a discussion regarding the results of meta-learning model selection methods is presented. Four different pipelines were tested. Surprisingly, the recommendation model with time series meta-features performed worse than the recommendation model with the meta-features originally used by the authors for non-time series-related problems. This finding provides an answer to question Q 2.2, as no improvement was achieved by focusing on the time series domain. As expected, training the recommended models with default hyperparameters resulted in the worst performance. Nevertheless, none of the pipelines could outperform the baseline of using k-NN for every dataset, answering the question Q 2.1. Results can be seen in figure 6.17.

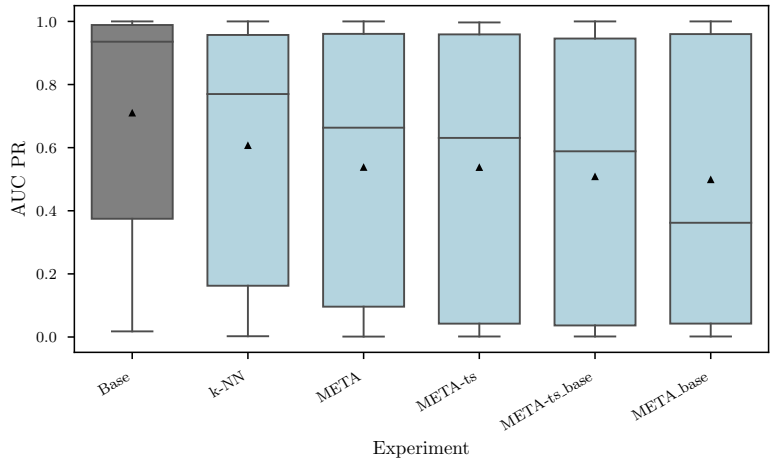


Figure 6.17: UCR: Comparison of performance for different meta-learning model selection pipelines, showing that none of the tested pipelines outperform the baseline of using k-NN for every dataset. Grey-colored boxplot represents the baseline upper bound if a perfect model selection was available.

NAB

NAB datasets were selectively chosen to simulate a fully unsupervised setting where anomalies were present in training splits of all datasets. The design of the pipelines remained the same. Except for meta-learning experiments where cross-validation was used to train the recommendation model as the number of NAB datasets is small. Lastly, the results are reported in a similar fashion to the results obtained on UCR datasets.

Before reviewing the results, the labeling process of the NAB datasets is presented as it affects the results of the experiments. To obtain ground truth labels for NAB datasets a group of human annotators labeled the datasets according to a given guideline. As a result of the process of aggregating the labels, datasets contain single-point anomalies (anomalies of length 1). These single-point anomalies were then converted into anomaly windows. The reasons are: anomalous data often occur over time and anomaly windows allow models to not be penalized for slightly early or late detection. The width of an anomaly window was calculated so that the total amount of window length in a single dataset is 10% of the dataset length.

Initially, a review of baseline performance is presented. The results can be seen in figure 6.18. The anomaly window labels added to NAB datasets lead to significant improvements when evaluating using the PA evaluation scheme. GTA model achieved near-perfect performance across all datasets. However, it ranked the worst when using other evaluation metrics. GTA is the lat-

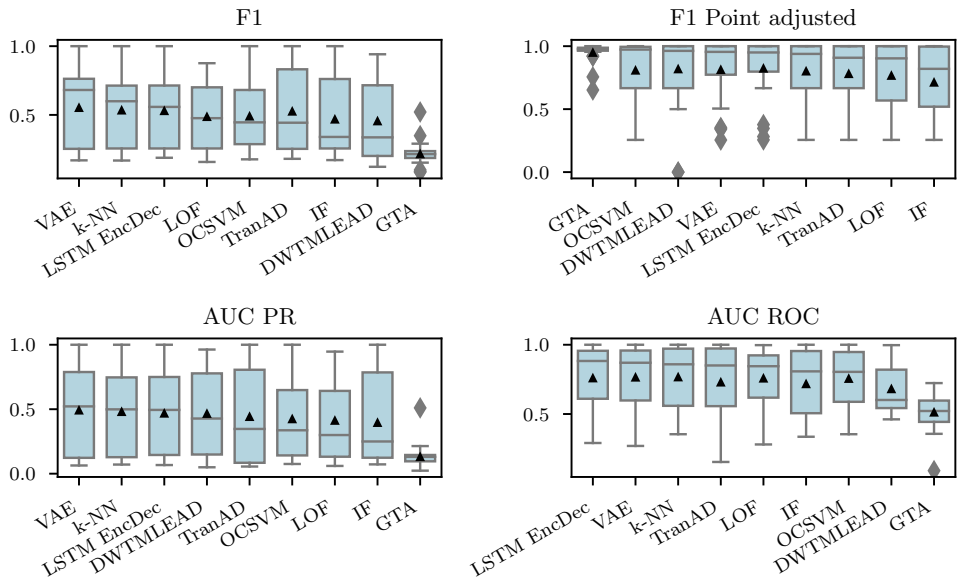


Figure 6.18: NAB: Baseline performance of selected anomaly detection models with default hyperparameters. A horizontal line inside a boxplot marks a median and the mean is marked with a black triangle. The boxplots are sorted by the median. Results of the GTA model seem to illustrate the influence of developing a model under the recent settings where slightly early or late detection is allowed with a combination of a setting where only a fraction of time points is needed to be marked as anomalous.

est SOTA method and the influence of being developed under PA evaluation scheme could be the reason for the results. As it stands, it seems that GTA excels in slightly early or late detection and only labels a small fraction of time points within an anomaly window. Compared to UCR results, k-NN and LOF did not outperform other models, yet they seem to be among the better ones. The performance of these models is expected to lower with increasing contamination rate.

In figure 6.19 a summary of the performances of all pipelines using the MV and EM metrics can be seen. Similarly to UCR results the metrics could not be used for model selection (Q 1.2). The recovery rate of the best model w.r.t. AUC-PR and F1 using the unsupervised metrics was around 17%, being even lower than in UCR results. Even if the perfect model selection was possible, none of the pipelines outperformed the baseline (Q 1.1). As a result of this, the unsupervised metrics can not be deemed as metrics for hyperparameter optimization. Extending the metrics in EUS-AE and EUS-VAE pipelines led to insignificant performance gains over the US but not outperforming the baseline (Q 1.3).

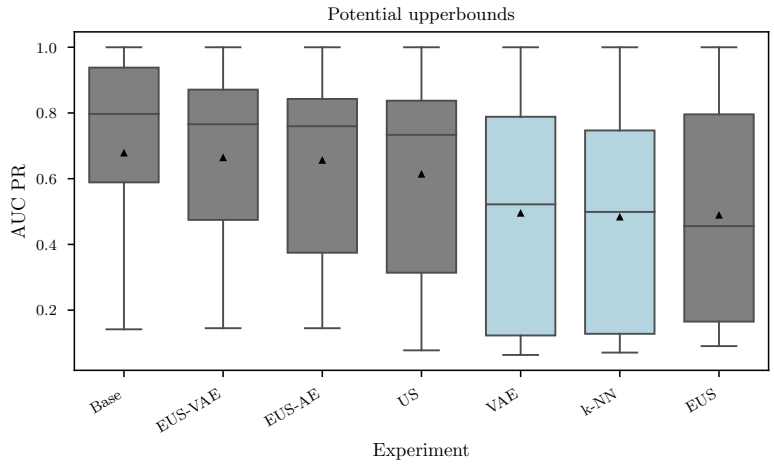


Figure 6.19: NAB: Figure compares the performance upper bounds of different pipelines with perfect model selection (grey colored boxes). None of the pipelines led to an improvement over the baseline. Suggesting that the unsupervised metrics cannot be reliably used for hyperparameter optimization and model selection.

As mentioned before, a cross-validation had to be used as the number of selected NAB datasets was 25 which is considerably low. The datasets were split into 5 folds. The training of the recommendation system was done on 4 folds and 1 remaining fold was used for prediction. This was repeated for each individual fold.

Figure 6.20 shows a comparison of meta-learning pipelines. Using the time series-specific meta features did not lead to any improvement, on the contrary, it has led to a severe decrease in performance (Q 2.2). Pipelines META and META_base achieved comparable performance to simply using VAE or k-NN with default parameters at a many-folds larger computational expense which is not considered satisfactory (Q 2.1).

6.6 Discussion

Two different AutoML approaches and many variations of pipeline compositions were thoroughly tested in two different learning settings.

Firstly, experiments using AutoPeriod as a window size selection algorithm were conducted. According to the literature, AutoPeriod achieves the best performance in selecting a window size for unsupervised anomaly detection tasks. In our experiments, a comparison was made between AutoPeriod and

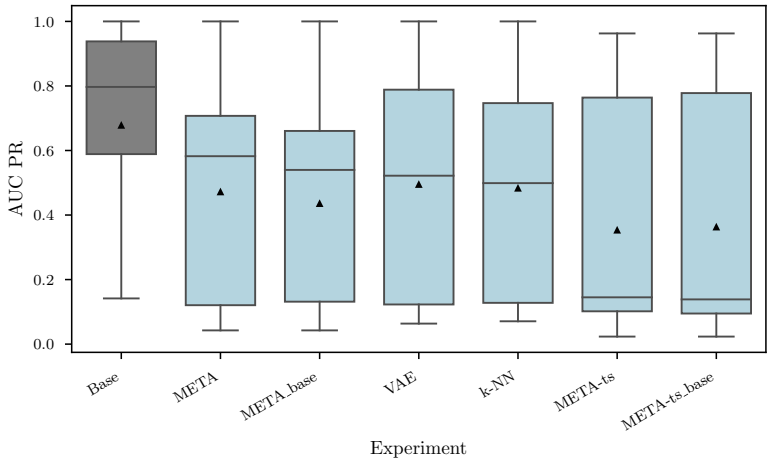


Figure 6.20: NAB: Figure shows a comparison of performance for different meta-learning model selection pipelines. Grey-colored boxplot represents the baseline upper bound if a perfect model selection was employed. META pipelines using the original meta features performed similarly to baselines of VAE and k-NN which were the two best-performing models in the baseline experiment. Using time series-specific meta features led to significantly worse performance.

the supervised learning procedure. A mean F1-score of 0.44 was achieved by the models using AutoPeriod-sized windows, which is approximately 19% lower than the mean of the best possible F1-scores of 0.54. This indicates the potential for improvement. AutoPeriod's task is to find seasonality in provided time series and a question arose if using windows larger than one period would on average improve models' performance. For this reason, a test was conducted on whether a coefficient for multiplying the period exists. With a coefficient equal to around 1.5 an improvement of 10% was seen. The mean f1-score of models improved to 0.48. The test was carried out on 10 datasets using 3 models. With such a small sample, any significant conclusion could not be drawn, and further investigation is needed.

Next, the proposed pipelines were tested in a semisupervised setting on UCR datasets. Experiments conducted using the unsupervised metrics of EM and MV have shown that they cannot be reliably used for either model selection or hyperparameter optimization. None of the pipelines could achieve better performance than using the k-NN method with default parameters. However, a possible explanation for this could be that the performance of the k-NN together with LOF was significantly better than other models and simply choosing one of the two models would be highly probable to result in a good performance. The reason for such performance of the two algorithms could

be possibly explained by the UCR datasets being semisupervised and by the fact that their contamination is extremely low. Which could favor them.

Next, placing an AutoEncoder instead of randomly sampling features within the EM and MV metrics in **EUS** pipeline significantly worsened the performance of the models, possibly indicating an inability to map anomalies to latent space which is the opposite of what was expected. Alternatively, the dimension of the latent space is likely too small to preserve all the information. If the latter is true, then Monte Carlo integration would be a limitation of the metrics as it requires an exponentially growing number of samples to allow for the same relative error of approximation. With higher dimensions, this becomes computationally infeasible. Additionally, replacing Monte Carlo is a difficult problem, given that not much is known about the properties of the learned latent space or about the manifold where the points lie when using AutoEncoder. Thorough research has been done on computing volume in high dimensional convex bodies [81]. To this end, VAE forcing latent space to conform to multivariate unit Gaussian would seem like a plausible solution as successfully trained VAE would map data points into a hypersphere. However, when using a scoring function of a model to select points in this latent space there are again no guarantees of the shape of the body of the points. A potential solution could be finding a convex hull around the points and computing its volume. However, this would assume that all the points lie in one cluster for the approximation to be close to real volume. Overall, these are pure speculations and a more detailed examination is needed. Leaving this an open question.

In this thesis, VAE has been used with Monte Carlo integration under the assumption that anomalous points would be mapped to points lying in the sparse regions (tails) of the multivariate unit Gaussian distribution, and a scoring function optimizing the unsupervised metrics would score these points more anomalous. On the contrary, if this assumption would hold a simple scoring function of distance from the center of the distribution would perform well.

Subsequently, **EUS-AE** and **EUS-VAE** approaches allowed for a wider range of models to be used but no significant improvement over the baseline was recorded. Additionally, using VAE instead of AE performed similarly. Seemingly invalidating previous assumptions. Meta-learning pipelines also performed worse than the baseline of using only k-NN. In addition, all meta-learning pipelines performed similarly. To conclude, improvement was not observed even when using time series-specific meta features.

Lastly, the unsupervised setting was created and tested by specifically choosing NAB datasets so that anomalies are present in both training and testing splits. The NAB datasets are labeled so that early or late detection of anomalies is

allowed. This led to a significant increase in F1-score under the PA evaluation scheme across all models. GTA is a model that was near perfect for all the NAB datasets when evaluated using the PA scheme but was the worst when other metrics were used. Which is a result of being developed with the PA scheme in mind. Nevertheless, none of the pipelines outperformed the baseline using unsupervised metrics, further indicating that these metrics cannot be reliably used for hyperparameter optimization or model selection. Meta-learning pipelines using time series-specific meta-features did not lead to any improvement and even resulted in a severe decrease in performance. When utilizing the original features performance comparable to the baseline of VAE and k-NN was seen but at a much higher computational cost. The failure of time series-specific meta features in the improvement of the performance could be attributed to the loss of the mentioned landmark meta features. These would be the features obtained from easy-to-construct outlier detection models. These extracted features are said to be more problem-specific [64]. Whether these landmark features in combination with time series-specific features would lead to a gain in performance is an open question.

Chapter 7

Conclusion

In this thesis, an effort has been made to address the challenge of anomaly detection in univariate time series data using AutoML techniques. In particular, the emphasis was placed on creating new pipelines that can handle both semisupervised and unsupervised settings. Currently, there are no reliable approaches to solve the task.

Before the main experiments were conducted, a side experiment on an unsupervised window size selection was carried out. Tested AutoPeriod method [73] showed good results w.r.t. supervised selection but the results suggest that improvements are still possible.

Mass-Volume [53] and Excess-Mass [54] metrics were thoroughly tested and extended to allow a wider range of models to be used. However, the experiments showed their inability to be used in hyperparameter optimization and model selection. Next, the meta-learning approach for unsupervised model selection introduced in [68] was tested in the time series domain using time series-specific meta features. Nevertheless, no significant performance gain has been measured.

The most important direction for future work to solve the problem is believed by us to be evaluation. For the reason that many various methodologies of evaluation are currently being used. Resulting in works with incompatible results. A single general procedure for evaluation is needed. Also, a general guideline for labeling datasets would help, as some datasets allow for detection near the anomalies, while others do not.

As seen in the results the nine selected methods could solve at least half of the datasets nearly perfectly with default hyperparameters if given a perfect model selection procedure. Indicating that model selection is more important than hyperparameter optimization and should be prioritized.

In conclusion, the results of this thesis offer new insights into available methods. The task of leveraging AutoML for unsupervised anomaly detection in time series is a crucial area that demands continuous effort in research. The current state of evaluation leaves a feeling of being fragmented and incomplete, leaving room for innovative ideas and new perspectives. Moving towards a more comprehensive and unified approach to AutoML-based anomaly detection in time series is called for. The possibilities are endless, and the future is exciting.

Bibliography

1. BISHOP, Christopher M. *Pattern Recognition and Machine Learning*. New York: Springer, 2006. Information Science and Statistics. ISBN 978-0-387-31073-2.
2. HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2009. Available also from: `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`.
3. *Current IOT forecast highlights*. [N.d.]. Available also from: `https://transformainsights.com/research/forecast/highlights`.
4. BRAEI, Mohammad; WAGNER, Sebastian. *Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art*. arXiv, 2020. No. arXiv:2004.00433.
5. GRUBBS, Frank E. Procedures for Detecting Outlying Observations in Samples. *Technometrics* [online]. 1969, vol. 11, no. 1, pp. 1–21 [visited on 2023-03-14]. ISSN 00401706. Available from: `http://www.jstor.org/stable/1266761`.
6. HAWKINS, D. M. *Identification of outliers*. London [u.a.]: Chapman and Hall, 1980. Monographs on applied probability and statistics. ISBN 041221900X. Available also from: `http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+02435757X&sourceid=fbw_bibsonomy`.
7. SCHMIDL, Sebastian; WENIG, Phillip; PAPENBROCK, Thorsten. Anomaly Detection in Time Series: A Comprehensive Evaluation. *Proceedings of the VLDB Endowment*. 2022, vol. 15, no. 9, pp. 1779–1797. ISSN 2150-8097. Available from DOI: `10.14778/3538598.3538602`.
8. LAI, Kwei-Herng; ZHA, Daochen; ZHAO, Yue; WANG, Guanchu; XU, Junjie; HU, Xia. Revisiting Time Series Outlier Detection: Definitions and Benchmarks. [N.d.].

9. CHOI, Kukjin; YI, Jihun; PARK, Changhwa; YOON, Sungroh. Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines. *IEEE Access*. 2021, vol. 9, pp. 120043–120065. ISSN 2169-3536. Available from DOI: 10.1109/ACCESS.2021.3107975.

10. BLÁZQUEZ-GARCÍA, Ane; CONDE, Angel; MORI, Usue; LOZANO, Jose A. *A Review on Outlier/Anomaly Detection in Time Series Data*. arXiv, 2020. No. arXiv:2002.04236.

11. ZHANG, Yuxin; CHEN, Yiqiang; WANG, Jindong; PAN, Zhiwen. *Unsupervised Deep Anomaly Detection for Multi-Sensor Time-Series Signals*. arXiv, 2021. No. arXiv:2107.12626.

12. KOZITSIN, Viacheslav; KATSER, Iurii; LAKONTSEV, Dmitry. Online Forecasting and Anomaly Detection Based on the ARIMA Model. *Applied Sciences*. 2021, vol. 11, no. 7. ISSN 2076-3417. Available from DOI: 10.3390/app11073194.

13. CHEN, Tianqi; GUESTRIN, Carlos. XGBoost. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. Available from DOI: 10.1145/2939672.2939785.

14. BREIMAN, Leo. Random Forests. *Machine Learning*. 2001, vol. 45, no. 1, pp. 5–32. ISSN 1573-0565. Available from DOI: 10.1023/A:1010933404324.

15. FRIEDMAN, Jerome H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*. 2001, vol. 29, no. 5, pp. 1189–1232. Available from DOI: 10.1214/aos/1013203451.

16. OLAH, Christopher. *Understanding LSTM networks*. 2015. Available also from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

17. HOCHREITER, Sepp. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*. 1998, vol. 06, no. 02, pp. 107–116. Available from DOI: 10.1142/S0218488598000094.

18. HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-Term Memory. *Neural Computation*. 1997, vol. 9, no. 8, pp. 1735–1780. ISSN 0899-7667. Available from DOI: 10.1162/neco.1997.9.8.1735.

19. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. *Attention Is All You Need*. 2017. Available from arXiv: 1706.03762 [cs.CL].

20. KIPF, Thomas N.; WELLING, Max. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. Available from arXiv: 1609.02907 [cs.LG].

21. DENG, Ailin; HOOI, Bryan. *Graph Neural Network-Based Anomaly Detection in Multivariate Time Series*. arXiv, 2021. No. arXiv:2106.06947.

22. CHEN, Zekai; CHEN, Dingshuo; ZHANG, Xiao; YUAN, Zixuan; CHENG, Xiuzhen. Learning Graph Structures with Transformer for Multivariate Time Series Anomaly Detection in IoT. *IEEE Internet of Things Journal*. 2022, vol. 9, no. 12, pp. 9179–9189. ISSN 2327-4662, ISSN 2372-2541. Available from DOI: 10.1109/JIOT.2021.3100509.

23. ZHOU, Haoyi; ZHANG, Shanghang; PENG, Jieqi; ZHANG, Shuai; LI, Jianxin; XIONG, Hui; ZHANG, Wancai. *Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting*. arXiv, 2021. No. arXiv:2012.07436. Available from DOI: 10.48550/arXiv.2012.07436.

24. KRAMER, Mark A. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*. 1991, vol. 37, no. 2, pp. 233–243. Available from DOI: https://doi.org/10.1002/aic.690370209.

25. KINGMA, Diederik P.; WELLING, Max. *Auto-Encoding Variational Bayes*. arXiv, 2013. No. arXiv:1312.6114.

26. DOERSCH, Carl. *Tutorial on Variational Autoencoders*. arXiv, 2021. No. arXiv:1606.05908.

27. COZZATTI, Michele; SIMONETTA, Federico; NTALAMPIRAS, Stavros. *Variational Autoencoders for Anomaly Detection in Respiratory Sounds*. arXiv, 2022. No. arXiv:2208.03326.

28. GUNDERSEN, Gregory. *The Reparameterization Trick*. 2018. Available also from: https://gregorygundersen.com/blog/2018/04/29/reparameterization/.

29. TULI, Shreshth; CASALE, Giuliano; JENNINGS, Nicholas R. *TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data*. 2022. Available from arXiv: 2201.07284 [cs.LG].

30. MALHOTRA, Pankaj; RAMAKRISHNAN, Anusha; ANAND, Gaurangi; VIG, Lovekesh; AGARWAL, Puneet; SHROFF, Gautam. *LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection*. arXiv, 2016. No. arXiv:1607.00148.

31. SENIN, Pavel; LIN, Jessica; WANG, Xing; OATES, Tim; GANDHI, S.; BOEDIHARDJO, Arnold P.; CHEN, Crystal; FRANKENSTEIN, Susan. Time series anomaly discovery with grammar-based compression. In: *International Conference on Extending Database Technology*. 2015.

32. BONIOL, Paul; PALPANAS, Themis. Series2Graph. *Proceedings of the VLDB Endowment*. 2020, vol. 13, no. 12, pp. 1821–1834. Available from DOI: 10.14778/3407790.3407792.

33. PATEL, P.; KEOGH, E.; LIN, J.; LONARDI, S. Mining motifs in massive time series databases. In: *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* 2002, pp. 370–377. Available from DOI: 10.1109/ICDM.2002.1183925.

34. JOLLIFFE, Ian T.; CADIMA, Jorge. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*. 2016, vol. 374, no. 2065, p. 20150202. Available from DOI: 10.1098/rsta.2015.0202.

35. HAN, Songqiao; HU, Xiyang; HUANG, Hailiang; JIANG, Mingqi; ZHAO, Yue. ADBench: Anomaly Detection Benchmark. 2022. Available from DOI: 10.48550/ARXIV.2206.09426.

36. HARMELING, Stefan; DORNHEGE, Guido; TAX, David; MEINECKE, Frank; MÜLLER, Klaus-Robert. From outliers to prototypes: Ordering data. *Neurocomputing*. 2006, vol. 69, no. 13, pp. 1608–1618. ISSN 0925-2312. Available from DOI: https://doi.org/10.1016/j.neucom.2005.05.015. Blind Source Separation and Independent Component Analysis.

37. RAKTHANMANON, Thanawin; CAMPANA, Bilson; MUEEN, Abdullah; BATISTA, Gustavo; WESTOVER, Brandon; ZHU, Qiang; ZAKARIA, Jesin; KEOGH, Eamonn. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping. In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '12*. Beijing, China: ACM Press, 2012, p. 262. ISBN 978-1-4503-1462-6. Available from DOI: 10.1145/2339530.2339576.

38. BREUNIG, Markus M.; KRIEGEL, Hans-Peter; NG, Raymond T.; SANDER, Jörg. LOF: Identifying Density-Based Local Outliers. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. Dallas, Texas, USA: Association for Computing Machinery, 2000, pp. 93–104. SIGMOD '00. ISBN 1581132174. Available from DOI: 10.1145/342009.335388.

39. SCHÖLKOPF, Bernhard; WILLIAMSON, Robert; SMOLA, Alex; SHAWE-TAYLOR, John; PLATT, John. Support Vector Method for Novelty Detection. In: 1999, vol. 12, pp. 582–588.

40. CORTES, Corinna; VAPNIK, Vladimir. Support-vector networks. *Machine Learning*. 1995, vol. 20, no. 3, pp. 273–297. ISSN 1573-0565. Available from DOI: 10.1007/BF00994018.

41. GOLDSTEIN, Markus; DENGEL, Andreas. Histogram-Based Outlier Score (HBOS): A Fast Unsupervised Anomaly Detection Algorithm. In: 2012.

42. THILL, Markus; KONEN, Wolfgang; BÄCK, Thomas. Time Series Anomaly Detection with Discrete Wavelet Transforms and Maximum Likelihood Estimation. In: 2019.

43. LIU, Fei Tony; TING, Kai; ZHOU, Zhi-Hua. Isolation Forest. In: 2009, pp. 413–422. Available from DOI: 10.1109/ICDM.2008.17.

44. POWERS, David. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness Correlation. *Mach. Learn. Technol.* 2008, vol. 2.

45. PAPARRIZOS, John; BONIOL, Paul; PALPANAS, Themis; TSAY, Ruey S.; ELMORE, Aaron; FRANKLIN, Michael J. Volume under the Surface: A New Accuracy Evaluation Measure for Time-Series Anomaly Detection. *Proceedings of the VLDB Endowment*. 2022, vol. 15, no. 11, pp. 2774–2787. ISSN 2150-8097. Available from DOI: 10.14778/3551793.3551830.

46. HANLEY, J A; MCNEIL, B J. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*. 1982, vol. 143, no. 1, pp. 29–36. Available from DOI: 10.1148/radiology.143.1.7063747. PMID: 7063747.

47. SAITO, Takaya; REHMSMEIER, Marc. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*. 2015, vol. 10, no. 3, pp. 1–21. Available from DOI: 10.1371/journal.pone.0118432.

48. WU, Renjie; KEOGH, Eamonn. Current Time Series Anomaly Detection Benchmarks Are Flawed and Are Creating the Illusion of Progress. *IEEE Transactions on Knowledge and Data Engineering*. 2021, pp. 1–1. ISSN 1041-4347, ISSN 1558-2191, ISSN 2326-3865. Available from DOI: 10.1109/TKDE.2021.3112126.

49. XU, Haowen; FENG, Yang; CHEN, Jie; WANG, Zhaogang; QIAO, Honglin; CHEN, Wenxiao; ZHAO, Nengwen; LI, Zeyan; BU, Jiahao; LI, Zhihan; LIU, Ying; ZHAO, Youjian; PEI, Dan. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*. ACM Press, 2018. Available from DOI: 10.1145/3178876.3185996.

50. SU, Ya; ZHAO, Youjian; NIU, Chenhao; LIU, Rong; SUN, Wei; PEI, Dan. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 2828–2837. KDD '19. ISBN 9781450362016. Available from DOI: 10.1145/3292500.3330672.

51. KIM, Siwon; CHOI, Kukjin; CHOI, Hyun-Soo; LEE, Byunghan; YOON, Sungroh. Towards a Rigorous Evaluation of Time-Series Anomaly Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2022, vol. 36, no. 7, pp. 7194–7201. ISSN 2374-3468, ISSN 2159-5399. Available from DOI: 10.1609/aaai.v36i7.20680.

52. BAHRI, Maroua; SALUTARI, Flavia; PUTINA, Andrian; SOZIO, Mauro. AutoML: State of the Art with a Focus on Anomaly Detection, Challenges, and Research Directions. *International Journal of Data Science and Analytics*. 2022, vol. 14, no. 2, pp. 113–126. ISSN 2364-415X, ISSN 2364-4168. Available from DOI: 10.1007/s41060-022-00309-0.

53. CLÉMENÇON, Stephan; THOMAS, Albert. *Mass Volume Curves and Anomaly Ranking*. arXiv, 2018. No. arXiv:1705.01305. Available from DOI: 10.48550/arXiv.1705.01305.

54. GOIX, Nicolas; SABOURIN, Anne; CLÉMENÇON, Stéphan. *On Anomaly Ranking and Excess-Mass Curves*. arXiv, 2015. No. arXiv:1502.01684. Available from DOI: 10.48550/arXiv.1502.01684.

55. THOMAS, Albert; FEUILLARD, Vincent; GRAMFORT, Alexandre; CLÉMENÇON, Stéphan. Learning Hyperparameters for Unsupervised Anomaly Detection. In: *ICML, Anomaly Detection Workshop*. 2016.

56. GOIX, Nicolas. *How to Evaluate the Quality of Unsupervised Anomaly Detection Algorithms?* arXiv, 2016. No. arXiv:1607.01152.

57. ELSHAWI, Radwa; MAHER, Mohamed; SAKR, Sherif. *Automated Machine Learning: State-of-The-Art and Open Challenges*. 2019. Available from arXiv: 1906.02287 [cs.LG].

58. LIASHCHYNSKYI, Petro; LIASHCHYNSKYI, Pavlo. *Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS*. 2019. Available from arXiv: 1912.06059 [cs.LG].

59. FALKNER, Stefan; KLEIN, Aaron; HUTTER, Frank. *BOHB: Robust and Efficient Hyperparameter Optimization at Scale*. 2018. Available from arXiv: 1807.01774 [cs.LG].

60. BERGSTRA, James; BARDENET, R.; KÉGL, Balázs; BENGIO, Y. Algorithms for Hyper-Parameter Optimization. In: 2011.

61. KATEHAKIS, Michael N.; VEINOTT, Arthur F. The Multi-Armed Bandit Problem: Decomposition and Computation. *Mathematics of Operations Research*. 1987, vol. 12, no. 2, pp. 262–268. Available from DOI: 10.1287/moor.12.2.262.

62. JAMIESON, Kevin; TALWALKAR, Ameet. *Non-stochastic Best Arm Identification and Hyperparameter Optimization*. 2015. Available from arXiv: 1502.07943 [cs.LG].

63. LI, Lisha; JAMIESON, Kevin; DESALVO, Giulia; ROSTAMIZADEH, Afshin; TALWALKAR, Ameet. *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization*. 2018. Available from arXiv: 1603.06560 [cs.LG].

64. VANSCHOREN, Joaquin. Meta-Learning. In: HUTTER, Frank; KOTTHOFF, Lars; VANSCHOREN, Joaquin (eds.). *Automated Machine Learning*. Cham: Springer International Publishing, 2019, pp. 35–61. ISBN 978-3-030-05317-8 978-3-030-05318-5. Available from DOI: 10.1007/978-3-030-05318-5_2.

65. DE SÁ, Alex; PINTO, Walter; OLIVEIRA, Luiz Otávio; PAPPA, Gisele. RECIPE: A Grammar-Based Framework for Automatically Evolving Classification Pipelines. In: 2017, pp. 246–261. ISBN 978-3-319-55695-6. Available from DOI: 10.1007/978-3-319-55696-3_16.

66. WOLPERT, David; MACREADY, William. No Free Lunch Theorems for Search. 1996.

67. THRUN, Sebastian; PRATT, Lorien Y. Learning to Learn: Introduction and Overview. In: *Learning to Learn*. 1998.

68. ZHAO, Yue; ROSSI, Ryan; AKOGLU, Leman. Automatic Unsupervised Outlier Model Selection. In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2021, vol. 34, pp. 4489–4502.

69. ZHAO, Yue; ROSSI, Ryan A.; AKOGLU, Leman. *Automating Outlier Detection via Meta-Learning* [online]. arXiv, 2021 [visited on 2022-12-12]. Available from: http://arxiv.org/abs/2009.10606. arXiv:2009.10606 [cs, stat].

70. ZHA, Daochen; LAI, Kwei-Herng; WAN, Mingyang; HU, Xia. *Meta-AAD: Active Anomaly Detection with Deep Reinforcement Learning*. 2020. Available from arXiv: 2009.07415 [cs.LG].

71. JÄRVELIN, Kalervo; KEKÄLÄINEN, Jaana. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 2002, vol. 20, pp. 422–446. Available from DOI: 10.1145/582415.582418.

72. ERMSHAUS, Arik; SCHÄFER, Patrick; LESER, Ulf. Window Size Selection Innbsp;Unsupervised Time Series Analytics: A Review Andnbsp;Benchmark. In: *Advanced Analytics and Learning on Temporal Data: 7th ECML PKDD Workshop, AALTD 2022, Grenoble, France, September 19–23, 2022, Revised Selected Papers*. Grenoble, France: Springer-Verlag, 2023, pp. 83–101. ISBN 978-3-031-24377-6. Available from DOI: 10.1007/978-3-031-24378-3_6.

73. VLACHOS, Michail; YU, Philip; CASTELLI, Vittorio. On Periodicity Detection and Structural Periodic Similarity. In: *Proceedings of the 2005 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2005, pp. 449–460. ISBN 978-0-89871-593-4 978-1-61197-275-7. Available from DOI: 10.1137/1.9781611972757.40.

74. WEN, Qingsong; HE, Kai; SUN, Liang; ZHANG, Yingying; KE, Min; XU, Huan. RobustPeriod: Time-Frequency Mining for Robust Multiple Periodicities Detection. *CoRR*. 2020, vol. abs/2002.09535. Available from arXiv: 2002.09535.

75. IMANI, Shima. Multi-Window-Finder: Domain Agnostic Window Size for Time Series Data. In: MileTS, 2021. Available also from: https://kdd-milets.github.io/milets2021/papers/MiLeTS2021_paper_9.pdf.

76. ERMSHAUS, Arik; SCHÄFER, Patrick; LESER, Ulf. ClaSP: parameter-free time series segmentation. *Data Mining and Knowledge Discovery*. 2023. Available from DOI: 10.1007/s10618-023-00923-x.

77. LAPTEV, Nikolay; AMIZADEH, Saeed; FLINT, Ian. Generic and Scalable Framework for Automated Time-Series Anomaly Detection. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2015, pp. 1939–1947. KDD '15. ISBN 9781450336642. Available from DOI: 10.1145/2783258.2788611.

78. AHMAD, Subutai; LAVIN, Alexander; PURDY, Scott; AGHA, Zuha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*. 2017, vol. 262, pp. 134–147. ISSN 0925-2312. Available from DOI: https://doi.org/10.1016/j.neucom.2017.04.070.

79. HUNDMAN, Kyle; CONSTANTINOU, Valentino; LAPORTE, Christopher; COLWELL, Ian; SÖDERSTRÖM, Tom. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding. *CoRR*. 2018, vol. abs/1802.04431. Available from arXiv: 1802.04431.

80. SU, Ya; ZHAO, Youjian; NIU, Chenhao; LIU, Rong; SUN, Wei; PEI, Dan. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 2828–2837. KDD '19. ISBN 9781450362016. Available from DOI: 10.1145/3292500.3330672.

81. SIMONOVITS, Miklos. How to compute the volume in high dimension? *Math. Program.* 2003, vol. 97, pp. 337–374. Available from DOI: 10.1007/s10107-003-0447-x.

Contents of the attached media

- `README.md` ..... Brief description of attached media contents
- `src`
  - `automltsad` ..... Implemented anomaly detection module
  - `experiments` File containing experiment scripts and Jupyter notebooks
- `text`
  - `thesis.pdf` ..... Thesis in PDF format