RETHINKING ANOMALY DETECTION:
FROM THEORY TO PRACTICE

Roel Christiaan Bouman

Institute for Computing and Information Sciences

RADBOUD UNIVERSITY PRESS

Radboud Dissertation Series

RETHINKING ANOMALY DETECTION: FROM THEORY TO PRACTICE

Roel Christiaan Bouman

COLOPHON

Roel Bouman
Rethinking Anomaly Detection: From Theory to Practice

Radboud Dissertation Series
ISSN: 2950-2772 (Online); 2950-2780 (Print)

Published by RADBOUD UNIVERSITY PRESS
Postbus 9100, 6500 HA Nijmegen, The Netherlands
www.radbouduniversitypress.nl

Design: Proefschrift AIO/Guus Gijben
Cover image: Guntra Laivacuma
Printing: DPN Rikken/Pumbo

ISBN: 9789465150765
DOI: 10.54195/9789465150765
Free download at: https://doi.org/10.54195/9789465150765

RADBOUD
UNIVERSITY
PRESS

RETHINKING ANOMALY DETECTION: FROM THEORY TO PRACTICE

Proefschrift ter verkrijging van de graad van doctor
aan de Radboud Universiteit Nijmegen
op gezag van de rector magnificus prof. dr. J.M. Sanders,
volgens besluit van het college voor promoties
in het openbaar te verdedigen op

dinsdag 15 april 2025
om 12:30 uur precies

door

Roel Christiaan Bouman
geboren op 14 december 1993
te Gendringen

Promotor

Prof. dr. Tom M. Heskes

Manuscriptcommissie

Prof. dr. Marco Loog
Prof. dr. Kerstin Bunte (*Rijksuniversiteit Groningen*)
Prof. dr. Thomas H.W. Bäck (*Universiteit Leiden*)

RETHINKING ANOMALY DETECTION: FROM THEORY TO PRACTICE

Dissertation to obtain the degree of doctor
from Radboud University Nijmegen
on the authority of the Rector Magnificus prof. dr. J.M. Sanders,
according to the decision of the Doctorate Board
to be defended in public on

Tuesday, April 15, 2025
at 12:30 p.m.

by

Roel Christiaan Bouman
born on December 14, 1993
in Gendringen (the Netherlands)

Supervisor

Prof. dr. Tom M. Heskes

Manuscript Committee

Prof. dr. Marco Loog
Prof. dr. Kerstin Bunte (*University of Groningen*)
Prof. dr. Thomas H.W. Bäck (*Leiden University*)

SUMMARY

The field of anomaly detection is growing at an increasing pace. New algorithms and applications are numerous. Yet, the fundamentals of anomaly detection are more rarely studied than anomaly detection is applied. In this thesis, we aim to provide guidelines on which algorithms actually perform well in practice, whether the autoencoders, one of the most popular methods, for anomaly detection are actually reliable, and lastly we present a novel use case of anomaly detection on the improvement load estimation on the Dutch power grid.

There are many algorithms for performing anomaly detection, and even more applications. If we want to perform anomaly detection, it is imperative to know which methods to use. In order to provide guidelines on this, we have performed the largest comparative study on real-world data to date. We find that actually only a few algorithms are needed in order to get satisfactory results on most data. We specifically find that k-nearest neighbors and extended isolation forest are able to detect the most commonly occurring types of anomalies.

Beyond traditional tabular applications we see an increasing number of studies and applications where autoencoders are used to detect anomalies. As a neural network architecture, autoencoders are most notably used to detect anomalies in fields, such as computer vision, where their ability to extract features is essential. In recent years, researchers have started questioning the assumptions behind how autoencoders detect anomalies. We study these assumptions in detail and definitively conclude that there are no guarantees that these assumptions hold in practice. This means that autoencoders are unreliable anomaly detectors.

We further apply our knowledge on anomaly detection to the energy domain. With the energy transition in full swing, we need to take every possible step in order to maximize our current use of the power grid. We combine anomaly detection through reliable methods such as statistical process control and binary segmentation in order to detect measurement errors and the rerouting of power in load measurements. By filtering these errors and rerouting, or switch, events we can acquire more reliable load estimates of the power grid in the Netherlands. By acquiring better load estimates through interpretable methods we can reduce unused capacity of the power grid, allowing for the much-needed leeway we need during the energy transition.

SAMENVATTING

Anomaly detection, soms vertaald als anomaliedetectie, is een klein, doch toenemend belangrijk, deel van de kunstmatige intelligentie. Nieuwe algoritmen en toepassingen zijn ontelbaar. Toch worden de basisbeginselen van *anomaly detection* minder vaak bestudeerd dan dat het wordt toegepast. In deze dissertatie streven we ernaar om onderzoekers houvast te bieden over welke algoritmen in de praktijk daadwerkelijk goed werken, en of *autoencoders* – een van de meest populaire methoden voor *anomaly detection* – daadwerkelijk betrouwbaar zijn. Tot slot presenteren we een nieuwe toepassing van *anomaly detection* voor het verbeteren van de capaciteitsinschattingen in het Nederlandse elektriciteitsnet.

Er zijn veel algoritmen beschikbaar voor het uitvoeren van *anomaly detection*. Als we dit willen doen, is het belangrijk om te weten welke methoden we moeten gebruiken. Om hier richtlijnen voor te bieden, hebben we het grootste vergelijkende onderzoek tot nu toe uitgevoerd. We concluderen dat slechts een paar algoritmen nodig zijn om in de meeste gevallen goede resultaten te behalen. Specifiek vinden we dat de *k-nearest neighbors* en *extended isolation forest* algoritmen de meest voorkomende typen anomalieën kunnen detecteren.

Naast traditionele tabulaire toepassingen zijn er een groeiend aantal studies en toepassingen waarin *autoencoders* worden gebruikt om anomalieën te detecteren. Neurale netwerken zoals *autoencoders* worden vooral gebruikt voor *anomaly detection* in vakgebieden zoals automatische beeldherkenning. In de afgelopen jaren zijn onderzoekers echter de aannames achter hoe *autoencoders* anomalieën detecteren in twijfel gaan trekken. Wij onderzoeken deze aannames in detail en concluderen definitief dat er geen garanties zijn dat deze aannames in de praktijk standhouden. Dit betekent dat *autoencoders* onbetrouwbare anomaliedetectoren zijn.

We passen onze kennis van *anomaly detection* verder toe in de energiesector. In het kader van de energietransitie moeten we alle mogelijke stappen ondernemen om ons huidige gebruik van het elektriciteitsnet te maximaliseren. We combineren *anomaly detection* door middel van betrouwbare methoden zoals *statistical process control* met *binary segmentation* om meetfouten en verschakelingen in belastingmetingen te detecteren. Door deze fouten en verschakelingen te filteren, kunnen we betrouwbaardere capaciteitsinschattingen van het elektriciteitsnet in Nederland verkrijgen. Door betere inschattingen te verkrijgen via interpreteerbare methoden, kunnen we ongebruikte capaciteit van het elektriciteitsnet verminderen, wat extra ruimte biedt die we nodig hebben tijdens de energietransitie.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

1 INTRODUCTION

In this introduction, we will provide an overview of what anomaly detection is, applications of anomaly detection, what current gaps there are in the field, a more detailed overview of methods and considerations, and how we have aimed to fill the gaps in anomaly detection.

1.1 WHAT IS ANOMALY DETECTION?

Anomaly detection, also commonly known as outlier detection or more rarely deviation detection, is the process of identifying data points that deviate significantly from the majority of the data. These data points, or anomalies, may, for example, represent rare events or measurements, such as fraud, network intrusions, equipment failures, rare cancer cells, or other significant events. Anomaly detection plays an increasingly crucial role in a wide range of applications across various fields, especially when the data cannot be used for more typical machine learning tasks such as classification or regression.

1.1.1 *What Is an Anomaly and Where Is It Applied?*

Anomalies are data points that are in some way dissimilar to normal data. In many cases anomalies are also more heterogeneous than the normal data. Generally we consider some m-by-n dataset $\mathbf{X} \in \mathcal{R}^n$ in which we want to detect one or more anomalous samples $\mathbf{a} = \mathbf{x}_i$, which are sufficiently anomalous, i.e., $f_{\text{anomaly score}}(\mathbf{a}) > \delta$, according to some chosen "anomalousness" criterion function. Dissimilarity, or anomalousness, in the context of anomalies can mean a variety of things and is generally domain or application dependent. Anomalies can, for example, be far away from all normal data, or occupy a region of low density. The method of characterization is quite different between different types anomaly detection algorithms, meaning some methods are better at detecting distinct types of anomalies. In Chapter 2 we therefore propose a new taxonomy of anomaly properties. Rather than define anomalies by binary properties, we instead ascribe non-exclusive continuous properties which can be used to give a more detailed description of different types of anomalies. An overview of many of the common methods and approaches used in anomaly detection can be found in Section 1.3.4.

1.1.2 *Applications*

Anomaly detection is done across a variety of domains, as long as identification of rare samples or patterns is of potential benefit.

In finance, anomaly detection is often employed for fraud detection and market analysis. Banks can for example use it to identify unauthorized transactions, money laundering, or identity theft by monitoring transaction behaviors in real time [2, 26]. Detecting irregular trading activities or sudden market shifts through anomaly detection provides early warnings of potential market disruptions or investment opportunities, assisting traders and analysts in making informed decisions [1, 20]. In financial auditing, anomaly detection may be employed as a preselection mechanism to detect suspicious or fraudulent activity in accounting statements [30].

Healthcare can similarly benefit from anomaly detection in for example disease outbreak detection, medical diagnostics, and disease monitoring. Epidemiologists can spot unusual patterns in patient data that might indicate disease outbreaks, allowing for faster responses and containment measures [58]. In medical diagnostics, anomalies in imaging data or patient vitals can signal the presence of diseases or conditions supporting doctors in making accurate diagnoses [16]. In disease monitoring anomaly detection can be used to monitor the presence of rare disease specific cells [17].

In cybersecurity, anomaly detection is used for network intrusion detection and user behavior analysis. Network traffic can be monitored to spot unusual patterns indicative of security breaches, such as unauthorized access or malware infections, enabling organizations to mitigate cyber attacks [32]. Additionally, analyzing user behavior for anomalies like unusual login times or access to sensitive data helps identify insider threats or compromised accounts, enhancing information system security [43].

Anomaly detection is an essential step in performing predictive maintenance and quality/process control [14, 29]. By analyzing sensor data from equipment, signs of wear, malfunctions, or failures can be identified before they occur, allowing for just-in-time, rather than scheduled, maintenance and reducing downtime. Detecting anomalies in production processes or product quality ensures defective products are identified and corrected early, maintaining high standards and minimizing waste [42].

Environmental monitoring employs anomaly detection in climate monitoring and wildlife conservation. It identifies significant changes or events in environmental data, such as temperature, precipitation, or pollution levels, providing early warnings for timely responses to climate change effects [12] or natural disasters [27]. Monitoring wildlife populations or behaviors for anomalies aids conservation efforts by identifying threats like poaching [6], or disease outbreaks [25].

The energy sector utilizes anomaly detection in for example smart grid monitoring [46]. It can aid in identifying power outages [40], or energy theft [59], helping maintain a stable and efficient energy supply whilst aiding in the energy transition. With the rapid advent of the energy transition, planning grid expansions is becoming increasingly important to make optimal use of the existing network with limited available resources. In Chapter 4 we discuss an application of change-point detection and anomaly detection to acquire better load estimates to facilitate grid expansion planning.

In the semiconductor industry, anomaly detection can help ensure the quality and reliability of integrated circuits and semiconductor devices. The complex manufacturing processes, including photolithography, etching, and deposition, can produce defects from even minor deviations. Real-time monitoring systems detect these irregularities early, allowing manufacturers to intervene promptly, reduce waste, improve yields, and maintain high product quality [33, 55].

In additive manufacturing, or 3D printing, anomaly detection is crucial for maintaining precision and integrity in produced components. The layer-by-layer construction process must be meticulously monitored to identify inconsistencies, temperature fluctuations, or material defects. Continuous anomaly detection enables immediate predictive discarding or corrective actions, minimizing defects and material wastage, ensuring the final products meet required specifications, and supporting the advancement of reliable, high-quality additive manufacturing [53, 54].

1.2 SOME GAPS IN THE FIELD OF ANOMALY DETECTION

In the past few years we have identified several unresolved issues, open questions, points for improvements, and novel applications in anomaly detection. In this work we summarize our main contributions. In this section, we will pose the research questions, which we summarily answer in Section 1.4. In the remaining chapters, we answer each of these questions in more detail.

There is a clear need to know which anomaly detection algorithms perform well in practice. Computational resources and time of researchers and domain experts is scarce, so any guideline which can help decide "where to start" is quickly invaluable for anyone applying anomaly detection. A large part of the anomaly detection research is focused on new applications and the introduction of new methods, but little comparative research has been performed in order to establish the actual efficacy of algorithms. What research has been done is often either outdated, small-scale, or on synthesized data. This leads us to the following research questions:

1. Which anomaly detection algorithms perform well on real-world data?

One of the most popular methods for anomaly detection, especially in deep learning, is the autoencoder, which we explain in more detail in Section 1.3.4.7. The autoencoder is a neural network which can be used to detect anomalies. An autoencoder learns an encoder, which finds a lower-dimensional representation of the input data, and an decoder which aims to reconstruct the input data from this representation. After training the autoencoder on normal data or a mixture of normal and anomalous data, we can use the reconstruction loss to determine whether something is an anomaly or not. A higher reconstruction loss is in this case assumed to be indicative of an anomaly. Recent works have however questioned this assumption. We then ask:

> 2. Is it reasonable to assume that anomalies are harder to reconstruct than normal data?

Beyond comparative and theoretical research there exists a plethora of new applications for anomaly detection. One such a novel application can be found within the energy domain. Due to the energy transition it is becoming increasingly important to make strategic decisions on grid expansion planning based on accurate data. However, measured load data is often contaminated with switch events and measurement errors. Filtering these is a laborious and time-consuming manual task, prone to human error. Presented with this problem, we wondered:

> 3. Can anomaly detection be used to automate and improve the filtering procedure in power grid load measurements?

1.3 AN IN-DEPTH LOOK AT ANOMALY DETECTION

We will now consider anomaly detection more in-depth. We will discuss several common paradigms of applying anomaly detection, as well as many of the most frequently used anomaly detection methods. We will assume the reader is familiar with basic concepts of statistics, data science, and machine learning. Many of the methods and concepts we describe find uses beyond anomaly detection, but for the sake of brevity we will not consider these in detail.

1.3.1 *Anomaly detection paradigms*

The type and quality of available data is one of the major driving forces in determining which anomaly detection algorithms can be applied. In general we discern between: supervised anomaly detection, where labels of both anomalies and normal data are available; unsupervised anomaly detection, where no labels are available; and semi-supervised anomaly detection, where labels are available only

for normal data. These different paradigms are explained in more detail in the following sections.

In some cases, labels are available, but of low quality due to what is called label noise [18]. This refers to the phenomenon where data has been labelled, but is in many cases labelled incorrectly. When label noise is prevalent, one has to be more conservative in which paradigm to apply, meaning unsupervised is often preferred over supervised anomaly detection.

1.3.1.1 *Supervised anomaly detection*

Supervised anomaly detection, a part of supervised learning, is the process of learning from data in which each measurement is connected to a target variable, which a learning algorithm will try to predict. When this target variable is categorical, we generally call the task "classification", while for continuous target variables we call the process "regression". Depending on the body of literature, target variables might also be called "dependent variables", or "outputs" [23]. Within classification, the supervised anomaly detection problem often suffers from class imbalance, and might be called rare class classification [24]. This means that not all classes are equally represented in the dataset. Multi-class problems in this paradigm are generally distilled to problems with only two classes: normal and anomalous. Often, the normal class is represented in most measurements, while anomalies are exceedingly rare [45]. When analyzing this type of data with classification algorithms, extra care has to be taken to circumvent the class imbalance problem [36], either through steps like under or oversampling, or using algorithms which can natively deal with the data imbalance. We generally represent the data in supervised anomaly detection in the form of a matrix and a vector, the matrix containing the measurement data $\mathbf{X} = \{\mathbf{X}^{\text{normal}}, \mathbf{X}^{\text{anomalous}}\}$, and the vector $\mathbf{y}$ containing the labels, 0 for normal data, and 1 for anomalies.

An important assumption in supervised anomaly detection is that even though anomalies are rare, they are homogeneous enough to be modelled, meaning that anomalies share similar characteristics. When future anomalies might exhibit new characteristics, for example when there are new modes of failure, or unprecedented events, semi-supervised anomaly detection is a better choice of paradigm.

1.3.1.2 *Unsupervised anomaly detection*

In unsupervised anomaly detection, there are no labels present or they are of low quality due to label noise. In many cases, labels are not freely available, for example, due to missing annotations of data, or because anomaly detection is applied as a more exploratory research step. When labels are absent, or when labels are unreliable due to label noise, unsupervised anomaly detection is the only op-

tion. Unsupervised anomaly detection is characterized by having no discernible "train" and "test" steps. Instead, we only consider a single dataset $\mathbf{X} = \{\mathbf{X}^{\text{normal}}, \mathbf{X}^{\text{anomalous}}\}$, where we are uncertain which samples are anomalous and which are not.

In contrast to supervised learning, unsupervised algorithms cannot discern between variables with and without predictive power for anomalousness. This means that each variable used as input has to be carefully selected, and vetted if expert knowledge deems it to be likely to be noise [57].

1.3.1.3 *Semi-Supervised Anomaly Detection*

Semi-supervised anomaly detection algorithms learn from labeled normal data to ideally increase their performance over their unsupervised counterparts. The dataset includes a small labeled subset of normal data used for training, $\mathbf{X}_{\text{train}} = \mathbf{X}_{\text{train, normal}}$, and a large unlabeled subset, $\mathbf{X}_{\text{unlabeled}}$ containing both anomalies and normal data which is generally not used for training. The labeled data can be used to learn a model of the normal data. Generally, semi-supervised anomaly detection can be seen as a subset of one-class classification, where the normal data is treated as the modeled class, and anomalies are all samples that do not fit the model [52]. The term novelty detection is nowadays mostly being used to denote anomaly detection in general, but historically mostly refers to the semi-supervised setting [52].

1.3.2 *Model Evaluation and Optimization*

To gauge the performance of anomaly detection algorithms in practice, we often have to employ specific strategies to estimate how well models can detect anomalies in unseen data. Most of these strategies are dependent on the availability of the labels, and therefore differ significantly between the different anomaly detection paradigms.

1.3.2.1 *Evaluation Metrics*

Evaluation metrics are crucial for assessing the performance of anomaly detection models. In this section we provide an overview of the most commonly used evaluation metrics. As nearly all anomaly detection applications handle imbalanced data, meaning normal data is much more common than anomalous data, we will only discuss those metrics that are useful on that type of data.

PRECISION AND RECALL Precision and recall are metrics commonly used in the context of imbalanced datasets. The precision is the ratio of true positive predictions to the total positive predictions.

High precision indicates that the model makes few false positive errors.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall, also called Sensitivity or True Positive Rate, is the ratio of true positive predictions to the total actual positives. High recall indicates that the model captures most of the actual anomalies.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

FB-SCORE The Fβ-score is a generalization of the F1-score that allows different weights to be assigned to precision and recall. It is particularly useful when the importance of precision and recall differs. The parameter β determines the weight of recall in the combined score. When $\beta = 1$, the Fβ-score is equivalent to the F1-score, giving equal weight to precision and recall. When $\beta > 1$, recall is weighted more heavily, and when $\beta < 1$, precision is weighted more heavily.

$$\text{F}\beta\text{-Score} = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$$

AREA UNDER THE RECEIVER OPERATING CHARACTERISTIC CURVE The Receiver Operating Characteristic, or ROC, curve is a graphical representation of the trade-off between the true positive rate, or recall, and the false positive rate at various threshold settings. The Area Under the Curve, or AUC, provides a single value that summarizes the performance of the model across all thresholds. An AUC of 1 indicates perfect performance, while an AUC of 0.5 suggests a model that performs no better than random guessing.

AREA UNDER THE PRECISION-RECALL CURVE Similar to the ROC curve, the Precision-Recall curve plots precision against recall for different threshold values. This is particularly informative for imbalanced datasets, as it focuses on the performance of the positive (anomalous) class. The Area Under the Precision-Recall Curve provides an aggregate measure of the model's performance across all thresholds.

1.3.2.2 *Supervised Anomaly detection*

In supervised anomaly detection, we often discern between separate "train", "test", and "validation" steps in which the data is split into subsets, generally using schemes like holdout or cross-validation. This allows for tuning the hyperparameters of the method, and for getting an estimate of the generalization performance of the model.

In supervised anomaly detection the problem is similar to supervised classification, with the exception that the problem is almost always binary; data is either normal, or anomalous. A train set is used to learn the model parameters so it can subsequently be applied to new data. The validation set is unseen data, i.e., not used during training, on which the learned model is applied in order to acquire a prediction of anomaly scores. When models trained with different hyperparameters are applied on the validation set, the best hyperparameters can be selected by choosing those hyperparameters which yield the highest performance metric on the validation set. To get an estimate of how well the model with the selected hyperparameters performs on unseen data, we need an unbiased estimate. We acquire this by evaluating the model on the test data. In some cases we only consider a train and a test set, and forgo the validation set. In that case, we can only optimize the hyperparameters or get an estimate of the model performance on unseen data, but not both.

Holdout validation is one of the more commonly employed to split the data into subsets. Holdout validation involves constructing the subsets by dividing the total collection of data according to certain predefined sizes of subsets. Typically, the training set is the largest set, and the test and validation sets are smaller. A simple example is when we split a dataset of 200 samples so that the training, validation, and test set each make up 60%, 20%, and 20% of the total data. In this way, we would end up with a train set of 120 samples, and validation and test sets of 40 samples each.

Holdout validation is sensitive to fluctuations in the evaluation measure due to the random splitting of the data. In order to alleviate this issue, cross-validation is often employed. In cross-validation, we repeatedly perform splits, and average the results over various splits. In this way we can acquire a more accurate estimate of the model performance. Typical strategies for cross-validation are k-fold cross-validation, and leave-p-out cross-validation. We will not consider these strategies in more detail in this treatise.

1.3.2.3 *Semi-Supervised Anomaly Detection*

Semi-supervised anomaly detection, although it can use similar validation schemes as supervised methods, faces the challenge of having labeled normal data but no labeled anomalies. The train set that is used to learn the model parameters generally contains only normal data. When one wants to optimize hyperparameters or gauge generalization performance, anomalies are needed in the validation or test set. In practice, these labels might not always be available, making actual optimization difficult.

In absence of labels we can use on internal, model-dependent, evaluation metrics such as reconstruction error or other types of model fit during validation. Using proxy measures such as the aforementioned

might however not necessarily lead to a better model performance in terms of the chosen metrics.

1.3.2.4 *Unsupervised Anomaly Detection*

In unsupervised anomaly detection presents unique challenges traditional model tuning and validation are exceedingly difficult due to the absence of labels.

Like the semi-supervised case, we might use model-dependent internal criteria for evaluation and optimization. Some researchers have proposed model-independent unsupervised criteria, or "internal evaluation strategies," to allow for optimization [19, 39, 41]. These include metrics such as the silhouette score for clustering algorithms or the reconstruction loss for autoencoders. However, comparative research has found that many of these strategies are not practically useful due to their inability to consistently correlate with actual anomaly detection performance [37].

Because model selection, optimization, or evaluation is impossible in those cases where labels are absent, it is important to choose a generally well-performing method. In supervised classification, large scale comparison studies have been performed [15]. Large scale comparison studies such as the aforementioned can help in picking generally well-performing algorithms. In unsupervised anomaly detection, comparative studies have been of much smaller scale, or have been performed on synthesized, rather than real-world anomalies. To close the gap we have in this work, in Chapter 2, performed the largest comparison study of unsupervised anomaly detection algorithms. We hope to thus provide better guidelines on which algorithms perform well and should be good “first picks”.

1.3.3 *Types of Data*

Anomaly detection is applied on a wide variety of data types. Different types of data generally warrant the use of different anomaly detection methods, though there is a lot of common ground between them.

1.3.3.1 *Tabular*

Tabular data is one of the most common and well-studied types of data. It is generally represented as an m-by-n matrix of m samples and n variables. Typically this type of data is stored in relational databases, spreadsheets, or CSV files. Though many other types of data can be stored or represented in a tabular way, we will only refer to them as being tabular when each sample is independent, and the feature ordering is unimportant. Many classical anomaly detection methods have been developed specifically for this type of data.

1.3.3.2 *Images*

Images are often the objects of interest in computer vision tasks. Image anomaly detection has surged in popularity due to the recent interest in computer vision with the advent of deep learning. Traditional analysis of images was done using dimensionality reduction techniques such as PCA. Nowadays, this has mostly been supplanted by a variety of neural network based anomaly detection models which can leverage the spatial connection between pixels.

1.3.3.3 *Time Series*

Time series data consists out of a sequence of data points in time order. This means that there is some temporal connection between subsequent samples. When analyzing time series data using anomaly detection, it is often vital to account for temporal dependencies and patterns. Taking trends and seasonality into account can enhance anomaly detection performance. Many applications also require real-time processing capabilities, necessitating efficient online algorithms. In many cases, time series anomaly detection is done by looking at the residuals of a regression where the next point in time is predicted. If the residuals are large, i.e., the model's prediction is inaccurate, a time point is deemed more likely to be an anomaly.

1.3.3.4 *Video*

Video data is quite common in anomaly detection applications such as camera surveillance. It can be viewed as a combination of traditional image and time series anomaly detection, as it has both a spatial relation between pixels, and a temporal relation between subsequent frames. In many applications it is however still common to see the input data being treated as separate images, treating lengths of normal video as separate normal images used to train semi-supervised anomaly detection algorithms [4].

1.3.4 *Common Methods*

Many methods exist for detecting anomalies, more than can feasibly be listed or explained thoroughly. We will rather give summary descriptions of a large number of common approaches for the detection of anomalies. We will do so in a manner that best preserves the commonalities between methods. We will focus on classical methods used for the detection of anomalies in tabular data. Many of the neural networks methods popular nowadays build on these principles, as we will discuss in Section 1.3.4.7. In the rest of this thesis, we will mostly discuss tabular and computer vision applications, which will build from the explanations below. In Chapter 4 we consider time-series

anomaly detection, which we will handle with tabular methods after applying extensive preprocessing in order to forgo the time component.

We have grouped many methods, but this taxonomy is by no means conclusive, as categories can just as easily be renamed and reshuffled, and methods can be interchangeable between categories. By presenting our summary in this way, we hope to have this introduction serve as a simple tutorial to the most commonly applied anomaly detection methods.

1.3.4.1 *Statistical Approaches*

In statistics, anomalies are often called outliers. One often wants to remove outliers to limit their influence on statistical models. Summary statistics such as the mean and standard deviation are for example heavily influenced by the presence of samples which deviate greatly from other data. Often when outliers are removed, it is either assumed that they are caused by experimental error, or they belong to some other distribution that is not of interest.

GRUBBS'S TEST A typical statistical method to detect a outliers is Grubbs's test [21]. Grubbs's test detects a single outlier in a one-dimensional normally distributed dataset $\mathbf{x}$ with samples $x_i \in \mathbf{x}$ by calculating the test statistic G as:

$$G = \frac{\max_i(|x_i - \bar{\mathbf{x}}|)}{s},$$

where $\max_i(|x_i - \bar{x}|)$ is the largest absolute deviation from the mean $\bar{x}$, and s is the standard deviation calculated over $\mathbf{x}$. This statistic is compared to a critical value G_{critical} from the reference table based on the sample size m and significance level α. If G exceeds G_{critical}, the extreme value is deemed an outlier. The test assumes normality of the input data and is designed to identify only one outlier at a time. Grubbs's test does not explicitly assume the presence of outliers in the dataset.

DIXON'S Q TEST Similar to Grubbs's test, Dixon's Q test is used to detect a single outlier in a small dataset which is assumed to be normally distributed [13]. In contrast to Grubbs's test, Dixon's Q test assumes the presence of an outlier, and aims to overcome the masking effect, where the outlier affects the statistics used to calculate the "outlierness", thereby making it harder to detect the outlier. The test statistic Q is calculated as:

$$Q = \frac{\text{gap}}{\text{range}},$$

where gap is the absolute difference between the suspected outlier and its nearest neighbor, and range is the difference between the max-

imum and minimum values in the dataset. The calculated Q is compared to a critical value $Q_{critical}$ from the reference table, which varies with sample size n and significance level α. If Q exceeds $Q_{critical}$, the suspected outlier is considered significant.

MEDIAN ABSOLUTE DEVIATION (MAD) The Median Absolute Deviation, or MAD, is commonly used to detect outlier [22, 47]. Rather than Dixon's and Grubb's tests, it is more often used to detect multiple outliers at once. The MAD is a robust statistic used to measure the dispersion of data around the median. It is commonly used for outlier detection by determining if a data point significantly deviates from the median compared to the typical deviation of the dataset. It is calculated by first finding the median $\tilde{x}$. Next, the absolute deviations from the median are computed as $|x_i - \tilde{x}|$ for each data point x_i. The MAD is then obtained by taking the median of these absolute deviations. Mathematically, this is expressed as:

$$\text{MAD} = \text{median}(|x_i - \tilde{x}|).$$

A data point x_i is often considered an outlier if its absolute deviation from the median is greater than a certain multiple of the MAD, such as 2.5 or 3. This method is robust to the presence of outliers and therefore more often used when a dataset is suspected to contain multiple outliers.

MAHALANOBIS DISTANCE When considering more than one variable, the Mahalanobis distance can be used to detect outliers. The Mahalanobis distance is a multivariate measure of the distance between a point and a distribution, accounting for the correlations between variables [38]. It is calculated as:

$$d_{\text{Mahalanobis}}(\boldsymbol{x}) = \sqrt{(\boldsymbol{x} - \bar{\boldsymbol{x}})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \bar{\boldsymbol{x}})},$$

where $\boldsymbol{\Sigma}^{-1}$ is the inverse of the covariance matrix of the dataset $\mathbf{X}$. The Mahalanobis distance quantifies how many standard deviations away a data point is from the mean of the distribution, taking into account the shape of the distribution.

In outlier detection, the Mahalanobis distance is used to identify points that are significantly different from the expected distribution. A data point is considered an outlier if its Mahalanobis distance is greater than a critical value derived from the Chi-squared distribution, given a chosen significance level and the number of dimensions n. This approach can be used for identifying outliers in multivariate datasets.

GAUSSIAN MIXTURE MODEL When the data is assumed to be both multivariate and multimodal, Gaussian Mixture Models, or GMMs

are often used. A GMM is a probabilistic model that assumes all the data points are generated from a mixture of k Gaussian distributions with unknown parameters. Each Gaussian in the mixture is defined by its mean vector μ_k and covariance matrix Σ_k. The probability density function of the GMM is given by:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k),$$

where K is the number of Gaussian components, π_k are the mixture weights, such that $\sum_{k=1}^{K} \pi_k = 1$, and $\mathcal{N}(x|\mu_k, \Sigma_k)$ represents the Gaussian distribution with mean μ_k and covariance Σ_k.

GMMs can be used for outlier detection by determining the likelihood of each data point under the fitted model. Data points with very low likelihoods are then considered outliers. This is done by calculating the log-likelihood of each point and identifying those that fall below a certain threshold, which is often determined based on a chosen significance level. GMMs are particularly useful in handling complex data distributions and can effectively model multimodal data where traditional methods may fail. As GMMs use underlying Gaussian distributions, they are not robust to the presence of outliers.

1.3.4.2 *Density-based methods*

In many cases, traditional statistical methods do not perform well or are hard to use. This can be caused by unfulfilled assumptions, or due to computational cost when too many parameters need to be estimated. Some of the most common methods used for anomaly detection originate from data mining and machine learning, where fewer parametric assumptions are made. Many of these models implicitly, rather than explicitly, model density, and detect anomalies when they occupy spaces of low density.

k-nearest neighbors (kNN) kNN is one of the more straightforward methods of implicit density estimation used for anomaly detection [44]. The principle behind kNN is to measure the distance of each data point to its k-nearest neighbors and use this distance to determine if the point is an anomaly. For each data point $\mathbf{x}_i$, the distances to all other points in the dataset are calculated. Then, the k-nearest neighbors of each data point are identified. The anomaly score for each point is then some aggregation over the distances to its kth nearest neighbors, such as the maximum or the average distance.

Mathematically, the anomaly score for a data point $\mathbf{x}_i$ can be represented as:

$$\text{score}(\mathbf{x}_i) = f_j^k(d(\mathbf{x}_i, \mathbf{x}_{i_j})),$$

where $d(\boldsymbol{x}_i, \boldsymbol{x}_{i_j})$ is some chosen distance between $\boldsymbol{x}_i$ and its jth nearest neighbor $\boldsymbol{x}_{i_j}$, and f_j^k is some aggregation function iterating over the $j = 1, \ldots, k$ nearest neighbors of sample $\boldsymbol{x}_i$.

In kNN based anomaly detection, data points with the highest anomaly scores are considered anomalies. This method is quite versatile and can be used with various distance metrics, such as Euclidean distance. It is particularly effective for detecting anomalies in datasets with no prior assumption about the data distribution.

LOCAL OUTLIER FACTOR (LOF) The Local Outlier Factor, or LOF, is a popular method used for anomaly detection [10]. It measures the local distance of a data point with respect to its neighbors. The LOF score is based on the concept of local density, where the density is estimated by the distance to the point's k-nearest neighbors and the distance of those neighbors to their k-nearest neighbors.

For the sake of notation we will forgo the passing of the entire dataset $\boldsymbol{X}$ to all functions where k is a parameter. Let us now define the reachability distance between two data points:

$$r_k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \max(d_k(\boldsymbol{x}_j), d(\boldsymbol{x}_i, \boldsymbol{x}_j)),$$

where $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ indicates some distance metric, and $d_k(\boldsymbol{x}_j)$ indicates the distance, according to some distance metric, to the k-th nearest neighbor of $\boldsymbol{x}_j$.

From this, the local reachability density is calculated:

$$\mathrm{LRD}_k(\boldsymbol{x}_i) = \left(\frac{\sum_{\boldsymbol{x}_j \in N_k(\boldsymbol{x}_i)} r_k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{|N_k(\boldsymbol{x}_i)|} \right)^{-1},$$

where $N_k(\boldsymbol{x}_i)$ is the set data points within $d_k(\boldsymbol{x}_i)$, which may be higher than k in case of ties, $|N_k(\boldsymbol{x}_i)|$ then indicates the cardinality of the set, or the number of data points within $d_k(\boldsymbol{x}_i)$. The LOF score is then computed by comparing the local reachability density of the point to the local reachability densities of its neighbors. Mathematically, the LOF score for a data point $\boldsymbol{x}_i$ can be expressed as:

$$\mathrm{LOF}(\boldsymbol{x}_i) = \frac{\sum_{\boldsymbol{x}_j \in N_k(\boldsymbol{x}_i)} \mathrm{LRD}_k(\boldsymbol{x}_j)}{|N_k(\boldsymbol{x}_i)| \cdot \mathrm{LRD}_k(\boldsymbol{x}_i)}.$$

A data point is considered an outlier if its LOF score is sufficiently greater than 1, indicating that it has a substantially lower density than its neighbors. LOF is particularly useful for detecting anomalies in multimodal datasets where the density of data clusters varies significantly across the data space.

1.3.4.3 *One-Class Support Vector Machine (OC-SVM)*

One-Class Support Vector Machines, or OC-SVMs, are used for anomaly detection by finding the decision boundary that best separates

the normal data points from the rest of the feature space [52]. The algorithm constructs a hyperplane in a high-dimensional space such that the majority of the data points lie on one side of the hyperplane, while suspected anomalies lie on the opposite side.

The OC-SVM builds heavily on the traditional support vector machine used in classification. We will not explain support vector machines from the ground up, but rather focus on the extensions needed for anomaly detection. The interested reader is refered to the seminal work of Schölkopf and Smola [51]. Mathematically, OC-SVM solves the following optimization problem:

$$\min_{\mathbf{w},\rho,\xi_i} \left(\frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu m}\sum_{i=1}^{m} \xi_i - \rho \right),$$

subject to

$$(\mathbf{w} \cdot \phi(\mathbf{x}_i)) \geqslant \rho - \xi_i, \quad \xi_i \geqslant 0, \quad \forall i = 1, \ldots, m,$$

where $\mathbf{w}$ is the normal vector to the hyperplane, $\phi(x_i)$ is the mapping of the data point x_i to a higher-dimensional space, ξ_i are slack variables, ρ is the offset, and ν is a parameter between 0 and 1 that controls the trade-off between maximizing the margin and minimizing the number of misclassifications.

A data point x_i is considered an anomaly if it lies on the side of the hyperplane where the majority of the data does not reside. OC-SVMs are very flexible with respect to the possible multidimensional and multimodal distributions that can be modelled, but they are highly sensitive to the hyperparameter ν and the choice of kernel.

1.3.4.4 *Support Vector Data Description*

Support Vector Data Description, or SVDD, is a support vector based method developed concurrently with OC-SVM. It can similarly be used for anomaly detection. Rather than finding a separating hyperplane, it finds the smallest hypersphere in a high-dimensional space that encloses the majority of the data points [56]. The objective is to minimize the volume of the hypersphere while allowing some flexibility for anomalies.

Mathematically, SVDD solves the following optimization problem:

$$\min_{R,\mathbf{a},\xi_i} \left(R^2 + \frac{C}{m}\sum_{i=1}^{m} \xi_i \right),$$

subject to

$$\|\phi(\mathbf{x}_i) - \mathbf{a}\|^2 \leqslant R^2 + \xi_i, \quad \xi_i \geqslant 0 \quad \forall i = 1, \ldots, m,$$

where R is the radius of the hypersphere, $\mathbf{a}$ is the center of the hypersphere, $\phi(x_i)$ is the mapping of the data point $\mathbf{x}_i$ to a higher-dimensional space, ξ_i are slack variables, and C is a regularization

parameter that controls the trade-off between the volume of the hypersphere and the allowance for outliers.

A data point x_i is considered an anomaly if its distance from the center of the hypersphere exceeds the radius R.

The one-class support vector machine is equivalent to SVDD when the training data is of unit norm [56].

SVDD can be extended to learn from anomalies as well, which can be useful if labels are available in the semi-supervised, supervised, or active learning settings. In this case, the optimization is extended to include anomalies as well:

$$\min_{R,\mathbf{a},\xi_i} \left(R^2 + \frac{C}{m} \sum_{i=1}^{m} \xi_i + \frac{C}{m'} \sum_{i'=1}^{m'} \xi_{i'} \right),$$

subject to

$$\|\phi(x_i) - \mathbf{a}\|^2 \leqslant R^2 + \xi_i, \quad \xi_i \geqslant 0 \quad \forall i = 1, \ldots, m, \text{and}$$

$$\|\phi(x_{i'}) - \mathbf{a}\|^2 \geqslant R^2 + \xi_{i'}, \quad \xi_{i'} \geqslant 0 \quad \forall i' = 1, \ldots, m',$$

where we now use index $i \in 1, \ldots, m$ to denote the normal samples, and index $i' \in 1, \ldots, m'$ to denote the anomalies.

1.3.4.5 *PCA*

PCA can be used to detect anomalies by reducing the dimensionality, and reconstructing the data from a lower-dimensional representation. We factorize a mean-centered matrix $\mathbf{X}$ as $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ using singular value decomposition (SVD), where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices containing the left- and right-singular vectors of $\mathbf{X}$, respectively, and $\mathbf{\Sigma}$ is a diagonal scaling matrix. We can then project the data matrix $\mathbf{X}$ onto the first d principal components by multiplying: $\mathbf{Y} = \mathbf{X}\mathbf{V}_d$. By then transforming the data back we get a reconstruction of the original data: $\hat{\mathbf{X}} = \mathbf{Y}\mathbf{V}_d^T$. We can then calculate the reconstruction loss for each sample by calculating the mean squared error, or MSE: $\text{MSE}(x_i, \hat{x}_i) = \frac{1}{n} \sum_{j=1}^{n} \left(x_{i,j} - \hat{x}_{i,j}\right)^2$ The assumption of using the reconstruction loss for anomaly detection is that anomalies are harder to reconstruct than normal data, and will thus have a higher MSE. However, PCA is sensitive to the presence of anomalies, as the eigenvectors will be heavily influenced by them.

In Chapter 3 we further study reconstruction-based methods such as PCA and autoencoders. There we specifically question the assumption that anomalies are harder to reconstruct than normal data.

1.3.4.6 *Isolation Forest*

Isolation Forest is an ensemble anomaly detection method that isolates data points by recursively partitioning the data space using random splits [34, 35]. The key idea is that anomalies are rare and differ from most data, thus they are easier to isolate. Isolation Forest creates an ensemble of isolation trees. Each tree is trained on a subsample of the data, and on a subset of variables. A tree is trained by randomly selecting a variable from the subset and then randomly selecting a split value between the minimum and maximum values of the selected feature. The samples are then divided across the two child nodes based on whether the sample is below or above the split value. This process is repeated until either the maximum tree depth is reached, or a node contains only a single sample.

The anomaly score for a data point is based on the path length from the root of the tree to the node where the sample ends. Since anomalies are easier to isolate, they tend to have shorter path lengths. The average path length over all trees in the forest is used to compute the anomaly score.

The anomaly score for a data point $\mathbf{x}_i$ is then given by:

$$\text{IsolationScore}(\mathbf{x}_i) = \mathbb{E}(h(\mathbf{x}_i)),$$

where $\mathbb{E}(h(\mathbf{x}_i))$ is the average path length of x_i across all the trees in the forest.

In this case, the lower the anomaly score, the more likely a sample is an anomaly, as it is more easily isolated from other data.

Different weighting schemes exist for this score which makes it possible to compare scores across models with different hyperparameters [34, 35].

Isolation Forests have a lower computational complexity and are therefore easy to apply in practice. They also deal well with clustered anomalies.

1.3.4.7 *Deep Learning*

Since the advent of deep learning, many researchers have started applying these principles in anomaly detection, with varying degrees of success. Most of the proposed methods rely on the underlying concepts and methods we presented in this section. They adapt these methods by making use of the capability of neural networks to perform feature extraction and effectively capture spatial or temporal information. As there are a plethora of methods used, we will give a bird's eye view of some of the more popular methods. While we will not go into explicit detail about the architectures, it should be noted that these methods can easily be adapted to be applicable on

tabular, image, time series, or video data. These methods are generally most popular within the computer vision domain when making use of convolutional layers.

AUTOENCODERS Autoencoders are very similar to PCA when used for anomaly detection [5, 28], as they both rely on dimensionality reduction and reconstruction. They are one of the most popular methods for deep learning anomaly detection, with many available variations for different use cases across tabular, image, time-series, and video data [11, 49, 50, 60]. Autoencoders learn to find a lower-dimensional encoding $\mathbf{Y}$, e.g. $d < n$, in the encoding space $\mathcal{Y} = \mathbb{R}^d$ by applying the function $g : \mathcal{X} \to \mathcal{Y}$. they then decode $\mathbf{Y}$ by transforming it back into the space $\mathcal{X}$ through the decoder $h : \mathcal{Y} \to \mathcal{X}$, yielding the reconstructed data $\hat{\mathbf{X}}$. Summarizing, they learn the concrete transformations $\mathbf{X} \xrightarrow{g} \mathbf{Y} \xrightarrow{h} \hat{\mathbf{X}}$ whilst minimizing the reconstruction loss: $\mathcal{L}_R(\mathbf{b}_{enc}, \mathbf{b}_{dec}; \mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{mn} \sum_{i=1}^{m} |\mathbf{x}_i^T - \hat{\mathbf{x}}_i^T|^2$, where m and n indicate the number of samples and features respectively. When using linear activation functions, this will find a solution in the same space as PCA. When using non-linear activation functions, it can be seen as a non-orthogonal, non-linear generalization of PCA. Anomalies are detected in much the same way, by looking at the sample MSE, where a higher MSE is assumed to be indicative of an anomaly. In recent years, some researchers have questioned whether the assumption that a higher MSE is indicative of an anomaly is valid. Some methods have been proposed to fix reported issues, but no works have gone into detail to show how and when autoencoders might fail to detect anomalies. We have endeavored to further elucidate this purported unreliability of autoencoders so as to provide a scaffold for fixing observed issues. This in-depth study is presented in Chapter 3.

VARIATIONAL AUTOENCODERS Variational autoencoders are an extension of the autoencoder and add a term to the reconstruction loss, the Kullback-Leibler divergence [31] in order to regularize the distribution of the latent variables to be close to achosen prior distribution. Similar to autoencoders the VAE consists of an encoder that maps the input data $\mathbf{X}$ to a distribution over the latent variables $\mathbf{Y}$, and a decoder that maps $\mathbf{Y}$ back to the data space.

The VAE optimizes on the Evidence Lower Bound, or ELBO, composed of two terms: the reconstruction loss and the Kullback-Leibler divergence. The ELBO is expressed as:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q(\mathbf{Y}|\mathbf{X})}[\log p(\mathbf{X}|\mathbf{Z})] - \text{KL}(q(\mathbf{Y}|\mathbf{X}) \parallel p(\mathbf{Y})),$$

where $q(\mathbf{Y}|\mathbf{X})$ is the approximate posterior, $p(\mathbf{X}|\mathbf{Y})$ is the likelihood, and $p(\mathbf{Y})$ is the prior over the latent variables. The first term represents the reconstruction loss, ensuring that the data can be accurately reconstructed from the latent variables. The second term is the KL

divergence, which regularizes the distribution of the latent variables to be close to the prior distribution.

In anomaly detection using VAEs, data points that have a high reconstruction error or a low likelihood under the model are considered anomalies. Sometimes, only the reconstruction error or the likelihood, but not both, is used [3].

DEEP SUPPORT VECTOR DATA DESCRIPTION (DEEPSVDD) Deep Support Vector Data Description, or DeepSVDD, extends the traditional SVDD approach of anomaly detection by using deep learning to learn a lower-dimensional representation [48]. The core idea is to map the data into a lower-dimensional space, much like an autoencoder, using a deep neural network, and then find a minimum volume hypersphere that encloses the majority of the mapped data points in this feature space. Two variations exist: soft-boundary DeepSVDD, which is suited for unsupervised anomaly detection, and one-class DeepSVDD, which is suited for semi-supervised anomaly detection.

For soft-boundary DeepSVDD we optimize on the following objective function:

$$\min_{R,\Theta} \left(R^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \max(0, \|\phi_\Theta(\mathbf{x}_i) - \mathbf{c}\|^2 - R^2) + \frac{\lambda}{2} \sum_{l=1}^{L} \|\mathbf{W}^l\|_F^2 \right),$$

where R is the radius of the hypersphere, $\mathbf{c}$ is the center of the hypersphere in the lower-dimensional representation, $\phi_\Theta(\mathbf{x}_i)$ is the feature representation of the data point $\mathbf{x}_i$ learned by the neural network with parameters Θ, n is the number of data points, ν is a hyperparameter that controls the trade-off between the volume of the hypersphere and the penalty for points outside the hypersphere, $\|\mathbf{W}^l\|_F^2$ is the Frobenius norm of the weights of the l-th layer of the network, and λ is the weighting term of the weight decay regularization in the last term.

For the one-class DeepSVDD, the objective function is simplified to:

$$\min_{\Theta} \frac{1}{n} \sum_{i=1}^{n} \|\phi_\Theta(\mathbf{x}_i) - \mathbf{c}\|^2 + \frac{\lambda}{2} \sum_{l=1}^{L} \|\mathbf{W}^l\|_F^2.$$

The one-class DeepSVDD employs a quadratic loss for penalizing the distance of every network representation $\phi_\Theta(\mathbf{x}_i)$ to the center $\mathbf{c}$, along with a weight decay regularizer with hyperparameter λ. This approach assumes that most of the training data is normal and focuses on minimizing the mean distance of all data representations to the center.

In either variation a data point is considered to be more anomalous the further it is away from the center of the hypersphere i.e., $s_i = \|\phi_\Theta(\mathbf{x}_i) - \mathbf{c}\|^2$. In soft-boundary DeepSVDD we can also calculate the score as $s_i = \|\phi_\Theta(\mathbf{x}_i) - \mathbf{c}\|^2 - R^2$. This means that if the distance of the point to the center exceeds the radius, the score is above 0, and subsequently considered anomalous. DeepSVDD does not suffer from the reconstruction issues that trouble autoencoders that we describe in Chapter 3. However, training and using DeepSVDD models is still hard. To prevent hypersphere collapse, DeepSVDD uses no biases, which makes training slow, and can cause the model to fail to converge.

1.4 CONTRIBUTIONS

In this work we have identified several gaps in the current state of anomaly detection. We have made several contributions aiming to alleviate these issues. Furthermore, we have applied anomaly detection to a practical use case related to the operation of the Dutch power grid.

1.4.1 *Unsupervised Anomaly Detection Algorithms on Real-world Data: How Many Do We Need?*

Researchers, engineers applied scientists, and other users need guidelines on which anomaly detection algorithms work well in practice. We specifically want to answer the research question:

> 1. Which anomaly detection algorithms perform well on real-world data?

In Chapter 2 we present the largest unsupervised anomaly detection benchmark study to date to answer this question. By performing a benchmark study without proposing new or self-developed methods, we can compare anomaly detection algorithms in an unbiased manner. By comparing 33 algorithms across 52 real-world tabular datasets, we find a subset of algorithms that perform well across the board. Specifically kNN and the Extended Isolation Forest significantly outperform the highest number of other methods on data with "local" and "global" anomalies respectively and are therefore good "first picks" when starting with anomaly detection. While studying the properties of anomalies we have found that there is no clear consensus on what properties anomalies can have. We further contribute to the understanding of anomaly detection by proposing a new taxonomy of anomaly types, aiming to move away from a binary categorization to distinct properties on gradual scales.

Chapter 2 has been published in the Journal of Machine Learning Research [7].

1.4.2 *Autoencoders for Anomaly Detection are Unreliable*

In Chapter 3 we study one of the most popular deep learning methods in anomaly detection: autoencoders. As we have noted in Section 1.3.4.7 this type of neural network is commonly applied under the assumption that "Anomalies are harder to reconstruct than normal data". Our second research question is then:

> 2. Is it reasonable to assume that anomalies are harder to reconstruct than normal data?

In this chapter we show that this assumption does not hold in practice for many trained autoencoders. In fact, anomalies can often be nearly perfectly be reconstructed by an autoencoder, even when it has seen no anomalies during the training phase. When this happens anomalies can in the worst case look to be less anomalous than all analyzed normal data. By studying this problem in detail, we hope to dissuade researchers from using autoencoders in safety critical applications, and to provide a scaffold for future research into perhaps solving this limitation.

Chapter 3 can be found as a preprint on ArXiv [8].

1.4.3 *Acquiring Better Load Estimates by Combining Anomaly and Change Point Detection in Power Grid Time Series Measurements*

In Section 1.1.2 we have discussed several applications of anomaly detection within the energy domain. In Chapter 4 we present a novel practical use case of anomaly detection to power grid time-series measurements. In this use case we want to improve the detection of measurement errors and switch events and automate it. Accomplishing this could allow for better load estimates, which are needed for planning power grid expansions. This use case motivates the third and final research question:

> 3. Can anomaly detection be used to automate and improve the filtering procedure in power grid load measurements?

By performing extensive baseline corrections on time series, we are able to apply tried-and-true methods such as binary segmentation and statistical process control in order to automatically filter anomalies and switch events, which are long baseline changes due to power rerouting, from the data. By performing this filtering, Dutch distribution system operator Alliander can acquire better estimates of the actual load of hundreds of routes across the Netherlands. We perform extensive hyperparameter tuning and validation to get an estimate of how well the best model performs in practice on new data. By combining relatively simple and interpretable models, the results generated

by the novel methodology can be interpreted by domain experts and explained to stakeholders.

Chapter 4 has been published at Sustainable Energy Grids And Networks [9].

REFERENCES

[1] M. Ahmed, N. Choudhury, and S. Uddin. "Anomaly detection on big data in financial markets." In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. 2017, pp. 998–1001.

[2] M. Ahmed, A. N. Mahmood, and M. R. Islam. "A survey of anomaly detection techniques in financial domain." In: *Future Generation Computer Systems* 55 (2016), 278–288.

[3] J. An and S. Cho. "Variational autoencoder based anomaly detection using reconstruction probability." In: *Special Lecture on IE* 2.1 (2015), pp. 1–18.

[4] M. Astrid, M. Z. Zaheer, J.-Y. Lee, and S.-I. Lee. "Learning not to reconstruct anomalies." In: *arXiv preprint arXiv:2110.09742* (2021).

[5] P. Baldi and K. Hornik. "Neural networks and principal component analysis: Learning from examples without local minima." In: *Neural Networks* 2.1 (1989), pp. 53–58.

[6] S. Ball. "Harnessing unlabelled data for automatic aerial poacher detection: Reducing annotation costs through unsupervised and self-supervised learning." MA thesis. University of the Witwatersrand, Johannesburg, 2023.

[7] R. Bouman, Z. Bukhsh, and T. Heskes. "Unsupervised Anomaly Detection Algorithms on Real-world Data: How Many Do We Need?" In: *Journal of Machine Learning Research* 25.105 (2024), pp. 1–34. URL: http://jmlr.org/papers/v25/23-0570.html.

[8] R. Bouman and T. Heskes. *Autoencoders for Anomaly Detection are Unreliable*. 2025. arXiv: 2501.13864 [cs.LG]. URL: https://arxiv.org/abs/2501.13864.

[9] R. Bouman, L. Schmeitz, L. Buise, J. Heres, Y. Shapovalova, and T. Heskes. "Acquiring better load estimates by combining anomaly and change point detection in power grid time-series measurements." In: *Sustainable Energy, Grids and Networks* 40 (2024), p. 101540. ISSN: 2352-4677. DOI: https://doi.org/10.1016/j.segan.2024.101540. URL: https://www.sciencedirect.com/science/article/pii/S2352467724002698.

[10] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. "LOF: identifying density-based local outliers." In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. 2000, pp. 93–104.

[11] J. K. Chow, Z. Su, J. Wu, P. S. Tan, X. Mao, and Y.-H. Wang. "Anomaly detection of defects on concrete structures with the convolutional autoencoder." In: *Advanced Engineering Informatics* 45 (2020), p. 101105.

[12] M. Das and S. Parthasarathy. "Anomaly detection and spatio-temporal analysis of global climate system." In: *Proceedings of the Third International Workshop on Knowledge Discovery From Sensor Data*. 2009, pp. 142–150.

[13] R. B. Dean and W. J. Dixon. "Simplified statistics for small numbers of observations." In: *Analytical Chemistry* 23.4 (1951), pp. 636–638.

[14] D. Divya, B. Marath, and M. Santosh Kumar. "Review of fault detection techniques for predictive maintenance." In: *Journal of Quality in Maintenance Engineering* 29.2 (2023), pp. 420–441.

[15] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. "Do we need hundreds of classifiers to solve real world classification problems?" In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3133–3181.

[16] T. Fernando, H. Gammulle, S. Denman, S.-h. Sridharan, and C. Fookes. "Deep learning for medical anomaly detection–A survey." In: *ACM Computing Surveys (CSUR)* 54.7 (2021), pp. 1–37.

[17] R. Folcarelli, S. Van Staveren, R. Bouman, B. Hilvering, G. H. Tinnevelt, G. Postma, O. F. Van Den Brink, L. M. Buydens, N. Vrisekoop, L. Koenderman, et al. "Automated flow cytometric identification of disease-specific cells by the ECLIPSE algorithm." In: *Scientific Reports* 8.1 (2018), p. 10907.

[18] B. Frénay and M. Verleysen. "Classification in the presence of label noise: A survey." In: *IEEE Transactions on Neural Networks and Learning Systems* 25 (2013), pp. 845–869.

[19] N. Goix. "How to evaluate the quality of unsupervised anomaly detection algorithms?" In: *arXiv preprint arXiv:1607.01152* (2016).

[20] K. Golmohammadi and O. R. Zaiane. "Time series contextual anomaly detection for detecting market manipulation in stock market." In: *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE. 2015, pp. 1–10.

[21] F. E. Grubbs. *Sample criteria for testing outlying observations*. University of Michigan, 1949.

[22] F. R. Hampel. "The influence curve and its role in robust estimation." In: *Journal of the American Statistical Association* 69.346 (1974), pp. 383–393.

[23] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[24] H. He and A. Ghodsi. "Rare class classification by support vector machine." In: *2010 20th International Conference on Pattern Recognition*. IEEE. 2010, pp. 548–551.

[25] M. He and H. Chen. "Anomaly Detection in Species Distribution Patterns: A Spatio-Temporal Approach for Biodiversity Conservation." In: *Journal of Biobased Materials and Bioenergy* 18.1 (2024), pp. 39–50.

[26] W. Hilal, S. A. Gadsden, and J. Yawney. "Financial fraud: a review of anomaly detection techniques and recent advances." In: *Expert Systems With Applications* 193 (2022), p. 116429.

[27] D. J. Hill, B. S. Minsker, and E. Amir. "Real-time Bayesian anomaly detection for environmental sensor data." In: *Proceedings of the Congress-International Association for Hydraulic Research*. 2007.

[28] N. Japkowicz, C. Myers, M. Gluck, et al. "A novelty detection approach to classification." In: *International Joint Conference on Artificial Intelligence*. Vol. 1. Citeseer. 1995, pp. 518–523.

[29] P. Kamat and R. Sugandhi. "Anomaly detection for predictive maintenance in industry 4.0-A survey." In: *E3S Web of Conferences*. Vol. 170. EDP Sciences. 2020, p. 02007.

[30] P. Kanhere and H. Khanuja. "A methodology for outlier detection in audit logs for financial transactions." In: *2015 International Conference on Computing Communication Control and Automation*. IEEE. 2015, pp. 837–840.

[31] D. P. Kingma and M. Welling. "Auto-encoding variational Bayes." In: *arXiv preprint arXiv:1312.6114* (2013).

[32] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. "A comparative study of anomaly detection schemes in network intrusion detection." In: *Proceedings of the 2003 SIAM International Conference on Data Mining*. SIAM. 2003, pp. 25–36.

[33] D.-Y. Liao, C.-Y. Chen, W.-P. Tsai, H.-T. Chen, Y.-T. Wu, and S.-C. Chang. "Anomaly detection for semiconductor tools using stacked autoencoder learning." In: *2018 International Symposium on Semiconductor Manufacturing (ISSM)*. IEEE. 2018, pp. 1–4.

[34] F. T. Liu, K. M. Ting, and Z.-H. Zhou. "Isolation forest." In: *2008 eighth IEEE International Conference on Data Mining*. IEEE. 2008, pp. 413–422.

[35] F. T. Liu, K. M. Ting, and Z.-H. Zhou. "Isolation-based anomaly detection." In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.1 (2012), pp. 1–39.

[36] R. Longadge and S. Dongre. *Class Imbalance Problem in Data Mining Review*. 2013. eprint: arXiv:1305.1707.

[37] M. Q. Ma, Y. Zhao, X. Zhang, and L. Akoglu. "The need for unsupervised outlier model selection: A review and evaluation of internal evaluation strategies." In: *ACM SIGKDD Explorations Newsletter* 25.1 (2023), pp. 19–35.

[38] P. C. Mahalanobis. "On the generalised distance in statistics." In: *Proceedings of the National Institute of Science of India*. Vol. 12. 1936, pp. 49–55.

[39] H. O. Marques, R. J. Campello, J. Sander, and A. Zimek. "Internal evaluation of unsupervised outlier detection." In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14.4 (2020), pp. 1–42.

[40] R. Moghaddass and J. Wang. "A hierarchical framework for smart grid anomaly detection using large-scale smart meter data." In: *IEEE Transactions on Smart Grid* 9.6 (2017), 5820–5830.

[41] T. T. Nguyen, U. Q. Nguyen, et al. "An evaluation method for unsupervised anomaly detection algorithms." In: *Journal of Computer Science and Cybernetics* 32.3 (2016), pp. 259–272.

[42] J. Oakland and J. S. Oakland. *Statistical process control*. Routledge, 2007.

[43] S. H. Oh and W. S. Lee. "An anomaly intrusion detection method by clustering normal user behavior." In: *Computers & Security* 22.7 (2003), pp. 596–612.

[44] S. Ramaswamy, R. Rastogi, and K. Shim. "Efficient algorithms for mining outliers from large data sets." In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. 2000, pp. 427–438.

[45] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng. "A survey of predictive maintenance: Systems, purposes and approaches." In: *arXiv preprint arXiv:1912.07383* (2019).

[46] B. Rossi, S. Chren, B. Buhnova, and T. Pitner. "Anomaly detection in smart grid data: An experience report." In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2016, pp. 002313–002318.

[47] P. J. Rousseeuw and C. Croux. "Alternatives to the median absolute deviation." In: *Journal of the American Statistical Association* 88.424 (1993), pp. 1273–1283.

[48] L. Ruff, R. A. Vandermeulen, N. Görnitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft. "Deep One-Class Classification." In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. 2018, pp. 4393–4402.

[49] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut. "Network anomaly detection using LSTM based autoencoder." In: *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*. 2020, pp. 37–45.

[50] M. Sakurada and T. Yairi. "Anomaly detection using autoencoders with nonlinear dimensionality reduction." In: *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*. 2014, pp. 4–11.

[51] B. Schölkopf and A. J. Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[52] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. "Support vector method for novelty detection." In: *Advances in Neural Information Processing Systems* 12 (1999).

[53] L. Scime and J. Beuth. "Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm." In: *Additive Manufacturing* 19 (2018), pp. 114–126.

[54] R. Stribos, R. Bouman, L. Jimenez, M. Slot, and M. Stoelinga. "A comparison of anomaly detection algorithms with applications on recoater streaking in an additive manufacturing process." In: *Rapid Prototyping Journal* (2024).

[55] G. A. Susto, M. Terzi, and A. Beghi. "Anomaly detection approaches for semiconductor manufacturing." In: *Procedia Manufacturing* 11 (2017), pp. 2018–2024.

[56] D. M. Tax and R. P. Duin. "Support vector data description." In: *Machine Learning* 54 (2004), pp. 45–66.

[57] H. Y. Teh, I. Kevin, K. Wang, and A. W. Kempa-Liehr. "Expect the unexpected: Unsupervised feature selection for automated sensor anomaly detection." In: *IEEE Sensors Journal* 21.16 (2021), pp. 18033–18046.

[58] W.-K. Wong, A. W. Moore, G. F. Cooper, and M. M. Wagner. "Bayesian network anomaly pattern detection for disease outbreaks." In: *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 2003, pp. 808–815.

[59] S.-C. Yip, W.-N. Tan, C. Tan, M.-T. Gan, and K. Wong. "An anomaly detection framework for identifying energy theft and defective meters in smart grids." In: *International Journal of Electrical Power & Energy Systems* 101 (2018), pp. 189–203.

[60] Y. Zhao, B. Deng, C. Shen, Y. Liu, H. Lu, and X.-S. Hua. "Spatio-temporal autoencoder for video anomaly detection." In: *Proceedings of the 25th ACM International Conference on Multimedia*. 2017, pp. 1933–1941.

2

UNSUPERVISED ANOMALY DETECTION ALGORITHMS ON REAL-WORLD DATA: HOW MANY DO WE NEED?

In this chapter we evaluate 33 unsupervised anomaly detection algorithms on 52 real-world multivariate tabular data sets, performing the largest comparison of unsupervised anomaly detection algorithms to date. On this collection of data sets, the EIF (Extended Isolation Forest) algorithm significantly outperforms the most other algorithms. Visualizing and then clustering the relative performance of the considered algorithms on all data sets, we identify two clear clusters: one with "local" data sets, and another with "global" data sets. "Local" anomalies occupy a region with low density when compared to nearby samples, while "global" occupy an overall low density region in the feature space. On the local data sets the kNN (k-nearest neighbor) algorithm comes out on top. On the global data sets, the EIF (extended isolation forest) algorithm performs the best. Also taking into consideration the algorithms' computational complexity, a toolbox with these two unsupervised anomaly detection algorithms suffices for finding anomalies in this representative collection of multivariate data sets. By providing access to code and data sets, the research in this chapter can be easily reproduced and extended with more algorithms and/or data sets.

2.1 INTRODUCTION

Anomaly detection is the study of finding data points that do not fit the expected structure of the data. Anomalies can be caused by unexpected processes generating the data. In chemistry an anomaly might be caused by an incorrectly performed experiment, in medicine a certain disease might induce rare symptoms, and in predictive maintenance an anomaly can be indicative of early system failure. Depending on the application domain, anomalies have different properties, and may also be called by different names. Within the domain of machine learning (and hence also in this chapter), anomaly detection is often used interchangeably with outlier detection.

Unsupervised, data-driven, detection of anomalies is a standard technique in machine learning. Throughout the years, many methods, or algorithms, have been developed in order to detect anomalies. Some of these algorithms aim to solve specific issues, such as high dimensionality. Other methods try to detect anomalies in the general sense, and focus on high performance or low computational or memory complexity. Due to the many algorithms available, it is hard to determine which algorithm is best suited for a particular use case,

especially for a user who is not intimately familiar with the field of anomaly detection. More details on the specific algorithms evaluated in this study can be found in section 2.3.1.

Several studies have been performed to provide guidelines on when to apply which algorithm. Some review studies [37, 44], give advice based on the theoretical properties of the algorithms. In recent years several studies have been conducted that empirically compare a number of anomaly detection algorithms on a range of data sets.

Emmott et al. [14] study 8 well-known algorithms on 19 data sets. They find Isolation Forest to perform the best overall, but recommend using ABOD (Angle-Based anomaly Detection) or LOF (Local Outlier Factor) when there are multiple clusters present in the data.

Campos et al. [10] compare 12 k-nearest neighbours based algorithms, on 11 base data sets. They find LOF to significantly outperform a number of other methods, while KDEOS (Kernel Density Estimation anomaly Score) performs significantly worse than most algorithms.

Goldstein and Uchida [19] compare 19 algorithms on 10 data sets. Unlike Campos et al., Goldstein and Uchida perform no explicit optimization or selection, but rather evaluate the average performance over a range of sensible hyperparameter settings. With methods based on k-nearest neighbours generally giving stable results, Goldstein and Uchida recommend kNN (k-nearest neighbours) for global anomalies, LOF for local anomalies, and HBOS (Histogram-Based anomaly Selection) in general (see section 2.2.2 for an explanation of global and local anomalies). Goldstein and Uchida compare on a data set basis, without any overall statistical analysis.

More recently, Domingues et al. [12], apply 14 algorithms on 15 data sets, some of which are categorical. They find IF (Isolation Forest) and robust KDE (Kernel Density Estimation) to perform best, but note that robust KDE is often too expensive too calculate for larger data sets.

Steinbuss and Böhm [52] propose a novel strategy for synthesizing anomalies in real-world data sets using several statistical distributions as a sampling basis. They compare 4 algorithms across multiple data sets derived from 19 base data sets, both using the original and synthesized anomalies. They find kNN and IF to work best for detecting global anomalies, and LOF to work best for local and dependency anomalies. In the same year, Soenen et al. [51] study the effect of hyperparameter optimization strategies on the evaluation and propose to optimize hyperparameters on a small validation set, with evaluation on a much larger test set. In their comparison of 6 algorithms on 16 data sets, IF performs the best, closely followed by CBLOF/u-CBLOF ((unweighted-)Cluster-Based Local Outlier Factor) and kNN, while OCSVM (One-Class Support Vector Machine) performs worst

unless optimized using a substantially larger validation set than the other algorithms.

Han et al. [21] performed an extensive comparison of anomaly detection methods, including supervised and semi-supervised algorithms. They compare 14 unsupervised algorithms on 47 tabular data sets using out-of-the-box, that is suggested by algorithm author or implementation, hyperparameter settings. They subsample larger data sets to a maximum of 10.000 samples, duplicate samples for those data sets smaller than 1000 samples. They find no significant differences between unsupervised algorithms. While real-world data sets are being used, the anomalies in each data set are generated synthetically according to 4 different type definitions (see section 2.2.2), and they compare the performance for each different type. Additionally, they have analyzed more complex benchmark data sets used in CV and NLP, such as CIFAR10 [30] and the Amazon data set [24] by performing neural-based feature extraction.

Other studies are of a more limited scope, and cover for example methods for high-dimensional data [55], or consider only ensemble methods [59].

The studies done by Campos et al. [10], Domingues et al. [12], Goldstein and Uchida [19], Han et al. [21], Soenen et al. [51], and Steinbuss and Böhm [52] have several limitations when used as a benchmark. Firstly, with the exception of Han et al., all studies were done on a rather small collection of data sets. Secondly, these studies cover only a small number of methods. Campos et al. compare only kNN-based approaches, while Goldstein and Uchida fail to cover many of the methods that have gained traction in the last few years, such as IF [35] and variants thereof [22]. Soenen et al. consider just 6 commonly used methods, Steinbuss and Böhm cover 4 methods and Han et al. cover 14 unsupervised methods.

Some of these studies consider the performance on data sets containing specific types of anomalies, such as global or local anomalies. Specifically, Steinbuss and Böhm look at the performance of different algorithms on data sets containing synthesized global, local, and dependency anomalies. Similarly, Han et al. synthesize these three types of anomalies as well as cluster anomalies for use in their comparison. Goldstein and Uchida's study is, to the best of our knowledge, the only one that analyzes real-world, that is, non-synthesized global and local anomalies. In particular, they analyze the 'pen-local' and 'pen-global' data set, two variants of the same data set where different classes were selected to obtain local and global anomalies specifically.

In practice, very little is known regarding what types of anomalies are present in commonly used benchmark data sets, and thus large scale comparisons on real-world data for specific types of anomalies are still missing. In this study we apply a large number of commonly

used anomaly detection methods on a large collection of multivariate data sets, to discover guidelines on when to apply which algorithms. We explicitly choose to perform no optimization of hyperparameters, so as to evaluate the performance of algorithms in a truly unsupervised manner. Instead, we evaluate every algorithm over a range of sensible hyperparameters, and compare average performances. This contrasts with Soenen et al. [51], who perform extensive optimization on a small validation set and thereby supply guidelines for semi-supervised detection or active learning. Our approach rather is similar to that used by Domingues et al. [12], who also compare out-of-the-box performance. To the best of our knowledge, ours is the largest study of its kind performed so far.

2.2 BACKGROUND

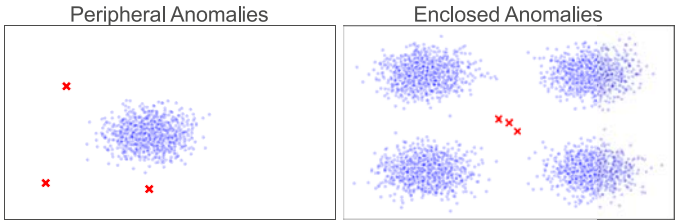
2.2.1 *Unsupervised Anomaly Detection*

Most anomaly detection tasks, including those done in this chapter, are conducted unsupervised. That means that no labels are available to the user. Consequently this means that regular optimization, like grid searches for optimal hyperparameters used in supervised learning, are not used within unsupervised anomaly detection. Most unsupervised anomaly detection algorithms produce scores, rather than labels, to samples. The most common convention is that a higher score indicates a higher likelihood that a sample is an anomaly, making unsupervised anomaly detection a ranking problem.

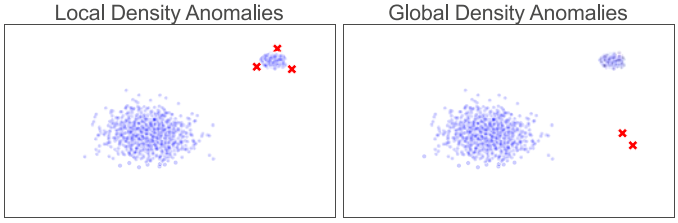
2.2.2 *Types of Anomalies*

Many different definitions of anomalies and their properties exist, many of these have been defined in an isolated context, not considering the relationships with other definitions or properties [9]. More recently, a review by Foorthuis [17] tried to unify definitions across multiple subdomains of anomaly detection in order to encompass all types of anomalies. These definitions however do not encompass many properties or types of anomalies, such as clustered or dependency anomalies.

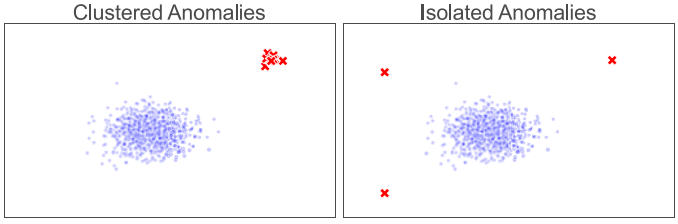
Rather than aiming to redefine every distinct type of anomaly, we treat anomalies as being able to have multiple, sometimes non-exclusive properties. Instead, we define four scales of non-exclusive properties which, when combined, encompass all types of anomalies found in multivariate tabular data in literature known to us. It should be noted that commonalities exist across these properties. For example the peripheral anomalies and isolated anomalies shown in Subfigures 1a and 1c have the same characteristics when considering no other properties.



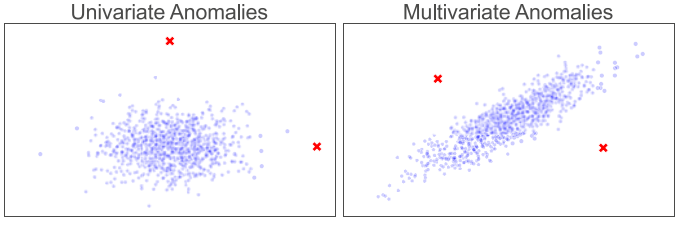
(a) Examples of peripheral (left) and enclosed (right) anomalies



(b) Examples of local (left) and global (right) anomalies



(c) Examples of clustered (left) and isolated (right) anomalies



(d) Examples of univariate (left) and multivariate (right) anomalies

Figure 1: Eight examples of different types of anomalies along the four defined property axes. Normal data are visualized as blue points, while anomalies are visualized as red crosses.

2.2.2.1 *Enclosed and Peripheral Anomalies*

Anomalies can be surrounded in the feature space by normal data. When this occurs, we define them as enclosed anomalies. On the other end of this axis, peripheral anomalies occupy the edges of the feature space, and have one or more attribute scores either below the minimum or above the maximum of the scores of the normal data region. Examples of both enclosed and peripheral anomalies can be found in Figure 1a.

2.2.2.2 *Global and Local Density Anomalies*

The most commonly discussed types of anomalies are the global and local anomalies. These definitions follow from the work of Breunig et al. [9]. Global anomalies are points which can be isolated from normal data because they occupy a globally low density region of the feature space. Local anomalies however, cannot be separated using just these criteria. This stems from the fact that density estimates are often imperfect, and based on a proxy measure, such as (average) distance. Local anomalies rather are located in regions with a density which is low compared to nearby normal regions, so just the distance as a proxy for density would fail to identify these points. Local anomalies occur when multiple clusters with differing density functions exist in the feature space. Examples of both global and local density anomalies can be found in Figure 1b . Specifically in the "local anomaly example", the red marked anomalies occupy a space close to the dense cluster, but where the density is low.

2.2.2.3 *Isolated and Clustered Anomalies*

Most often, anomalies are isolated, and are single datapoints without any additional, normal or anomaly, datapoints nearby. In many practical cases, anomalies are not that singular, and small groups of anomalies form clusters, leading to clustered anomalies. Clustered anomalies are closely related to the phenomenon known as “masking", where similar anomalies mask each other’s presence by forming a cluster [35]. Examples of both isolated and clustered anomalies can be found in Figure 1c.

2.2.2.4 *Univariate and Multivariate Anomalies*

Some anomalies are clearly univariate in nature. That is, they can be identified by just a single feature score in an anomalous range. Other anomalies are multivariate in nature, requiring a specific combination of feature scores to be identified as anomalies. These multivariate anomalies are also often called dependency anomalies, as they differ from the normal dependency, or causal, structure of the data. Exam-

ples of both isolated and clustered anomalies can be found in Figure 1d.

2.3 MATERIALS AND METHODS

We evaluate the effectiveness of 33 different algorithms, listed in Table 1. We evaluate each algorithm multiple times for each data set, each with a different set of sensible hyperparameters. The results are then averaged across hyperparameter settings, leading to a single average ROC-AUC score for each method-data set combination. We refrain from optimizing hyperparameters, for example, using cross-validation, to reflect the real-world situation in which no labels are available for training the models. It should be noted, that the neural network architectures covered in this study are optimized with loss functions not using the original class labels, autoencoders for example use the mean squared error between the original and reconstructed features. While unsupervised optimization of hyperparameters has been studied by Thomas, Gramfort, and Clémençon [54] it has not been applied to most algorithms considered in this study.

2.3.1 *Algorithms*

Of the 33 methods, 27 were used as implemented in the popular Python library for anomaly detection, PyOD [57]. As part of this research, we made several contributions to this open source library, such as a memory-efficient implementation of the COF (Connectivity-based Outlier Factor) method, as well as an implementation of the Birgé and Rozenholc [7] method for histogram size selection, which is used in HBOS and LODA (lightweight on-line detector of anomalies). For EIF (extended isolation forest) we used the implementation provided by the authors in the Python package "eif" by Hariri, Kind, and Brunner [22]. We implemented the ODIN (Outlier Detection using Indegree Number) method in Python, and it is being prepared as a submission to the PyOD package. The ensemble-LOF method, which implements the LOF score calculation in line with the original paper [9] was implemented using the base LOF algorithm from PyOD. DeepSVDD is applied based on the publicly available code by its author Lukas Ruff[1], and we modified it to work on general tabular data sets. The DynamicHBOS method was applied using the code by Kanatoko[2]

We left out several of the implemented methods in the PyOD package, such as LOCI and ROD, because they have a time or memory

1 The DeepSVDD Git repository can be found at `https://github.com/lukasruff/Deep-SVDD-PyTorch`.

2 The DynamicHBOS Git repository can be found at `https://github.com/Kanatoko/HBOS-python`.

complexity of $\mathcal{O}(n^3)$, with n the number of data points. The PyOD SOS method was also ignored, due to its $\mathcal{O}(n^2)$ memory requirement. None of these methods performed notably well compared to the other algorithms that we included on the smaller data sets where evaluation was feasible. We have thoroughly optimized several of the slower methods in PyOD, specifically the SOD, COF, and LMDD methods.

2.3.2 *Data*

2.3.2.1 *Datasets*

In this study we consider a large collection of data sets from a variety of sources. We focus on real-valued, multivariate, tabular data, comparable to the data sets used by Campos et al. [10], Domingues et al. [12], Fernández-Delgado et al. [16], Goldstein and Uchida [19], and Soenen et al. [51]. Table 2 contains a summary of the data sets, listing each data set's origin, number of samples, number of features, number and percentage of anomalies. While we recognize that other types of data, such as timeseries, categorical, or visual, are of interest, they cannot be readily compared in a single study.

Our collection consists for the most part of data sets from the ODDS data collection [41], specifically the multi-dimensional point data sets. It is a collection of various data sets, mostly adapted from the UCI machine learning repository [13]. All data sets are real-valued, without any categorical data. Curation of this collection is sadly not fully up-to-date, causing some of the listed data sets to be unavailable. The unavailable data sets were omitted from this comparison.

In addition to the ODDS data set, we also incorporate publicly available data sets used in earlier anomaly detection research. These include several data sets from the comparison by Goldstein and Uchida [19], from the comparison of Campos et al. [10] using ELKI [47], from a study on Generative Adversial Active Learning, or GAAL [36], from a study on extended Autoencoders [48], from the ADBench comparison [21], and from a study on Efficient Online Anomaly Detection (EOAD) [8]. data sets from these latter sources that are (near-)duplicates of data sets present in the ODDS collection are left out. In Table 2 we specify exactly where each data set was downloaded or reconstructed from.

Emmott et al. [14] and Emmott et al. [15] present a systematic methodology to construct anomaly detection benchmarks, which is then also extensively applied by Pevnỳ [39]. In this chapter, we chose not to construct our own benchmark data sets, which inevitably leads to some arbitrariness and possibly bias, but instead we rely on a large collection of different data sets used in earlier comparison studies. Synthetic data sets are not included in this study, as real-world data sets are generally considered the best available tool for benchmark-

Name	Hyperparameters	Publication
ABOD	FastABOD, $k = 60$	[29]
AE	$n_{layers} = 1, 2, 3$, shrinkage factor= 0.2, 0.3, 0.5	[27]
ALAD	$n_{layers} = 3$, shrinkage factor= 0.2, 0.3, 0.5	[56]
CBLOF	$k = 2, 5, 10, 15$, $\alpha = 0.7, 0.8, 0.9$, $\beta = 3, 5, 7$	[25]
COF	$k = 5, 10, 15, 20, 25, 30$	[53]
COPOD		[33]
DeepSVDD	$n_{layers} = 1, 2, 3$, shrinkage factor= 0.2, 0.3, 0.5	[45]
DynamicHBOS		[18]
ECOD		[34]
EIF	$n_{trees} = 1000$, $n_{samples} = 128, 256, 512, 1024$, no replacement, extension levels: 1, 2, 3	[22]
ensemble-LOF	maximum LOF score over $k = 5, ..., 30$	[9]
gen2out		[32]
GMM	$n_{gaussians} = 2, ..., 14$	[1]
HBOS	n_{bins} based on Birgé-Rozenholc criterium	[18]
IF	$n_{trees} = 1000$, $n_{samples} = 128, 256, 512, 1024$, no replacement	[35]
INNE	200 estimators	[5]
KDE	Gaussian kernel	[31]
kNN	$k = 5, 8, ..., 29$, mean distance	[40]
kth-NN	$k = 5, 8, ..., 29$, largest distance	[40]
LMDD	$n_{shuffles} = 100$, MAD dissimilarity function	[4]
LODA	n_{bins} based on Birgé-Rozenblac criterium, 100 random projections	[39]
LOF	$k = 5, 8, ..., 29$	[9]
LUNAR	$k = 5, 10, 15, 20, 25, 30$	[20]
MCD	subset fraction= 0.6, 0.7, 0.8, 0.9	[43]
OCSVM	RBF kernel, $\nu = 0.5, 0.6, 0.7, 0.8, 0.9$, $\gamma = 1/d$	[46]
ODIN	$k = 5, 8, ..., 29$	[23]
PCA	selected PCs explain $> 30, 50, 70, 90\%$ of variance	[49]
sb-DeepSVDD	$n_{layers} = 1, 2, 3$, shrinkage factor= 0.2, 0.3, 0.5	[45]
SOD	$k = 20, 30$, $l = 10, 18$, $\alpha = 0.7, 0.9$	[28]
SO-GAAL	stop epochs= 50	[36]
u-CBLOF	$k = 2, 5, 10, 15$, $\alpha = 0.7, 0.8, 0.9$, $\beta = 3, 5, 7$	[2]
VAE	$n_{layers} = 1, 2, 3$, shrinkage factor= 0.2, 0.3, 0.5	[3]
β-VAE	$n_{layers} = 1, 2, 3$, shrinkage factor= 0.2, 0.3, 0.5, $\gamma = 10, 20, 50$	[58]

Table 1: Overview of the algorithms, the setting of the hyperparameters, the year of original publication, and the author(s). For the neural networks, the "shrinkage factor" hyperparameter indicates that any subsequent layer in the encoder is defined by: layer size$_{n+1}$ = layer size$_n$ $\times$ shrinkage factor.

ing algorithms [12, 14, 44]. While real-world data sets are preferred for benchmarking, we note the usefulness of synthetic data in when studying specific properties of anomaly detection algorithms.

2.3.2.2 *Preprocessing*

Several steps have been undertaken to be able to compare the performance of the various algorithms on the different data sets. Most importantly all features in all data sets have been scaled and centered. Centering is done for each feature in a data set by subtracting the median. Scaling is performed by dividing each feature by its interquartile range. Our choice of centering and scaling procedure is deliberate, as both the median and interquartile range are influenced less by the presence of anomalies than the mean and standard deviation. This procedure is generally considered to be more stable than standardization when anomalies are known to be present [42]. Our choice of scaling is further motivated because although some algorithms, such as Isolation Forest, can implicitly handle features with different scales, methods that involve, for example, distance or cross-product calculations are strongly affected by the scale of the features.

2.3.3 *Evaluation Procedure*

In the unsupervised anomaly detection setting, it is generally more common and useful to evaluate anomaly scores, rather than binary labels as also produced by some algorithms. An anomaly score is a real-valued score, where a higher value indicates a higher likelihood, according to the score producing algorithm, that a specific sample is an anomaly. Using these scores, samples can be ranked according to apparent anomalousness, providing insights into the underlying nature of anomalies. For each data set, we calculate anomaly scores on all available data at once, without using any cross-validation or train-test splits, procedures common in the supervised setting. The scores from this unsupervised analysis are then compared to the ground truth labels, which indicates whether a sample is an anomaly (1), or not (0), to evaluate the performance of the algorithm. In order to compare the different algorithms we calculate the performance for each algorithm-data set combination in terms of the AUC (area under the curve) value resulting from the ROC (receiver operating characteristic) curve. This is the most commonly used metric in anomaly detection evaluations [10, 19, 55], which can be readily interpreted from a probabilistic view. We considered using other metrics, such as the R-precision or average precision and their chance-adjusted variants introduced by Campos et al. [10], but found these to be less stable, and harder to interpret.

For each data set we rank the AUC scores calculated from the scores produced by each algorithm. Following the recommendations

Name	Origin	#samples	#features	#outliers	%outliers	#removed features
aloi	Goldstein	49999	27	1507	(3.01%)	0
annthyroid	ODDS	7200	6	534	(7.42%)	0
arrhythmia	ODDS	452	257	66	(14.6%)	17
breastw	ADBench	683	9	239	(34.99%)	0
campaign	ADBench	41188	62	4640	(11.27%)	0
cardio	ODDS	1831	21	176	(9.61%)	0
cover	ODDS	286048	10	2747	(0.96%)	0
donors	ADBench	619326	10	36710	(5.93%)	0
fault	ADBench	1941	27	673	(34.67%)	0
glass	ODDS	214	9	9	(4.21%)	0
hepatitis	ELKI	80	19	13	(16.25%)	0
hrss_anomalous_optimized	ex-AE	19634	18	4517	(23.01%)	0
hrss_anomalous_standard	ex-AE	23645	18	5670	(23.98%)	0
http	ODDS	567498	3	2211	(0.39%)	0
internetads	ELKI	1966	1555	368	(18.72%)	0
ionosphere	ODDS	351	33	126	(35.9%)	0
landsat	ADBench	6435	36	1333	(20.71%)	0
letter	ODDS	1600	32	100	(6.25%)	0
magic.gamma	ADBench	19020	10	6688	(35.16%)	0
mammography	ODDS	11183	6	260	(2.32%)	0
mi-f	ex-AE	25286	40	2161	(8.55%)	5
mi-v	ex-AE	25286	40	3942	(15.59%)	5
mnist	ODDS	7603	78	700	(9.21%)	22
musk	ODDS	3062	166	97	(3.17%)	0
nasa	ex-AE	4687	32	755	(16.11%)	0
optdigits	ODDS	5216	62	150	(2.88%)	2
pageblocks	ELKI	5393	10	510	(9.46%)	0
parkinson	ELKI	195	22	147	(75.38%)	0
pen-global	Goldstein	808	16	90	(11.14%)	0
pen-local	Goldstein	6723	16	10	(0.15%)	0
pendigits	ODDS	6870	16	156	(2.27%)	0
pima	ODDS	768	8	268	(34.9%)	0
satellite	ODDS	6435	36	2036	(31.64%)	0
satimage-2	ODDS	5803	36	71	(1.22%)	0
seismic-bumps	ODDS	2584	21	170	(6.58%)	3
shuttle	ODDS	49097	9	3511	(7.15%)	0
skin	ADBench	245057	3	50859	(20.75%)	0
smtp	ODDS	95156	3	30	(0.03%)	0
spambase	GAAL	4206	57	1678	(39.9%)	0
speech	ODDS	3686	400	61	(1.65%)	0
stamps	ELKI	340	9	31	(9.12%)	0
thyroid	ODDS	3772	6	93	(2.47%)	0
vertebral	ODDS	240	6	30	(12.5%)	0
vowels	ODDS	1456	12	50	(3.43%)	0
waveform	GAAL	3442	21	99	(2.88%)	0
wbc	ODDS	378	30	21	(5.56%)	0
wbc2	ADBench	223	9	10	(4.48%)	0
wilt	ELKI	4819	5	257	(5.33%)	0
wine	ODDS	129	13	10	(7.75%)	0
wpbc	ADBench	198	33	47	(23.74%)	0
yeast	ADBench	1484	8	507	(34.16%)	0
yeast6	EOAD	1484	8	35	(2.36%)	0

Table 2: Summary of the 52 multivariate data sets used in our anomaly detection algorithm comparison: the colloquial name of the data set, origin of the data set, the number of samples, features, and anomalies, as well as the percentage of anomalies, and the number of removed features.

for the comparison of classifiers by Demšar [11], we use the Iman-Davenport statistic [26] in order to determine whether there is any significant difference between the algorithms. If this statistic falls below the desired critical value corresponding to a p-value of 0.05, we apply the Nemenyi post-hoc test [38] to then assess which algorithms differ significantly from each other.

In some of the visualizations in this chapter we plot the percentage of maximum AUC, defined as

$$\widetilde{\mathrm{AUC}}(a, d) = \frac{\mathrm{AUC}(a, d)}{\max_{a' \in A} \mathrm{AUC}(a', d)} \times 100\,,$$

with a one of the algorithms and d one of the data sets.

2.3.4 *Reproducibility*

In order to reproduce all our experiments, we have provided access to a public GitHub repository[3] containing the code, including the optimized PyOD methods, and data sets used for all experiments as well as for the production of all figures and tables presented in this chapter.

2.4 RESULTS

2.4.1 *Overall Algorithm Performance*

In order to gauge the performance, we evaluated each algorithm on each data set using the AUC measure corresponding to a ROC curve. To evaluate the performance across multiple sensible hyperparameters the AUC value for a given method is the average of the AUC of each hyperparameter setting evaluated. In our analysis, we found three data sets on which nearly every evaluated algorithm produced close to random results, that is with all AUC values between 0.4 and 0.6. These data sets, the 'hrss_anomalous_standard' and 'wpbc' data sets, were therefore excluded from further analysis. It is likely that these data sets contain no discernible anomalies. The existence of newer versions of these data sets, further motivates our choice of removal. Some data sets showed no AUC values above 0.6, but did show AUC values below 0.4. In these cases, the detector performs better when the labels are inverted. This behaviour was observed in the 'yeast', 'skin' and 'vertebral' data sets. The original construction of the latter two data sets was done based on treating the largest group of samples as the normal (label 0) class, and the smaller group as the anomaly (label 1) class, as is done commonly in anomaly detection research. Yet, for both these sets, the more heterogeneous group

3 The Git repository can be found at: `https://github.com/RoelBouman/outlierdetection`.

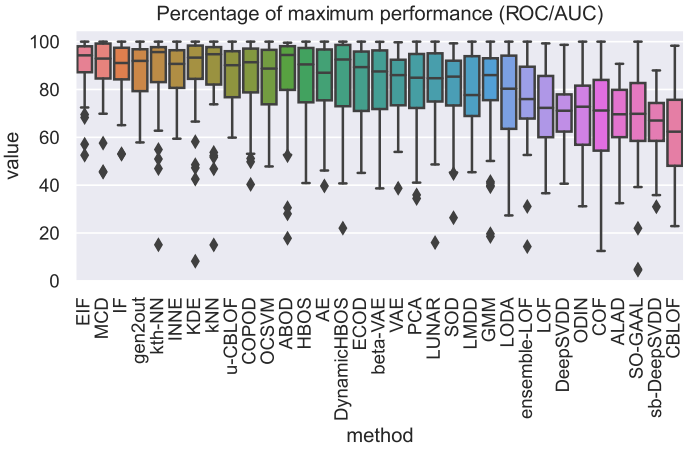


Figure 2: Boxplots of the performance of each algorithm on each data set in terms of percentage of maximum AUC. The maximum AUC is the highest AUC value obtained by the best performing algorithm on that particular data set. The whiskers in the boxplots extend 1.5 times the interquartile range past the low and high quartiles. data set-algorithm combinations outside of the whiskers are marked as diamonds.

is chosen as the normal class, in contrast to normal anomaly definitions. For these data sets, we inverted the labelling and recalculated the AUC values to be more in line with the general anomaly property that anomalies are more heterogenous than normal data.

Figure 2 shows the distribution of the performance for each method. In order to compare the AUC across different data sets, which might have different baseline performances, in a boxplot we express the AUC in terms of its percentage of the maximum AUC value obtained by the best performing algorithm on that particular data set.

It can be seen that many of the algorithms perform comparably, with a median percentage of maximum AUC around 90%. Several lower medians, as well as wider quartiles, can be observed.

To determine whether any of the observed differences in performance from Figure 2 are significant, we apply the Iman-Davenport test. This yielded a test-statistic of 16.395 , far above the critical value of 0.625 , thus refuting the null hypothesis. We then applied the Nemenyi post-hoc test to establish which algorithms significantly outperform which other algorithms. The results of this Nemenyi post-hoc test are summarized in Table 3.

Table 3 reveals that there are indeed several algorithms significantly outperforming many other algorithms. Most notable here is EIF, which significantly outperforms 14/16 algorithms evaluated in

	CBLOF	sb-DeepSVDD	SO-GAAL	ALAD	COF	ODIN	DeepSVDD	LOF	ensemble-LOF	LODA	GMM	LMDD	SOD	LUNAR	PCA	VAE	**Mean AUC**
EIF	++	++	++	++	++	++	++	++	++	++	+	++	++	+	++	++	0.770
MCD	++	++	++	++	++	++	++	++	++	++		++					0.762
IF	++	++	++	++	++	++	++	++	++	++		++					0.759
gen2out	++	++	++	++	++	++	++	++		+							0.747
kth-NN	++	++	++	++	++	++	++	++	++	++		++	++		+		0.745
INNE	++	++	++	++	++	++	++	++									0.743
KDE	++	++	++	++	++	++	++	++		++							0.743
kNN	++	++	++	++	++	++	++	++	++	++		++					0.741
u-CBLOF	++	++	++	++	++	++	++	+									0.740
COPOD	++	++	++	++	++	++	++	++									0.729
OCSVM	++	++	++	++	++	++	++										0.723
ABOD	++	++	++	++	++	++	++	++									0.721
HBOS	++	++	++	++	++	++	++										0.720
AE	++	++	++	++		+	++										0.719
DynamicHBOS	++	++	++	++	++	++	++										0.717
ECOD	++	++	++	++		+	++										0.713
beta-VAE	++	++	++	++			++										0.711
VAE	++	++	+	++			+										0.704
PCA	++	++		++													0.700
LUNAR	++	++	++	++			++										0.691
SOD	++	++		+													0.682
LMDD		++															0.675
GMM	++	++	++	++			+										0.664
LODA		+															0.658
ensemble-LOF		++															0.643
LOF																	0.617
DeepSVDD											-			--		-	0.596
ODIN																	0.588
COF																	0.586
ALAD											--		-	--	--	--	0.576
SO-GAAL											--			--		-	0.575
sb-DeepSVDD									--	-	--	--	--	--	--	--	0.551
CBLOF											--		--	--	--	--	0.515

Table 3: Significant differences between algorithms based on Nemenyi post-hoc analysis. ++/+ denotes that the row algorithm outperforms the column algorithm at $p = 0.05/p = 0.10$ respectively, while -- denotes that the row algorithm is outperformed by the column algorithm at $p = 0.05$. Rows and columns are sorted by descending and ascending mean performance, respectively. Columns are not shown when the column algorithm is not outperformed by any other algorithms at $p = 0.05$ or $p = 0.10$. The last column shows the mean AUC.

this study at the $p = 0.05/p = 0.10$ significance level respectively. Since the computational complexity of Isolation Forest and variants thereof scales linearly with the number of samples n, this may give them a clear further edge over methods such as kNN and derivatives for large data sets, with a computational complexity that scales quadratically or at best with $\mathcal{O}(n \log n)$ when optimized.

From Figure 2 and Table 3 we can also observe that the original CBLOF method is by far the worst performing method based on the mean AUC, being significantly outperformed by 22 at the $p = 0.05$ significance level. This corroborates the results of Goldstein and Uchida [19], who also found CBLOF to consistently underperform, while its unweighted variant, u-CBLOF, performs comparably to other algorithms. Notable is also the sb-DeepSVDD method, which performs slightly better in terms of mean AUC, but is significantly outperformed by 24/25 algorithms at the $p = 0.05/p = 0.10$ significance level respectively.

From these overall results it is clear that many of the neural networks do not perform well. (Soft-boundary) DeepSVDD, ALAD, and SO-GAAL all occupy the lower segment of overall method performance. We surmise that there are three likely reasons for this phenomenon. Firstly, these methods were not designed with tabular data in mind, and they can't leverage the same feature extraction capabilities that give them an edge on their typical computer vision tasks. Secondly, these methods are relatively complex, making it exceedingly hard to specify general hyperparameter settings and architectures which work on a large variety of data sets. Lastly, many of the data sets in this study are likely not sufficiently large enough to leverage the strengths of neural network approaches. Not all neural networks suffer from these problems equally, as the auto-encoder and variants, as well as LUNAR, perform about average. This is likely caused by a more straightforward architecture and optimisation criterion. More specifically, we suspect lack of convergence is a problem for the generative adversarial methods and DeepSVDD.

In addition to the neural networks, the local methods, such as LOF, ODIN, COF, and CBLOF, are some of the most underperforming methods. This result for LOF stands in stark contrast to the results of Campos et al. [10], who found LOF to be among the best performing methods. This is most likely caused by their evaluation on a small number of data sets with a low percentage of anomalies, which causes LOF to suffer less from swamping or masking [35]. We further study this finding in Section 2.4.3.

2.4.2 *Clustering Algorithms and data sets*

To visualize the similarities between algorithms on one hand, and the data sets on the other, Figure 3 shows a heatmap of the perfor-

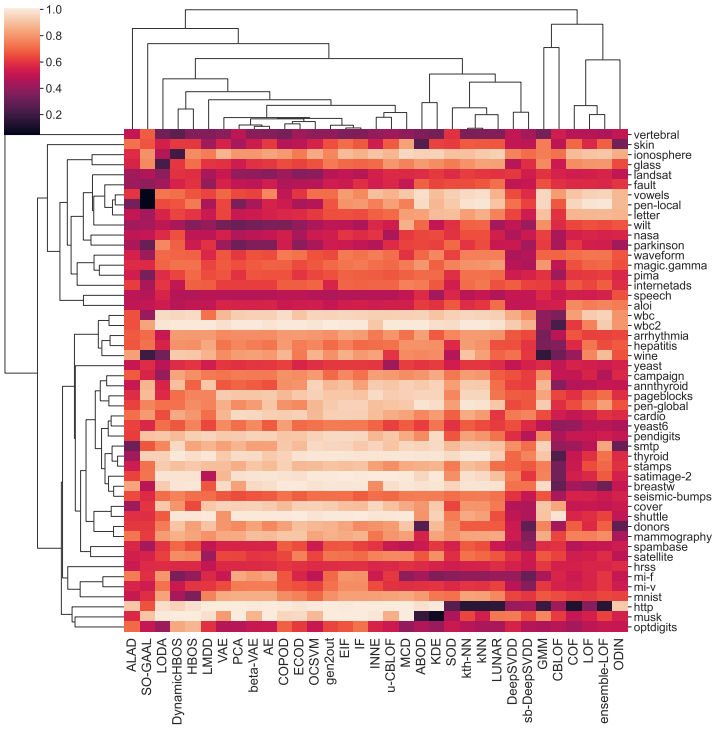


Figure 3: Clustered heatmap of the ROC/AUC performance of each algorithm. The algorithms and data sets are each clustered using hierarchical clustering with average linkage and the Pearson correlation as metric.

mance of each data set/algorithm combination and dendrograms of two hierarchical clusterings, one on the data sets, and one on the algorithms. For these clustering steps, the Pearson correlation was used as a distance measure, as this best shows how similar methods or data sets are when looking at the calculated performance. We furthermore used average linkage cluster analysis to construct more robust clusters. For the sake of visualisation the leaf orderings were optimized using the method of Bar-Joseph, Gifford, and Jaakkola [6].

Figure 3 shows that many similar algorithms cluster together in an expected way, with families of algorithms forming their own clusters. Some interesting patterns can be observed at a larger level. For the algorithms, we obtain several fairly distinct clusters. Firstly, CBLOF is distinct, as it underperforms on nearly every data set. Similarly, underperforming methods such as SO-GAAL, GMM, and ALAD only cluster together at a large distance, indicating that they have little correlation to other methods. The local methods, COF, ensemble-LOF, LOF and ODIN, form a separate cluster. These algorithms, which are

specifically designed to detect local anomalies, work well on a few (approximately a quarter) of the data sets, but do not perform well for most other data sets. We have a small cluster of kNN and related methods such as LUNAR and SOD, that performs decently for all data sets. Lastly, as can be seen seen in the top left half of Figure 3, a large cluster of methods seems to negatively correlate with the local methods, performing well for most (approximately three-quarters) of the data sets, but less so for the remainder.

The data sets split into two clearly distinct clusters: one cluster of data sets on which the local algorithms perform well, and another cluster of data sets on which the large cluster of algorithms performs well. Combining the two-way clustering with knowledge of the algorithms suggests that approximately one third of the data sets comprises so-called local problems, while the other two-thirds comprises global problems. This is corroborated by specifically constructed local and global sets 'pen-local' and 'pen-global', that clearly belong to their expected clusters. This observation is corroborated by research by Steinbuss and Böhm [52] and Emmott et al. [14], who similarly find differences between what they categorize as local/dependency and multi-cluster anomalies respectively, and global anomalies. We cannot clearly observe any other clear patterns of different anomaly properties arising from our analysis. The 'vertebral' data set furthermore seems distinct from either the global or local clusters.

To the best of our knowledge, no previous study on naturally occurring anomalies in real-world data has looked, in detail, into the difference between the performance of algorithms when specifically being applied to either global or local anomaly detection problems.

2.4.3 *Performance on Global and Local Problems*

In the previous section, we discovered a clear distinction between two clusters of data sets: one with the "local" data sets 'aloi', 'fault', 'glass', 'internetads', 'ionosphere', 'landsat', 'letter', 'magic.gamma', 'nasa', 'parkinson', 'pen-local', 'pima', 'skin', 'speech', 'vowels', 'waveform', and 'wilt', and another with the remaining "global" data sets, excluding 'vertebral',. Suspecting that different methods may do well on different types of data sets, we repeated the significance testing procedure from Section 2.4.1 for both clusters separately.

Performance boxplots for all algorithms applied on the collection of local data sets can be found in Figure 4. Figure 4 clearly shows the reversed performance of some of the local methods for anomaly detection. Where COF, ensemble-LOF, and LOF were among the worst performers over the entire collection, they are among the best performers when applied to the problems for which they were specifically developed. This phenomenon is a fine example of Simpson's

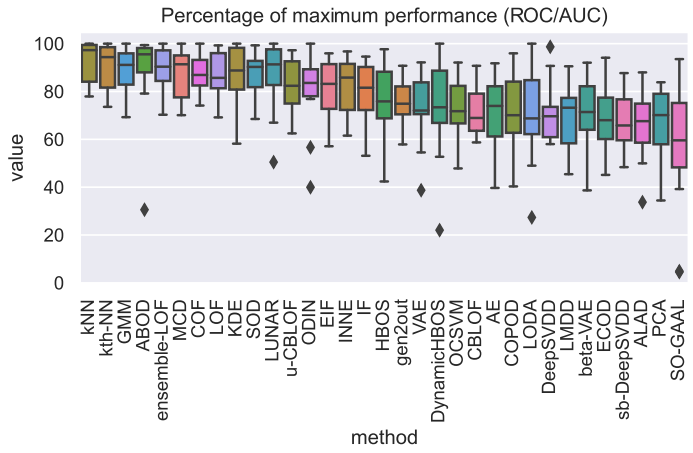


Figure 4: Boxplots of the performance of each algorithm on the "local" data sets in terms of percentage of maximum AUC. The maximum AUC is the highest AUC value obtained by the best performing algorithm on that particular data set. The whiskers in the boxplots extend 1.5 times the interquartile range past the low and high quartiles. data set-algorithm combinations outside of the whiskers are marked as diamonds.

paradox [50]. This also partially explains the difference in findings of our overall comparison and the comparison of Campos et al. [10].

We then repeated the Nemenyi-Friedman post hoc test on just the local data sets. The results for this analysis are summarized in Table 4. kNN is the top performers, and significantly outperform 17/18 other methods at the $p = 0.05/p = 0.10$ significance level respectively.

We then repeated the analysis for the global data sets, leading to the performance boxplots in Figure 5 and the significance results in Table 5.

From Figure 5 and Table 5 we can see that the Extended Isolation Forest has the highest mean performance, closely followed by the regular Isolation Forest. The Extended Isolation Forest outperforms 13/14 methods at $p = 0.05/p = 0.10$ respectively. Coincidentally, these methods also have the lowest computational and memory requirement, leaving them as the most likely choices for global anomalies.

2.5 DISCUSSION

In our study we compared the performance of anomaly detection algorithms on 52 semantically meaningful real-world tabular data sets, more than any other recent comparison studies [10, 12, 19, 21, 51, 52,

	SO-GAAL	PCA	ALAD	sb-DeepSVDD	ECOD	beta-VAE	LMDD	DeepSVDD	LODA	COPOD	AE	CBLOF	OCSVM	DynamicHBOS	VAE	gen2out	HBOS	ODIN	**Mean AUC**
kNN	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++	++	+	0.737
kth-NN	++	++	++	++	++	++	++	++	++	++	++	++	++	+	++	++	+		0.722
GMM	++	++	++	++	++	++	++	+			+								0.710
ABOD	++	++	++	++	++	++	++	++	++	++	++	++	++		++	++			0.709
ensemble-LOF	++	++	++	++	++	++	++	++		+	++	+	+						0.708
MCD	++	++	++	++	++	++	++	+			+								0.702
COF	++	++	++	++	++	++	++												0.698
LOF	++	++	++	++	++	++	++												0.695
KDE	++	++	++	++	++	++	++	+			+								0.693
SOD	++	++	++	++	++	++	++												0.693
LUNAR	++	++	++	++	++	++	++												0.693
u-CBLOF	++	++	++	+															0.657
ODIN																			0.646
EIF	++	++	++																0.646
INNE	++	+	+																0.644
IF	+																		0.637
HBOS																			0.609
gen2out																			0.601
VAE																			0.585
DynamicHBOS																			0.578
OCSVM																			0.573
CBLOF																			0.566
AE																			0.565
COPOD																			0.561
LODA																			0.555
DeepSVDD																			0.554
LMDD																			0.539
beta-VAE																			0.539
ECOD																			0.538
sb-DeepSVDD																			0.531
ALAD																			0.514
PCA																			0.513
SO-GAAL																			0.444

Table 4: Significant differences between algorithms on the collection of local problems based on Nemenyi post-hoc analysis. ++/+ denotes that the row algorithm outperforms the column algorithm at $p = 0.05/p = 0.10$. Rows and columns are sorted by descending and ascending mean performance, respectively. Columns are not shown when the column algorithm is not outperformed by any other algorithms at $p = 0.05$ or $p = 0.10$. The last column shows the mean AUC.

	CBLOF	COF	ODIN	sb-DeepSVDD	LOF	ALAD	ensemble-LOF	DeepSVDD	SO-GAAL	GMM	SOD	LUNAR	LODA	ABOD	**Mean AUC**
EIF	++	++	++	++	++	++	++	++	++	++	++	++	++	+	0.849
IF	++	++	++	++	++	++	++	++	++	++	++	++	++		0.837
gen2out	++	++	++	++	++	++	++	++	++	++	++	++	++		0.836
COPOD	++	++	++	++	++	++	++	++	++	++	++	++	++		0.831
ECOD	++	++	++	++	++	++	++	++	++	+	++				0.815
OCSVM	++	++	++	++	++	++	++	++	++		++				0.813
beta-VAE	++	++	++	++	++	++	++	++	++		++				0.813
AE	++	++	++	++	++	++	++	++	++		++				0.812
INNE	++	++	++	++	++	++	++	++	++		++				0.807
PCA	++	++	++	++	++	++	++	++	++						0.807
MCD	++	++	++	++	++	++	++	++	++		++				0.805
DynamicHBOS	++	++	++	++	++	++	++	++	++		++				0.804
u-CBLOF	++	++	++	++	++	++	++	++	++						0.793
HBOS	++	++	++	++	++	++	++	++	++		+				0.790
KDE	++	++	++	++	++	++	++	++	++		++				0.782
VAE	++	++	++	++	++	++	+	++							0.778
kth-NN	++	++	++	++	++	++	++	++	++		++				0.770
LMDD	++	++	++	++	++	++		++							0.757
kNN	++	++	++	++	++	++	++	++	+						0.755
ABOD	++	++	++	++	++	++		++							0.740
LODA	++	++		++											0.724
LUNAR		+													0.701
SOD															0.679
GMM															0.650
SO-GAAL															0.642
DeepSVDD														--	0.621
ensemble-LOF															0.615
ALAD														--	0.611
LOF														--	0.580
sb-DeepSVDD													--	--	0.566
ODIN														--	0.560
COF												-	--	--	0.530
CBLOF													--	--	0.487

Table 5: Significant differences between algorithms on the collection of global problems based on Nemenyi post-hoc analysis. ++/+ denotes that the row algorithm outperforms the column algorithm at $p = 0.05/p = 0.10$. Rows and columns are sorted by descending and ascending mean performance, respectively. Columns are not shown when the column algorithm is not outperformed by any other algorithms at $p = 0.05$ or $p = 0.10$. The last column shows the mean AUC.

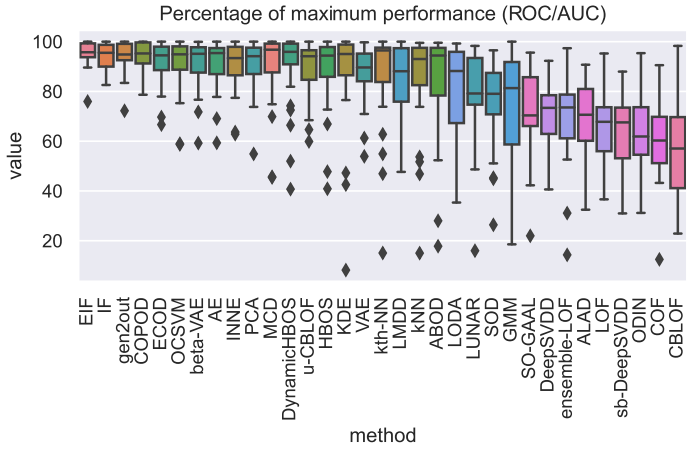


Figure 5: Boxplots of the performance of each algorithm on the global data sets in terms of percentage of maximum AUC. The maximum AUC is the highest AUC value obtained by the best performing algorithm on that particular data set. The whiskers in the boxplots extend 1.5 times the interquartile range past the low and high quartiles. data set-algorithm combinations outside of the whiskers are marked as diamonds.

55]. A somewhat comparable study by Fernández-Delgado et al. [16] on classification algorithms easily considered 121 data sets. The main reason for this discrepancy is that data sets for comparing anomaly detection algorithms rarely include categorical features, which are not an issue for comparing classification algorithms. It is certainly possible to further extend the collection of data sets, for example, through data set modifications. Campos et al. [10], Emmott et al. [14] and Emmott et al. [15], and Steinbuss and Böhm [52] modified data sets in different ways to create similar data sets with differing characteristics from a single base data set. While such modifications can be useful for targeted studies, near-duplicate data sets are far from independent and then seem detrimental to a proper statistical comparison of anomaly detection algorithms, such as can be observed in Emmott et al. [14].

In this study we compared 33 of the most commonly used algorithms for anomaly detection. This collection is certainly not exhaustive: many more methods exist [12, 14, 19, 44, 47], and likely even more will be invented. Also along this axis, there is a clear discrepancy with the study by Fernández-Delgado et al. [16] on classification algorithms, who incorporated 179 classifiers from 17 different families. Apparently, the number of classification algorithms largely exceeds the number of anomaly detection algorithms. But perhaps more

importantly, there are many more solid and easy-to-use implementations of classification algorithms in many different machine learning libraries than there are out-of-the-box implementations of anomaly detection algorithms. Before being able to perform the comparison in this study, we had to spend quite some effort to clean up and sometimes re-implement (parts of) existing code.

In this research we chose not to cover meta-techniques for ensembling. While ensembles are of great interest, a better understanding of the performance of base learners is an essential prerequisite before moving on to a study of ensemble methods.

While we evaluated neural networks in our comparison, no general guidelines exist on how to construct a well-performing network for any given data set, which is essential for the unsupervised setup considered in this study. The strength of many of these methods comes from high-level feature extraction implicitly performed by the network, which cannot be leveraged on the smaller tabular data sets used in this benchmark. Like Ruff et al. [44], we recognize that there is a major discrepancy between the availability of classification and anomaly detection benchmark data sets useful for deep learning approaches. More anomaly detection benchmark data sets useful for deep learning based anomaly detection would be a welcome addition to the field.

Cross-comparing the performance of algorithms on data sets, we noticed a clear separation between two clusters of data sets and roughly two clusters of corresponding algorithms. We characterized these clusters as "local" and "global" data sets and algorithms, in correspondence with common nomenclature in the literature [9, 19]. However, we are well aware that this characterization may turn out to be an oversimplification when analyzing more data sets and more algorithms in closer detail. For example, the local and global problems likely have quite some overlap, but need not be fully equivalent with multimodal and unimodal anomaly detection problems, respectively. Overlap between multimodal and local problems occurs when the different modes start having different densities, so that local algorithms that try to estimate these local densities fare better than global algorithms that cannot make this distinction. Further theoretical and empirical studies, for example, on carefully simulated data sets are needed to shed further light. We acknowledge that there is a gap in both theoretical and empirical studies on determining what types of anomalies are present in a data set, which would directly help in selecting an appropriate algorithm in conjunction with this research. Furthermore, we have not readily observed several well-described properties of anomalies. This exemplifies the need for more, varied, benchmark data sets.

2.6 CONCLUSION

Based on our research we can establish general guidelines on when users should apply which anomaly detection methods for their problem.

In general, when a user has no *a priori* knowledge on whether or not their data set contains local or global anomalies, EIF, Extended Isolation Forest, is the best choice. It outperforms 14 out of 33 other evaluated methods at $p = 0.05$, and is one the highest performing method based on its mean AUC score.

When a data set is known or suspected to contain local anomalies, which might for example occur when the data is known to contain multiple different density clusters, the best performing method is kNN, which outperforms 17 out of 33 methods at $p = 0.05$.

Datasets containing just global anomalies are best analyzed using EIF, which is the top performing algorithm on the data sets containing global anomalies. COPOD, gen2out, INNE and k-thNN all perform comparably, and these methods all outperform at least 10 other methods at $p = 0.05$. IF and EIF are the algorithms with the lowest computational complexity, which are usually preferable in practice.

Contemplating the above considerations, we are tempted to answer the question in the title of this chapter with "two": a toolbox with kNN, and EIF seems sufficient to perform well on the type of multivariate data sets considered in our study. These two algorithms are due to the scope of this study likely to perform well on unseen real-world multivariate data sets. This conclusion is open for further consideration when other algorithms and/or data sets are added to the bag, which should be relatively easy to check when extending the code and the data set pre-processing procedures that we open-sourced.

Future work following this study may seek to extend our comparative analysis with diverse types of data such as raw images, texts and time-series. All of these types of data require specific methods and tailored comparisons. Furthermore, automatically determining properties of anomalies in a data set before further analysis is an unexplored avenue of study which might provide users with even more detailed guidelines on which algorithm to apply.

ACKNOWLEDGMENTS AND DISCLOSURE OF FUNDING

The research reported in this chapter has been partly funded by the NWO grant NWA.1160.18.238 (`https://primavera-project.com/`), as well as BMK, BMDW, and the State of Upper Austria in the frame of the SCCH competence center INTEGRATE [(FFG grant no. 892418)] part of the FFG COMET Competence Centers for Excellent Technologies Programme.

REFERENCES

[1] D. Agarwal. "Detecting anomalies in cross-classified streams: a Bayesian approach." In: *Knowledge and Information Systems* 11.1 (2007), pp. 29–44.

[2] M. Amer and M. Goldstein. "Nearest-neighbor and clustering based anomaly detection algorithms for Rapidminer." In: *RapidMiner Community Meeting and Conference*. 2012, pp. 1–12.

[3] J. An and S. Cho. "Variational autoencoder based anomaly detection using reconstruction probability." In: *Special Lecture on IE* 2.1 (2015), pp. 1–18.

[4] A. Arning, R. Agrawal, and P. Raghavan. "A linear method for deviation detection in large databases." In: *Knowledge Discovery and Data Mining* 1141.50 (1996), pp. 972–981.

[5] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, Y. Zhu, and J. R. Wells. "Isolation-based anomaly detection using nearest-neighbor ensembles." In: *Computational Intelligence* 34.4 (2018), pp. 968–998.

[6] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola. "Fast optimal leaf ordering for hierarchical clustering." In: *Bioinformatics* 17.suppl_1 (2001), S22–S29.

[7] L. Birgé and Y. Rozenholc. "How many bins should be put in a regular histogram." In: *ESAIM: Probability and Statistics* 10 (2006), pp. 24–45.

[8] A. Brandsæter, E. Vanem, and I. K. Glad. "Efficient on-line anomaly detection for ship systems in operation." In: *Expert Systems with Applications* 121 (2019), pp. 418–437.

[9] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. "LOF: identifying density-based local outliers." In: *ACM SIGMOD International Conference on Management of Data*. 2000, pp. 93–104.

[10] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle. "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study." In: *Data Mining and Knowledge Discovery* 30.4 (2016), pp. 891–927.

[11] J. Demšar. "Statistical comparisons of classifiers over multiple data sets." In: *The Journal of Machine Learning Research* 7 (2006), pp. 1–30.

[12] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui. "A comparative evaluation of outlier detection algorithms: Experiments and analyses." In: *Pattern Recognition* 74 (2018), pp. 406–421.

[13] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[14] A. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong. "A meta-analysis of the anomaly detection problem." In: *arXiv preprint arXiv:1503.01158* (2015).

[15] A. F. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong. "Systematic construction of anomaly detection benchmarks from real data." In: *ACM SIGKDD workshop on outlier detection and description*. 2013, pp. 16–21.

[16] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. "Do we need hundreds of classifiers to solve real world classification problems?" In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3133–3181.

[17] R. Foorthuis. "On the nature and types of anomalies: A review of deviations in data." In: *International Journal of Data Science and Analytics* 12.4 (2021), pp. 297–331.

[18] M. Goldstein and A. Dengel. "Histogram-based Outlier Score (HBOS): A fast unsupervised anomaly detection algorithm." In: *KI-2012: Poster and Demo Track*. 2012, pp. 59–63.

[19] M. Goldstein and S. Uchida. "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data." In: *PloS one* 11.4 (2016), e0152173.

[20] A. Goodge, B. Hooi, S.-K. Ng, and W. S. Ng. "LUNAR: Unifying local outlier detection methods via graph neural networks." In: *AAAI Conference on Artificial Intelligence*. Vol. 36. 2022, pp. 6737–6745.

[21] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao. "ADBench: Anomaly Detection Benchmark." In: *Neural Information Processing Systems*. 2022.

[22] S. Hariri, M. C. Kind, and R. J. Brunner. "Extended Isolation Forest." In: *IEEE Transactions on Knowledge and Data Engineering* 33.4 (2019), pp. 1479–1489.

[23] V. Hautamaki, I. Karkkainen, and P. Franti. "Outlier detection using k-nearest neighbour graph." In: *International Conference on Pattern Recognition*. Vol. 3. IEEE. 2004, pp. 430–433.

[24] R. He and J. McAuley. "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering." In: *International Conference on World Wide Web*. 2016, pp. 507–517.

[25] Z. He, X. Xu, and S. Deng. "Discovering cluster-based local outliers." In: *Pattern Recognition Letters* 24.9-10 (2003), pp. 1641–1650.

[26] R. L. Iman and J. M. Davenport. "Approximations of the critical region of the fbietkan statistic." In: *Communications in Statistics - Theory and Methods* 9.6 (1980), pp. 571–595.

[27] N. Japkowicz, C. Myers, M. Gluck, et al. "A novelty detection approach to classification." In: *International Joint Conference on Artificial Intelligence*. Vol. 1. Citeseer. 1995, pp. 518–523.

[28] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. "Outlier detection in axis-parallel subspaces of high dimensional data." In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2009, pp. 831–838.

[29] H.-P. Kriegel, M. Schubert, and A. Zimek. "Angle-based outlier detection in high-dimensional data." In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2008, pp. 444–452.

[30] A. Krizhevsky and G. Hinton. *Learning multiple layers of features from tiny images*. Tech. rep. University of Toronto, 2009.

[31] L. J. Latecki, A. Lazarevic, and D. Pokrajac. "Outlier detection with kernel density functions." In: *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer. 2007, pp. 61–75.

[32] M.-C. Lee, S. Shekhar, C. Faloutsos, T. N. Hutson, and L. Iasemidis. "gen2Out: Detecting and ranking generalized anomalies." In: *IEEE International Conference on Big Data*. IEEE. 2021, pp. 801–811.

[33] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu. "COPOD: copula-based outlier detection." In: *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2020, 1118–1123.

[34] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. H. Chen. *ECOD: Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions*. 2022. eprint: 2201.00382.

[35] F. T. Liu, K. M. Ting, and Z.-H. Zhou. "Isolation forest." In: *IEEE International Conference on Data Mining*. IEEE. 2008, pp. 413–422.

[36] Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang, and X. He. "Generative adversarial active learning for unsupervised outlier detection." In: *IEEE Transactions on Knowledge and Data Engineering* 32.8 (2019), pp. 1517–1528.

[37] K. Malik, H. Sadawarti, and K. G S. "Comparative analysis of outlier detection techniques." In: *International Journal of Computer Applications* 97.8 (2014), pp. 12–21.

[38] P. B. Nemenyi. *Distribution-free multiple comparisons.* Princeton University, 1963.

[39] T. Pevnỳ. "Loda: Lightweight on-line detector of anomalies." In: *Machine Learning* 102.2 (2016), pp. 275–304.

[40] S. Ramaswamy, R. Rastogi, and K. Shim. "Efficient algorithms for mining outliers from large data sets." In: *International Conference on Management of Data*. 2000, pp. 427–438.

[41] S. Rayana. *ODDS Library*. 2016. URL: http://odds.cs.stonybrook.edu.

[42] P. J. Rousseeuw and C. Croux. "Alternatives to the median absolute deviation." In: *Journal of the American Statistical Association* 88.424 (1993), pp. 1273–1283.

[43] P. J. Rousseeuw and K. V. Driessen. "A fast algorithm for the minimum covariance determinant estimator." In: *Technometrics* 41.3 (1999), pp. 212–223.

[44] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller. "A unifying review of deep and shallow anomaly detection." In: *Institute of Electrical and Electronics Engineers* (2021).

[45] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft. "Deep semi-supervised anomaly detection." In: *arXiv preprint arXiv:1906.02694* (2019).

[46] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt, et al. "Support vector method for novelty detection." In: *Neural Information Processing Systems*. Vol. 12. 1999, pp. 582–588.

[47] E. Schubert and A. Zimek. "ELKI: A large open-source library for data analysis. ELKI Release 0.7.5 "Heidelberg"." In: *CoRR* abs/1902.03616 (2019). arXiv: 1902.03616. URL: https://arxiv.org/abs/1902.03616.

[48] S. Y. Shin and H.-j. Kim. "Extended autoencoder for novelty detection with reconstruction along projection pathway." In: *Applied Sciences* 10.13 (2020), p. 4497.

[49] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. *A novel anomaly detection scheme based on principal component classifier*. Tech. rep. University of Miami, department of Eletrical and Computer Engineering, 2003.

[50] E. H. Simpson. "The interpretation of interaction in contingency tables." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 13.2 (1951), pp. 238–241.

[51] J. Soenen, E. Van Wolputte, L. Perini, V. Vercruyssen, W. Meert, J. Davis, and H. Blockeel. "The effect of hyperparameter tuning on the Comparative evaluation of unsupervised anomaly detection methods." In: *Knowledge Discovery and Data Mining Workshop on Outlier Detection and Description*. 2021, pp. 1–9.

[52] G. Steinbuss and K. Böhm. "Benchmarking Unsupervised Outlier Detection with Realistic Synthetic Data." In: *ACM Transactions on Knowledge Discovery from Data* 15.4 (2021), pp. 1–20.

[53] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung. "Enhancing effectiveness of outlier detections for low density patterns." In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2002, pp. 535–548.

[54] A. Thomas, A. Gramfort, and S. Clémençon. "Learning Hyperparameters for Unsupervised Anomaly Detection." In: *Conférence sur L'apprentissage automatique-Cap 2016*. 2016.

[55] X. Xu, H. Liu, L. Li, and M. Yao. "A comparison of outlier detection techniques for high-dimensional data." In: *International Journal of Computational Intelligence Systems* 11.1 (2018), pp. 652–662.

[56] H. Zenati, M. Romain, C.-S. Foo, B.
Lecouat, and V. Chandrasekhar. "Adversarially learned anomaly detection." In: *International Conference on Data Mining*. IEEE. 2018, pp. 727–736.

[57] Y. Zhao, Z. Nasrullah, and Z. Li. *PyOD: A Python toolbox for scalable outlier detection*. 2019. eprint: `1901.01588`.

[58] L. Zhou, W. Deng, and X. Wu. "Unsupervised anomaly localization using VAE and beta-VAE." In: *arXiv preprint arXiv:2005.10686* (2020).

[59] A. Zimek, R. J. Campello, and J. Sander. "Ensembles for unsupervised outlier detection: challenges and research questions a position paper." In: *ACM SIGKDD Explorations Newsletter* 15.1 (2014), pp. 11–22.

3

AUTOENCODERS FOR ANOMALY DETECTION ARE UNRELIABLE

Autoencoders are frequently used for anomaly detection, both in the unsupervised and semi-supervised settings. They rely on the assumption that when trained using the reconstruction loss, they will be able to reconstruct normal data more accurately than anomalous data. Some recent works have posited that this assumption may not always hold, but little has been done to study the validity of the assumption in theory. In this work we show that this assumption indeed does not hold, and illustrate that anomalies, lying far away from normal data, can be perfectly reconstructed in practice. We revisit the theory of failure of linear autoencoders for anomaly detection by showing how they can perfectly reconstruct out of bounds, or extrapolate undesirably, and note how this can be dangerous in safety critical applications. We connect this to non-linear autoencoders through experiments on both tabular data and real-world image data, the two primary application areas of autoencoders for anomaly detection.

3.1 INTRODUCTION

Autoencoders are one of the most popular architectures within anomaly detection, either directly, or as a scaffold or integral part in larger pipelines or architectures. They are commonly used across a variety of domains, such as predictive maintenance [20], network anomaly detection [28], and intrusion detection [15], but find much contemporary use in computer vision anomaly detection, with applications such as industrial inspection [32], medical imaging [33], structural health monitoring [12] and video surveillance [13, 36]. Many of these applications are safety critical, meaning that the reliability of these algorithms is of utmost importance in order to prevent catastrophic failure and associated dangers and consequences.

Anomaly detection using autoencoders typically relies on using the reconstruction loss, often the mean squared error (MSE), as a proxy for "anomalousness". The underlying assumption is that anomalies are harder to reconstruct, and will therefore have a higher reconstruction loss. However, the validity of this assumption has been questioned in recent years. Merrill and Eskandarian [25] and Beggel, Pfeiffer, and Bischl [5] for example state that anomalies in the training data might lead to reconstruction of anomalies, leading to unreliable detectors. Some researchers have noted that reconstruction of unseen anomalies might also occur in the semi-supervised setting, where the training data is assumed to contain no anomalies [1, 2, 17, 38]. Yet,

little work has been done on the nature of failure and reliability of autoencoders.

In this work we provide valuable insights into the reliability of autoencoders for anomaly detection. Following the seminal works of Bourlard and Kamp [9], and Baldi and Hornik [3] we mathematically and experimentally study autoencoder failure modes, whilst briefly examining how various activation functions influence these failures. We show that failure is not just a rarely occurring edge case, but also show failure cases on tabular data and on real-world image data commonly used in anomaly detection benchmarks. By doing this we provide a foundation for future research in solving the demonstrated unreliability of autoencoders for anomaly detection and furthering our understanding of autoencoders. We show that for different architectures and activations functions, even with sufficiently low-dimensional latent spaces, these problems persist. By developing our understanding of autoencoders for anomaly detection, we hope to function as a warning regarding the reliability of autoencoders in practice, and as a scaffold for future developments striving for more robust anomaly detection algorithms. To ensure reproducibility of all experiments we use only open-source data and provide code for all experiments.

3.2 RELATED WORK

This work is not the first to recognize that autoencoders have several issues as anomaly detectors. The most discussed type of failure is the unwanted reconstruction of anomalies, which is also the focus of this work. Several causes of this unwanted behavior have been proposed.

Many works focus on the unsupervised setting, and observe that contrary to prior belief, autoencoders will fairly easily reconstruct any anomalies present in the training data, leading to an unusable detector [5, 11, 25, 31].

Several works cover the anomaly reconstruction problem within the semi-supervised setting. Most commonly, it is only experimentally observed that anomalies are well reconstructed [1, 2, 11, 17, 23, 26, 29, 38]. Based on these experimental results, some solutions have been proposed. Gong et al. [17] mention that out-of-bounds reconstruction can occur and propose adding a memory module to the autoencoder to alleviate the issue. While the addition of the memory module can aid in limiting out-of-bounds reconstruction, it also leads to a severely decreased reconstruction ability and substantial added complexity in training and optimizing the network. Zong et al. [38] note that while some anomalies have a high reconstruction loss, some occupy the region of normal reconstruction loss, and add a Gaussian mixture model to the latent space to aid in detection of anomalies under the assumption that anomalies occupy a low-density region in

the latent space. Similarly, Cheng et al. [11] aim to detect anomalies in the latent space by looking at the distance to the centroid. From our experiments we can glean that relying on distances or densities in the latent space does not always work in practice. Astrid et al. [1, 2] make use of constructed pseudo anomalies in training the autoencoder. They add adaptive noise to normal samples to generate pseudo anomalies. In the reconstruction, they then optimize the reconstruction loss between the pseudo anomaly and the normal sample used to generate it. While they show promising results and greater discriminative power on benchmark datasets, they do not quantify to which degree their performance gains can be attributed to the reduction of the out-of-bounds reconstruction. Salehi et al. [29] aim to limit the reconstruction of anomalies by generating adversarial samples. The adversarial examples are generated by perturbing the normal samples, and minimizing the effect those perturbations have in the latent space. This is similar to the concept of denoising autoencoders. Based on our experiments, we do not think this results in a reliable autoencoder, as often adversarial anomalies can occupy the latent space close to normal data.

Some authors have moved beyond the experimental, and propose explanations for the anomaly reconstruction problem. For example You et al. [35], Lu et al. [22] and Bercea, Rueckert, and Schnabel [6] propose that anomaly reconstruction can happen because an autoencoder can learn an “identical shortcut", where both normal data and anomalous data is effectively reconstructed using an identity mapping. This point has however been countered by Cai, Chen, and Cheng [10] who show that by constraining the latent space to be sufficiently low dimensional, this problem can be avoided.

The second line of thought follows from VAE anomaly detection, where Havtorn et al. [18] theorize that in out-of-distribution detection, unwanted reconstruction can happen due to a high correlation between learned low-level features for in- and out-of-distribution data.

A third line of thought is proposed by Zhou [37] who propose that reconstruction of out-of-distribution samples can happen due to out-of-distribution data having smaller neural activations than in-distribution data.

Finally, some works theorize that autoencoders can perfectly reconstruct data due to the anomalies occupying the reconstruction hyperplane, or latent space manifold. Denouden et al. [14] for example note this phenomenon, and aim to solve it by adding the Mahalanobis distance in the latent space to the loss. The most detailed work is that of Yoon, Noh, and Park [34] who provide an example of the hyperplane interpolating between clusters of data. They solve this by introducing a normalized autoencoder which reinterprets the autoencoder as a likelihood-based energy model. We specifically follow up on this line of reasoning, and revisit mathematical proofs and provide new exper-

imental evidence of anomaly reconstruction due to both unwanted extrapolation and inter-class interpolation.

3.3 BACKGROUND

3.3.1 *Anomaly Detection*

In practical anomaly detection, we attribute a score $s_i = f_{\text{anomaly score}}(\mathbf{x}_i)$ for each sample $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^n$, i.e. the i-th row of dataset $\mathbf{X}$ with size m-by-n. The score should then be higher for anomalous samples than for normal data.

When applied to some dataset consisting of both normal data and anomalies, i.e. $\mathbf{X} = \{\mathbf{X}^{\text{normal}}, \mathbf{X}^{\text{anomalous}}\}$, a perfect anomaly detector will assign higher scores to anomalies than to normal data: $\min_i(f_{\text{anomaly score}}(\mathbf{x}_i^{\text{anomalous}})) > \max_i(f_{\text{anomaly score}}(\mathbf{x}_i^{\text{normal}}))$.

The two most common anomaly detection settings are unsupervised and semi-supervised. Unsupervised anomaly detection is characterized by having no discernible "train" and "test" splits. Instead, we only consider a single dataset $\mathbf{X} = \{\mathbf{X}^{\text{normal}}, \mathbf{X}^{\text{anomalous}}\}$, where we are uncertain which samples are anomalous and which are not. In the semi-supervised setting we instead have a conventional "train" and "test" set. The train set consists out of only normal samples: $\mathbf{X}^{\text{train}} = \{\mathbf{X}^{\text{train, normal}}\}$, while the test set is unknown, and can contain both normal and anomalous samples: $\mathbf{X}^{\text{test}} = \{\mathbf{X}^{\text{test, normal}}, \mathbf{X}^{\text{test, anomalous}}\}$. In this paper we will only consider the semi-supervised case, and simplify the notation with $\mathbf{X} = \mathbf{X}^{\text{train}}$ and $\mathbf{x}_i$ referring to an individual training data point, which in the semi-supervised case by definition is not an anomaly. We then consider a new data point $\mathbf{a}$ to determine whether this is an anomaly or not. In older literature, semi-supervised anomaly detection is often called one-class classification.

3.4 OUT-OF-BOUNDS RECONSTRUCTION

In this section we will show that autoencoders can yield zero-loss reconstruction far away from all training data, and that these autoencoders will then fail to detect certain anomalies. Our analysis follows the results of Bourlard and Kamp [9], moving from PCA to linear autoencoders to non-linear autoencoders.

Out-of-bounds reconstruction is unwanted within the application of anomaly detection, as it leads to a low reconstruction loss for data that can be considered anomalous, thereby leading to false negatives. These regions of out-of-bounds reconstruction can also be exploited by targeted adversarial evasion attacks.

In the worst case, unwanted perfect reconstruction causes an anomaly $\mathbf{a} \in \mathbb{R}^n$ far from all training data to be ranked as being less

anomalous than or equally anomalous as all training data, that is: $f_{\text{anomaly score}}(\mathbf{a}) \leqslant \min_i(f_{\text{anomaly score}}(\mathbf{x}_i))$.

3.4.1 *Anomaly Detection Using the Reconstruction Loss*

Both PCA and autoencoders are dimensionality reduction techniques that can be used to detect anomalies using their reconstruction loss, commonly known as the mean squared error, or MSE. We can calculate the reconstruction loss by comparing a sample $\mathbf{x}_i$ to its reconstruction $\hat{\mathbf{x}}_i$: $\mathcal{L}_R(\mathbf{x}_i, \hat{\mathbf{x}}_i) = \frac{1}{n}\sum_{j=1}^{n}\left(x_{i,j} - \hat{x}_{i,j}\right)^2$, for each sample vector $\mathbf{x}_i$. This reconstruction loss often serves as a proxy for detecting anomalies, with the underlying assumption that a higher reconstruction loss indicates a higher likelihood of the sample being an anomaly.

For both PCA and autoencoders we want to find a lower-dimensional encoding $\mathbf{Y}$, e.g. $d < n$, in the encoding space $\mathcal{Y} = \mathbb{R}^d$ by applying the function $g : \mathcal{X} \to \mathcal{Y}$. We then decode $\mathbf{Y}$ by transforming it back into the space $\mathcal{X}$ through the decoder $h : \mathcal{Y} \to \mathcal{X}$, yielding the reconstructed data $\hat{\mathbf{X}}$. Summarizing, we learn the concrete transformations $\mathbf{X} \xrightarrow{g} \mathbf{Y} \xrightarrow{h} \hat{\mathbf{X}}$.

We can then formulate the anomaly scoring function in terms of the reconstruction loss, encoder, and decoder:
$f_{\text{anomaly score}}(\mathbf{x}_i) = \mathcal{L}_R(\mathbf{x}_i, h(g(\mathbf{x}_i)))$.

The worst case can then be formulated as: there exists an $\mathbf{a}$ far from all training data such that $\mathcal{L}_R(\mathbf{a}, \hat{\mathbf{a}}) \leqslant \min_i(\mathcal{L}_R(\mathbf{x}_i, \hat{\mathbf{x}}_i))$.

3.4.2 *PCA*

In PCA, we factorize $\mathbf{X}$ as $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\intercal$ using singular value decomposition (SVD), where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices containing the left- and right-singular vectors of $\mathbf{X}$, respectively, and $\mathbf{\Sigma}$ is a diagonal scaling matrix. The encoding, or latent, space is then obtained by projecting onto the first d right-singular vectors: $\mathbf{Y} = g(\mathbf{X}) = \mathbf{X}\mathbf{V}_d$, where $\mathbf{V}_d$ contains the first d columns of $\mathbf{V}$. The transformation back into $\mathcal{X}$ is given by $\hat{\mathbf{X}} = h(\mathbf{Y}) = \mathbf{Y}\mathbf{V}_d^\intercal$.

Revisiting Bourlard and Kamp [9], we will show that there exist some $\mathbf{a} \in \mathcal{R}^n$ for which the reconstruction loss is zero, but that are far away from the normal data, i.e. $\min_i(\text{dist}(\mathbf{x}_i, \mathbf{a})) > \delta$, for any arbitrary choice of δ. Hereby we prove that it is possible to find anomalous, adversarial, examples with perfect out-of-bounds reconstruction. We can prove this even in the semi-supervised setting, where we guarantee that the model was not exposed to anomalous data at training time. Due to the semi-supervised setting being more restrictive, the proofs also apply to the unsupervised case.

Let us now look for some $\mathbf{a} \in \mathbb{R}^n$. We aim to prove that if $\mathbf{a}$ lies in the column space of $\mathbf{V}_d$, then the reconstruction loss $\mathcal{L}_R(\mathbf{a}, h(g(\mathbf{a}))) = 0$.

We now need to show that there exists some $\mathbf{a}$ such that $h(g(\mathbf{a})) = \mathbf{a}$. For PCA, this condition can be written as:

$$\mathbf{a}\mathbf{V}_d\mathbf{V}_d^T = \mathbf{a}.$$

Assume $\mathbf{a}$ is in the row space of $\mathbf{V}_d^T$. Then $\mathbf{a}$ can be expressed as a linear combination of the rows in $\mathbf{V}_d^T$. Let $\mathbf{c} \in \mathbb{R}^d$ be such that:

$$\mathbf{a} = \mathbf{c}\mathbf{V}_d^T.$$

Substitute $\mathbf{a}$ into the left-hand side of the reconstruction equation:

$$\mathbf{a}\mathbf{V}_d\mathbf{V}_d^T = \mathbf{c}\mathbf{V}_d^T\mathbf{V}_d\mathbf{V}_d^T.$$

Since $\mathbf{V}_d$ is composed of orthonormal columns, $\mathbf{V}_d^T\mathbf{V}_d = \mathbf{I}_d$, where $\mathbf{I}_d$ is the d-by-d identity matrix. Therefore:

$$\mathbf{c}\mathbf{V}_d^T\mathbf{V}_d\mathbf{V}_d^T = \mathbf{c}\mathbf{V}_d^T = \mathbf{a}.$$

Thus, $\mathbf{a}$ satisfies the condition $h(g(\mathbf{a})) = \mathbf{a}$, implying that the reconstruction loss $\mathcal{L}_R(\mathbf{a}, g(h(\mathbf{a}))) = 0$.

Now we will prove that there exists some adversarial example $\mathbf{a} \in \mathbb{R}^n$ that is far from all normal data, i.e. $\min_i(\text{dist}(\mathbf{x}_i, \mathbf{a})) > \delta$, for an arbitrary δ and the Euclidean distance metric, but still has a reconstruction loss $\mathcal{L}_R(\mathbf{a}, g(h(\mathbf{a}))) = 0$.

Let us first recall that any $\mathbf{a}$ in the column space of $\mathbf{V}_d$ will have zero reconstruction loss.

If we then define $\mathbf{a} = \mathbf{x}_i\mathbf{V}_d\mathbf{V}_d^T + \mathbf{c}\mathbf{V}_d^T$, $\mathbf{a}$ will still have zero reconstruction loss.

Then for the Euclidean distance it follows that:

$$\text{dist}(\mathbf{x}_i, \mathbf{a})^2 = \|\mathbf{x}_i - \mathbf{x}_i\mathbf{V}_d\mathbf{V}_d^T\|^2 + \|\mathbf{a} - \mathbf{x}_i\mathbf{V}_d\mathbf{V}_d^T\|^2,$$

or the squared Euclidean distance is equal to the distance from $\mathbf{x}_i$ to its projection onto the hyperplane $\mathbf{x}_i\mathbf{V}_d\mathbf{V}_d^T$ plus the distance from that projection $\mathbf{x}_i\mathbf{V}_d\mathbf{V}_d^T$ to $\mathbf{a}$.

It then follows that:

$$\begin{aligned}&\text{dist}(\mathbf{x}_i, \mathbf{a})^2 \geqslant \|\mathbf{a} - \mathbf{x}_i\mathbf{V}_d\mathbf{V}_d^T\|^2 \\ &= \|\mathbf{x}_i\mathbf{V}_d\mathbf{V}_d^T + \mathbf{c}\mathbf{V}_d^T - \mathbf{x}_i\mathbf{V}_d\mathbf{V}_d^T\|^2 = \|\mathbf{c}\mathbf{V}_d^T\|^2,\end{aligned}$$

which we can increase to arbitrary length. This can be intuited as moving the projection of $\mathbf{x}_i$ along the hyperplane.

To ensure that we increase the distance to all points $\mathbf{x}_i$ rather than just a single one, we need to move outward starting from a sample on the convex hull enclosing $\mathbf{X}\mathbf{V}_d$. Any point in this convex set, that is the set of points occupying the convex hull, can be moved along the hyperplane to increase the distance to all points $\mathbf{x}_i\mathbf{V}_d$, and therefore

to all points x_i as long as c lies in the direction from $x_i V_d$ to the exterior of the convex hull.

We can thus always find a $a = x_i V_d V_d^T + c V_d^T$, for some $x_i V_d$ in the convex set of XV_d and choose c so that it points from $x_i V_d$ to the exterior of the convex hull and is of sufficient length such that $\min_i(\text{dist}(x_i, a)) > \delta$, for an arbitrary δ.

Hence, all vectors $a \in \mathbb{R}^n$ found in this way are constructed anomalies, or adversarial examples, that are far from all normal data, but still have zero reconstruction loss.

We posit that the same principle applies to other distance metrics, and the intuition is that this follows the same line of reasoning as presented here for the Euclidean distance.

3.4.3 *Linear Autoencoders*

Linear neural networks, like PCA, can also exhibit out-of-bounds reconstruction for certain anomalous data points.

Linear autoencoders consist of a single linear encoding layer and a single linear decoding layer. Given a mean-centered dataset X, the encoding and decoding transformations can be represented as follows:

$$Y = g(X) = XW_{enc},$$

$$\hat{X} = h(Y) = YW_{dec}^T = XW_{enc}W_{dec}^T$$

where W_{enc} is the n-by-d weight matrix of the encoder, and W_{dec}^T is the d-by-n weight matrix of the decoder. We assume the autoencoder to have converged to the global optimum. Note that we define W_{dec}^T in its transposed form to illustrate its relation to V_d^T. Due to the full linearity of the model, even multiple layer networks can be simplified to a single multiplication. It is known that a well-converged linear autoencoder finds an encoding in the space spanned by the first d principal components [3]. In other words, the encoding weight matrix can be expressed in terms of the first d principal components and some invertible matrix C:

$$W_{enc} = V_d C.$$

At the global optimum W_{dec}^T can be expressed as the inverse of W_{enc} :

$$W_{dec}^T = W_{enc}^{-1} = (V_d C)^{-1} = C^{-1} V_d^{-1} = C^{-1} V_d^T.$$

To show that linear autoencoders can exhibit perfect out-of-bounds reconstruction, we follow the same lines as for PCA.

Let us now look for some $a \in \mathbb{R}^n$. We aim to prove that if a lies in the column space of V_d, then the reconstruction loss $\mathcal{L}_R(a, h(g(a))) = 0$.

We need to show that there exists some $\mathbf{a}$ such that $h(g(\mathbf{a})) = \mathbf{a}$. For linear autoencoders, this condition can be written as:

$$\mathbf{a}\boldsymbol{W}_{\text{enc}}\boldsymbol{W}_{\text{dec}}^{\mathsf{T}} = \mathbf{a}.$$

Assume $\mathbf{a}$ is in the row space of $\boldsymbol{V}_d^{\mathsf{T}}$. Then $\mathbf{a}$ can be expressed as a linear combination of the rows in $\boldsymbol{V}_d$. Let $\mathbf{c} \in \mathbb{R}^d$ be such that:

$$\mathbf{a} = \mathbf{c}\boldsymbol{V}_d^{\mathsf{T}}.$$

Then it follows that:

$$\begin{aligned}\mathbf{a}\boldsymbol{W}_{\text{enc}}\boldsymbol{W}_{\text{dec}}^{\mathsf{T}} &= \mathbf{c}\boldsymbol{V}_d^{\mathsf{T}}\boldsymbol{W}_{\text{enc}}\boldsymbol{W}_{\text{dec}}^{\mathsf{T}} \\ &= \mathbf{c}\boldsymbol{V}_d^{\mathsf{T}}\boldsymbol{V}_d\boldsymbol{C}\boldsymbol{C}^{-1}\boldsymbol{V}_d^{\mathsf{T}} = \mathbf{c}\boldsymbol{V}_d^{\mathsf{T}}\boldsymbol{V}_d\boldsymbol{V}_d^{\mathsf{T}} = \mathbf{c}\boldsymbol{V}_d^{\mathsf{T}} = \mathbf{a},\end{aligned}$$

indicating that $\mathbf{a}$ satisfies the condition $h(g(\mathbf{a})) = \mathbf{a}$, implying that the reconstruction loss $\mathcal{L}_R(\mathbf{a}, g(h(\mathbf{a}))) = 0$.

By proving this, the case of the linear autoencoder reduces to that of PCA, with the same proof that adversarial examples satisfying $\min_i(\text{dist}(\mathbf{x}_i, \mathbf{a})) > \delta$, with a reconstruction loss $\mathcal{L}_R(\mathbf{a}, g(h(\mathbf{a}))) = 0$, exist.

An extension of this proof to the case of linear networks with bias terms applied on non-centered data can be found in Appendix 1.

3.4.4 *Non-Linear Autoencoders*

In this section we show that datasets exist for which we can prove that non-linear neural networks perform the same unwanted out-of-bounds reconstruction. Then we experimentally show that this behavior indeed occurs in more complex real-world examples. In this way, we illustrate how autoencoders demonstrate unreliability even in real-world scenarios.

3.4.4.1 *Failure of a Non-Linear Network with ReLU Activations*

We can show on a simple dataset that unwanted reconstruction behavior can occur in non-linear autoencoders. We consider a two-dimensional dataset $\boldsymbol{X}$ consisting out of normal samples $\mathbf{x}_i = \alpha_i(1, 1)$, where α_i is some scalar. Simply put, every normal sample $\mathbf{x}_i$ occupies the diagonal. This dataset can be perfectly reconstructed by a linear autoencoder with $\boldsymbol{W}_{\text{enc}} = \beta(1, 1)^{\mathsf{T}}$, where β is some scalar. The simplest non-linear autoencoder with ReLU activation will find the same weights, but with a bias such that $\mathbf{x}_i\boldsymbol{W}_{\text{enc}} + \mathbf{b}_{\text{enc}} > 0$ for all $\mathbf{x}_i \in \boldsymbol{X}$, i.e. $\mathbf{b}_{\text{enc}} \geqslant \min_i(\mathbf{x}_i\boldsymbol{W}_{\text{enc}})$. This will then lead to a perfect reconstruction for all $\mathbf{x}_i$. Adversarial anomalies $\mathbf{a}$ can also be easily found as $\mathbf{a} = c(1, 1)$, where $c \gg \frac{\max_i(\mathbf{x}_i)}{(1,1)}$ is some sufficiently large scalar such that $\min_i(\text{dist}(\mathbf{x}_i, \mathbf{a})) > \delta$ is satisfied. We theorize that even beyond

this simple case, similar linear behavior can occur beyond the convex hull that the normal data occupies. We experimentally show this anomaly reconstruction behavior in later sections.

3.4.4.2 *Tabular Data*

On more complex datasets, we observe similar behavior. We have synthesized several two-dimensional datasets to show how non-linear autoencoders behave when used for anomaly detection. These datasets, as well as contours of the MSE of autoencoders trained on these datasets, are visualized in Figure 6. In each of these cases, we have synthesized a dataset by sampling 100 points per distribution, either from a single or from two Gaussians as in 6a, 6b, 6e, and 6f, or from $x_2 = x_1^2$ with added Gaussian noise in 6c and 6d. In all cases we use an autoencoder with layer sizes [2,5,1,5,2], except for 6d, where we use layer sizes of [2,100,20,1,20,100,2] to better model the non-linearity of the data. All layers have ReLU (Subfigures 6a, 6b, 6c, and 6d), or sigmoid (Subfigures 6e and 6f) activations, except for the last layer, which has a linear activation. In these figures the color red is used to highlight those areas where the autoencoder is able to nearly perfectly reconstruct the data, i.e. MSE $< \epsilon = 0.1$.

RELU ACTIVATION FAILURE ON TABULAR DATA

We can readily observe some of the problematic behavior of autoencoders as anomaly detectors. Firstly, in Figure 6a we observe that well outside the bounds of the training data there is an area with a near-perfect reconstruction. Worryingly, the reconstruction loss is lower than for a large part of the area which the normal data occupies. If we move in the $(-1, -1)$ direction, the encoder and decoder will no longer match perfectly. Even so, problematically low reconstruction losses can be found in this direction. In Figures 6c and 6d we see the same linear out-of-bounds behavior. In each of these cases, the mismatch between encoder and decoder in the linear domain is less noticeable, leading to even larger areas of near-perfect reconstruction. Lastly, in Figure 6b that there is an area between the two clusters with a good reconstruction. Likely the units responsible for this area are still close to their initialization, and due to the simple shallow network architecture can not meaningfully contribute to the reconstruction of any samples.

Our intuition of this behavior directly relates to the proof of out-of-bounds reconstruction we have provided for linear autoencoders. At the edge of the data space, only a few of the ReLU neurons activate. Beyond this edge, no new neurons will activate, nor will any of the activated ones deactivate. This can lead to linear behavior on some edge of the data space, i.e., in this area the network reduces to a linear transformation $\boldsymbol{W}_{\text{enc}}$. If we now synthesize some $\mathbf{a}$ such that it lies in the column space of $\boldsymbol{W}_{\text{enc}}$, we can again find some adversarial

anomalies $\mathbf{a} = c\boldsymbol{W}_{\text{enc}}^{\mathsf{T}}$. Like we have observed in Figure 6a, there may be a mismatch between the encoder and decoder, even at the global optimum, so we might not be able to increase c towards infinity and still find adversarial examples with $\mathcal{L}_R(\mathbf{a}, g(h(\mathbf{a}))) < \epsilon$.

SIGMOID ACTIVATION AUTOENCODERS

Nowadays, full sigmoid networks have mostly fallen out of favor in deeper networks due to their vanishing gradients [16, 19]. However, sigmoids are more attractive to use in anomaly detection because they lead to networks that do not exhibit the hyperplane issues that the ReLU suffers from. While sigmoids have the more desirable property of tapering off at high or low input, making it hard to perfectly reconstruct data far away from normal data, autoencoders with just sigmoid activation can still behave unexpectedly, albeit less so than those with ReLU activation.

We can see in Figure 6e that the data is nicely approximated by a sigmoid autoencoder. It extends nicely to the first and last samples on the direction of the first principal component, and does not extend beyond that. When we extend this example to multimodal data, as in Figure 6f, we can see different undesirable behavior arising. There exists an area where the sigmoids reconstructing both clusters intersect. Due to the two distinct sigmoids mixing, we can find a hyperplane orthogonal to the first principal component where the reconstruction loss is much lower than would be expected. While in this case there are no points on the hyperplane which would have a lower reconstruction loss than all normal data, there is still a substantial area where the reconstruction loss is lower than for many of the normal data points.

OTHER ACTIVATION FUNCTIONS

While we have explicitly discussed the ReLU and sigmoid activation functions, the behavior shown is similar for other activation functions. Effectively, we can categorize most activation functions as those having an order of continuity of C^0 like the ReLU, or C^∞ like the sigmoid. In summary, activation functions with an order of continuity of C^0 suffer most from out-of-bounds reconstruction, but allow for more easily trainable deep networks. In contrast, activation functions with an order of continuity of C^∞ generally have more desirable properties for anomaly detection, but are harder to use in deep networks due to the vanishing gradient.

3.4.5 *Convolutional Autoencoders*

All the previous examples clearly illustrate autoencoders' possible failure and unreliability when used for anomaly detection on tabular data. Yet, many applications of anomaly detection are in computer

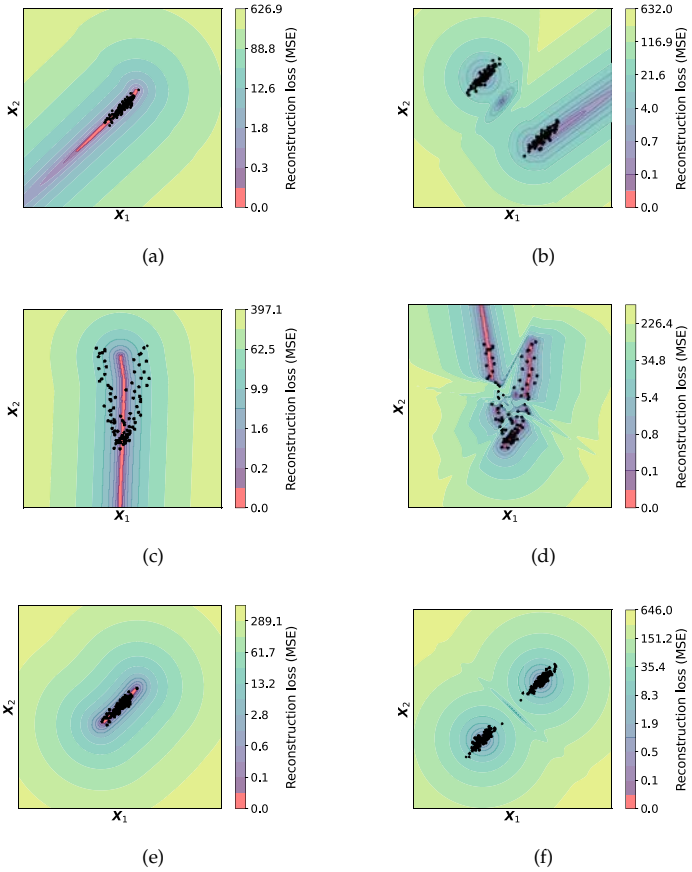


Figure 6: Plots of the contours of the reconstruction loss of non-linear autoencoders when applied to 3 distinct datasets. The datasets consist of 100 samples from a 2D Gaussian (a, e), 100 samples for each of 2 different 2D Gaussians (b, f), and 100 samples from a classic banana-shaped function with Gaussian noise (c, d). In (a, b, c, e, f) a [2,5,1,5,2] autoencoder is used, while in (d) a deeper [2,100,20,1,20,100,2] autoencoder is used. The contour plot is colored red whenever the MSE is below a set threshold $\epsilon < 0.1$ to indicate a near-perfect reconstruction. Note that the color scaling is exponential to better visualize the MSE loss.

vision, where anomaly detection can be used to detect foreign objects. Typical examples of computer vision anomaly detection are surveillance, where videos are analyzed to find possible security threats [27, 30], structural health monitoring [4], and industrial inspection [7].

For most applications of autoencoders on image data, the architecture is fairly straightforward. ReLU activation functions are most commonly used throughout the network, with a sigmoid activation at the final layer. Connections to and from the bottleneck layer are often chosen to be just linear, to allow for a richer internal representation. As most layers have ReLU activation functions, these networks do not suffer from the vanishing gradient. Yet, due to using a sigmoid at the last layer, these networks suffer less from the issues encountered in full ReLU/linear networks as discussed in Section 3.4.4.2. Nonetheless, we will show that even on more complex real-world problems, autoencoders remain unreliable and are often able to reconstruct out-of-bounds.

3.4.5.1 *Failure on real-world data: MNIST*

To show that deeper non-linear networks trained on real-world image data can still undesirably reconstruct anomalies we will study an autoencoder for anomaly detection that was trained on the well-known MNIST dataset [21]. Benchmarking computer vision anomaly detection algorithms is not as standardized as classification benchmarking, as datasets with “true” anomalies are exceedingly rare. The common method for benchmarking these algorithms is to take a classification dataset and select a subset of classes as “normal” data and another distinct subset as “anomalies”. This is analogous to other, more-developed, fields such as tabular anomaly detection [8]. There is no general consensus on which digits are taken as the normal data, and how many. In our experiments, both shown and non-shown, we have tried several different combinations and observe that in some cases out-of-bounds reconstruction occurs.

The 2D convolutional autoencoder we will discuss has a 2-layer encoder and 2-layer decoder. Down- and upsampling to and from the latent space is done using a fully connected linear layer. The convolutional layers all use ReLU activations, except for the last one, which is a sigmoid to bound the data to the original 0-1 range. In these experiments, the latent space is set to be two dimensional, far below the maximum to avoid the "identical shortcut" as noted by Cai, Chen, and Cheng [10]. This serves as proof that the "identical shortcut" is not the cause of anomaly reconstruction. In Figures 7a and 7b we show how the reconstruction loss behaves in the latent space when we apply this autoencoder on a train set consisting out of a subset of digits. These contourplots are constructed by sampling each point in the latent space, decoding it to get an artificial sample, and then calculating the reconstruction loss between the artificial sample and

its reconstruction loss. We subsequently show the latent representations of all normal data in the same space. We should note that as the encoder is a many-to-one mapping, the reconstruction loss in the grid does not necessarily correspond to the reconstruction loss of a real sample occupying the same point in that grid.

Looking at Figure 7a we see that a 2D latent space is able to separate the digits 4 and 5, with 7 occupying the middle between the two classes. As expected, the reconstruction loss grows the larger the distribution shift becomes. However, the reconstruction loss landscape is fairly skewed, with the MSE starkly increasing towards the right, and slowly towards any other direction, indicating model bias. Most notably, around $(-4.2, -5.2)$ we observe an out-of-bounds area of low reconstruction loss. Due to this type of visualization, we can easily generate an adversarial anomaly by simply decoding the latent space sample: $\mathbf{a} = h((-4.2, -5.2))$. This leads to the adversarial anomaly shown in Figure 7c. The adversarial anomaly shares some features with the digits used for training, but does not resemble any of them specifically, making it a clear false negative. Indeed, this sample fulfills our earlier criterion of $\mathcal{L}_R(\mathbf{a}_i, \hat{\mathbf{a}}_i) \leqslant \min_i(\mathcal{L}_R(\mathbf{x}_i, \hat{\mathbf{x}}_i))$, as for this example $\mathcal{L}_R(\mathbf{a}_i, \hat{\mathbf{a}}_i) = 0.014$, and $\min_i(\mathcal{L}_R(\mathbf{x}_i, \hat{\mathbf{x}}_i)) = 8.47$.

We also looked at a simpler example, where we train on the digits 0 and 1 to get a clearer separation of the two classes. In our previous experiments with sigmoid activation functions in Section 3.4.4.2 we observed that at the intersection of the two modalities some unwanted interpolation can occur. In Figure 7b we can observe the same thing, where at the intersection of the two classes we have a very small area in the latent space with a very low reconstruction loss. The normal data close to this area is however not well reconstructed. By generating an artificial sample from the lowest MSE in this latent space, at $(0.535, -0.353)$, we can find an adversarial anomaly $\mathbf{a} = h((0.535, -0.353))$ with $\mathcal{L}_R(\mathbf{a}_i, \hat{\mathbf{a}}_i) = 0.022$, substantially lower than $\min_i(\mathcal{L}_R(\mathbf{x}_i, \hat{\mathbf{x}}_i)) = 1.61$. This adversarial anomaly is visualized in Figure 7d. We find, unsurprisingly, that the adversarial anomaly here is a mix of the features of the 0 and 1 class.

Similar to our experiments on the digits 4, 5, and 7 autoencoder, we identified an area at the edge of the 1 class where the reconstruction loss is low, but where few normal data points can be found. In contrast to our previous experiment, this area corresponds to a more uncommon diagonally drawn 1, as shown in Figure 7e, which is still within the bounds of what we can consider normal data. From this we can conclude that although out-of-bounds reconstruction can be unwanted, in some cases it aligns with the expectations of an anomaly detector. More generally speaking we observe that some generalization of the autoencoder can align with the expectation of a user. In some cases, generalization can lead to unwanted inter- or extrapolation. This unwanted generalization can cause anomalous data to

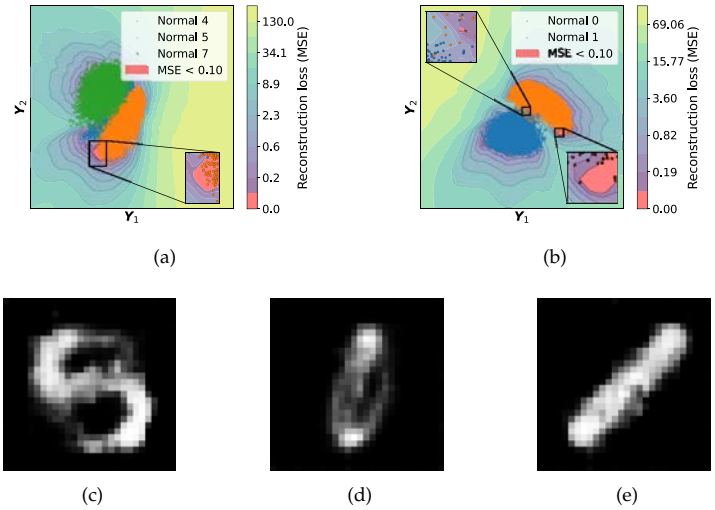


Figure 7: Plots of the contours of the reconstruction loss in the 2D latent space of a convolutional autoencoder when applied on subsets of MNIST (a, b), plots of constructed adversarial anomalies (c, d), and a plot of non-problematic out-of-bounds reconstruction (e). Subplots (a, c) show the results for an autoencoder trained on digits 4, 5, and 7, and Subplots (b, d, e) show the results for an autoencoder trained on digits 0, and 1. The visualized samples, i.e., the points in (a, b) are the latent representations of the training data. The shown digits are constructed by sampling from the $\epsilon < 0.1$ zone within the marked area, and correspond to these from left to right. The contour plot is colored red whenever the MSE is below a set threshold $\epsilon < 0.1$ to indicate a near-perfect reconstruction. Note that the color scaling is exponential to better visualize the MSE loss.

stay fully undetected. This is similar to the phenomena observed by Nalisnick et al. [26] for variational autoencoders, who observe some out-of-distribution samples to remain fully undetected.

While conducting the experiments on the MNIST data, we found that the problems shown above do not seem to arise in every case. We observe that depending on the random seed used for initialization and the digits selected as normal data, the out-of-bounds reconstruction may or may not be easily detected. In some cases, the out-of-bounds behavior seems very non-monotonous, meaning that the regions of reconstruction disappear one epoch, and reappear the next. This solidifies our belief that autoencoders may fail, but in many cases outwardly seem to work well. The crux lies in the fact that in a semi-supervised or unsupervised setting, it is not possible to accurately judge whether a network will fail on future data. This is further

complicated by the fact that due to the heterogeneity of anomalies, some may be detected while others go unnoticed.

3.5 CONCLUSION

In this work we provide an analysis of the unwanted reconstruction of anomalies that autoencoders can exhibit when used for anomaly detection. We move beyond existing theories of unwanted reconstruction happening in interpolation and show how unwanted out-of-bounds reconstruction can occur when extrapolating as well, and how this can lead to anomalies staying fully undetected. We show through several experiments that these issues can arise in real-world data and not just in theory. This leads us to some safety concerns, where autoencoders can catastrophically fail to detect obvious anomalies. This unreliability can have major consequences when trust is put into the anomaly detector in safety-critical applications.

In general, we solidify the growing belief that the reconstruction loss is not a reliable proxy for anomaly detection, especially when the network is explicitly trained to lower the reconstruction loss for normal data without constraining the reconstruction capability beyond the bounds of the normal training data such as has been done by Yoon, Noh, and Park [34]. We find that this issue is most prevalent for (conditionally) linear units such as the ReLU, but similar issues exist for sigmoid networks, albeit to a lesser degree. The reconstruction issue is mostly caused by the fact that a point in the lower-dimensional latent space corresponds to a hyperplane in the original space that the data occupies. Next to interpolation and out-of-bounds reconstruction, we find that anomalies can remain undetected when they occupy the latent space where normal classes border.

Users of autoencoders for anomaly detection should be aware of these issues. Good practice would be to at least check whether a trained non-linear autoencoder exhibits the undesirable out-of-bounds reconstruction. In this paper's illustrative examples, we checked for this by searching for adversarial anomalies. This was relatively easy, as it could be done either visually in the latent space, or through a simple 2D grid search. For more complex datasets, requiring larger latent spaces, a feasible strategy might be to again synthesize samples from the latent space and formulate the search for adversarial anomalies as an optimization in terms of projected gradient descent [24].

By describing exactly how autoencoders are unreliable anomaly detectors by describing anomaly reconstruction, we hope to provide a scaffold for future research on fixing and avoiding the identified issues in a targeted manner.

ACKNOWLEDGMENTS AND DISCLOSURE OF FUNDING

The research reported in this chapter has been partly funded by the NWO grant NWA.1160.18.238 (PrimaVera); as well as BMK, BMDW, and the State of Upper Austria in the frame of the SCCH competence center INTEGRATE [(FFG grant no. 892418)] part of the FFG COMET Competence Centers for Excellent Technologies Programme. We also want to thank Alex Kolmus and Marco Loog for the excellent input and discussions on this topic.

REFERENCES

[1] M. Astrid, M. Z. Zaheer, D. Aouada, and S.-I. Lee. "Exploiting autoencoder's weakness to generate pseudo anomalies." In: *Neural Computing and Applications* (2024), pp. 1–17.

[2] M. Astrid, M. Z. Zaheer, J.-Y. Lee, and S.-I. Lee. "Learning not to reconstruct anomalies." In: *arXiv preprint arXiv:2110.09742* (2021).

[3] P. Baldi and K. Hornik. "Neural networks and principal component analysis: Learning from examples without local minima." In: *Neural Networks* 2.1 (1989), pp. 53–58.

[4] Y. Bao, Z. Tang, H. Li, and Y. Zhang. "Computer vision and deep learning–based data anomaly detection method for structural health monitoring." In: *Structural Health Monitoring* 18.2 (2019), pp. 401–421.

[5] L. Beggel, M. Pfeiffer, and B. Bischl. "Robust anomaly detection in images using adversarial autoencoders." In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*. Springer. 2020, pp. 206–222.

[6] C. I. Bercea, D. Rueckert, and J. A. Schnabel. "What do aes learn? challenging common assumptions in unsupervised anomaly detection." In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2023, 304–314.

[7] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger. "The MVTec anomaly detection dataset: a comprehensive real-world dataset for unsupervised anomaly detection." In: *International Journal of Computer Vision* 129.4 (2021), pp. 1038–1059.

[8] R. Bouman, Z. Bukhsh, and T. Heskes. "Unsupervised Anomaly Detection Algorithms on Real-world Data: How Many Do We Need?" In: *Journal of Machine Learning Research* 25.105 (2024), pp. 1–34. URL: http://jmlr.org/papers/v25/23-0570.html.

[9] H. Bourlard and Y. Kamp. "Auto-association by multilayer perceptrons and singular value decomposition." In: *Biological Cybernetics* 59.4 (1988), pp. 291–294.

[10] Y. Cai, H. Chen, and K.-T. Cheng. "Rethinking autoencoders for medical anomaly detection from a theoretical perspective." In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2024, pp. 544–554.

[11] Z. Cheng, S. Wang, P. Zhang, S. Wang, X. Liu, and E. Zhu. "Improved autoencoder for unsupervised anomaly detection." In: *International Journal of Intelligent Systems* 36.12 (2021), pp. 7103–7125.

[12] J. K. Chow, Z. Su, J. Wu, P. S. Tan, X. Mao, and Y.-H. Wang. "Anomaly detection of defects on concrete structures with the convolutional autoencoder." In: *Advanced Engineering Informatics* 45 (2020), p. 101105.

[13] E. Cruz-Esquivel and Z. J. Guzman-Zavaleta. "An examination on autoencoder designs for anomaly detection in video surveillance." In: *IEEE Access* 10 (2022), pp. 6208–6217.

[14] T. Denouden, R. Salay, K. Czarnecki, V. Abdelzad, B. Phan, and S. Vernekar. "Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance." In: *arXiv preprint arXiv:1812.02765* (2018).

[15] F. Farahnakian and J. Heikkonen. "A deep auto-encoder based approach for intrusion detection system." In: *2018 20th International Conference on Advanced Communication Technology (ICACT)*. IEEE. 2018, pp. 178–183.

[16] X. Glorot, A. Bordes, and Y. Bengio. "Deep Sparse Rectifier Neural Networks." In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by G. Gordon, D. Dunson, and M. Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, 2011, pp. 315–323. URL: https://proceedings.mlr.press/v15/glorot11a.html.

[17] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. v. d. Hengel. "Memorizing Normality to Detect Anomaly: Memory-augmented Deep Autoencoder for Unsupervised Anomaly Detection." In: *IEEE International Conference on Computer Vision (ICCV)*. 2019.

[18] J. D. Havtorn, J. Frellsen, S. Hauberg, and L. Maaløe. "Hierarchical vaes know what they don't know." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 4117–4128.

[19] S. Hochreiter. "Untersuchungen zu dynamischen neuronalen Netzen." In: *Diploma, Technische Universität München* 91.1 (1991), p. 31.

[20] P. Kamat and R. Sugandhi. "Anomaly detection for predictive maintenance in industry 4.0-A survey." In: *E3S Web of Conferences*. Vol. 170. EDP Sciences. 2020, p. 02007.

[21] Y. LeCun. "The MNIST database of handwritten digits." In: *http://yann. lecun. com/exdb/mnist/* (1998).

[22] R. Lu, Y. Wu, L. Tian, D. Wang, B. Chen, X. Liu, and R. Hu. "Hierarchical vector quantized transformer for multi-class unsupervised anomaly detection." In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 8487–8500.

[23] O. Lyudchik. *Outlier detection using autoencoders*. Tech. rep. CERN, 2016.

[24] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. "Towards deep learning models resistant to adversarial attacks." In: *arXiv preprint arXiv:1706.06083* (2017).

[25] N. Merrill and A. Eskandarian. "Modified autoencoder training and scoring for robust unsupervised anomaly detection in deep learning." In: *IEEE Access* 8 (2020), pp. 101824–101833.

[26] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. "Do Deep Generative Models Know What They Don't Know?" In: *International Conference on Learning Representations*. 2019.

[27] R. Nayak, U. C. Pati, and S. K. Das. "A comprehensive review on deep learning-based methods for video anomaly detection." In: *Image and Vision Computing* 106 (2021), p. 104078.

[28] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut. "Network anomaly detection using LSTM based autoencoder." In: *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*. 2020, pp. 37–45.

[29] M. Salehi, A. Arya, B. Pajoum, M. Otoofi, A. Shaeiri, M. H. Rohban, and H. R. Rabiee. "Arae: Adversarially robust training of autoencoders improves novelty detection." In: *Neural Networks* 144 (2021), pp. 726–736.

[30] W. Sultani, C. Chen, and M. Shah. "Real-world anomaly detection in surveillance videos." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6479–6488.

[31] A. Tong, G. Wolf, and S. Krishnaswamy. "Fixing bias in reconstruction-based anomaly detection with lipschitz discriminators." In: *Journal of Signal Processing Systems* 94.2 (2022), pp. 229–243.

[32] D.-M. Tsai and P.-H. Jen. "Autoencoder-based anomaly detection for surface defect inspection." In: *Advanced Engineering Informatics* 48 (2021), p. 101272.

[33] Q. Wei, Y. Ren, R. Hou, B. Shi, J. Y. Lo, and L. Carin. "Anomaly detection for medical images based on a one-class classification." In: *Medical Imaging 2018: Computer-Aided Diagnosis*. Vol. 10575. SPIE. 2018, pp. 375–380.

[34] S. Yoon, Y.-K. Noh, and F. Park. "Autoencoding under normalization constraints." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12087–12097.

[35] Z. You, L. Cui, Y. Shen, K. Yang, X. Lu, Y. Zheng, and X. Le. "A unified model for multi-class anomaly detection." In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 4571–4584.

[36] Y. Zhao, B. Deng, C. Shen, Y. Liu, H. Lu, and X.-S. Hua. "Spatio-temporal autoencoder for video anomaly detection." In: *Proceedings of the 25th ACM International Conference on Multimedia*. 2017, pp. 1933–1941.

[37] Y. Zhou. "Rethinking reconstruction autoencoder-based out-of-distribution detection." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 7379–7387.

[38] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen. "Deep autoencoding gaussian mixture model for unsupervised anomaly detection." In: *International Conference on Learning Representations*. 2018.

4

ACQUIRING BETTER LOAD ESTIMATES BY COMBINING ANOMALY AND CHANGE POINT DETECTION IN POWER GRID TIME SERIES MEASUREMENTS

In this chapter we present novel methodology for automatic anomaly and switch event filtering to improve load estimation in power grid systems. By leveraging unsupervised methods with supervised optimization, our approach prioritizes interpretability while ensuring robust and generalizable performance on unseen data. Through experimentation, a combination of binary segmentation for change point detection and statistical process control for anomaly detection emerges as the most effective strategy, specifically when ensembled in a novel sequential manner. Results indicate the clear wasted potential when filtering is not applied. The automatic load estimation is also fairly accurate, with approximately 90% of estimates falling within a 10% error margin, with only a single significant failure in both the minimum and maximum load estimates across 60 measurements in the test set. Our methodology's interpretability makes it particularly suitable for critical infrastructure planning, thereby enhancing decision-making processes.

4.1 INTRODUCTION

The global energy landscape is undergoing a transformative shift towards sustainability, driven by the urgent need to mitigate climate change and reduce reliance on fossil fuels [38]. This process, more commonly known as the energy transition, presents a multitude of challenges that must be addressed to achieve a successful energy transition. These challenges encompass technical, economic, social, and political aspects, demanding innovative solutions and collaborative efforts on a global scale [18]. One of the key bottlenecks in implementing the energy transition in the Netherlands is the growth of electrical infrastructure [47]. In order to replace fossil fuel energy sources, the capacity of the power grid needs to increase significantly. However, increasing the capacity of the power grid involves several challenges. Due to a scarcity of resources [25] identification of key areas where additional capacity is most needed is imperative.

In addition to the need for additional capacity, the way the grid is being used is also changing [5]. Due to the increasing reliance on solar and wind energy, the centralized production of energy is a fading paradigm. Decentralized production in the form of a multitude of wind and solar parks, as well as solar panels covering a large percentage of urban housing, are changing the ways in which electricity is

distributed [28, 44]. Where previously gas was the primary source of heating in houses, heat pumps are increasingly being used in households [17]. Electrical cooling of houses has seen a substantial increase in recent years [12]. Electric motors are quickly replacing fossil fuel combustion engines in personal vehicles [40]. These vehicles are now often charged at home, at the workplace, or near other hubs [10]. These changing requirements drive a need to expand the power grid in a smart manner, where expansion is done there where it is most needed in the near future.

Next to expansion, smarter use of the existing grid is essential. This can be done through for example flexible energy contracts, the use of redundant grid capacity for power generation, and the use of batteries [33]. Better insight into grid usage over time is needed to facilitate these changes. With more measurements this can be achieved, but this means also more data that needs to be cleaned before it can be used for analysis.

In order to determine key points of expansion, an accurate overview of the current state of grid capacity needs to be made. In this study, we specifically study primary substation-level measurements on the Dutch power grid managed by Alliander. In order to know what percentage of a primary substation's capacity is being used, accurate estimates of the minimum and maximum load of the subgrid that substation supplies must be made. This process is more commonly known as demand modeling [3]. Primary substation-level time series load measurements, however, cannot be directly interpreted without giving a distorted view. These measurements are contaminated with anomalies, for example caused by measurement errors, and with switch events. In a switch event power from different primary substations is rerouted to or from part of the measured primary substation's route. An example of this can be found in Figure 8, where due to a cable failure power is rerouted from primary substation 1 to supply secondary substations E and F. Rerouting power reduces the blackout time in case of damage to a cable in the power grid, and is possible when the grid is sufficiently redundant to circumvent the broken cable. Switch events can also happen when long-term maintenance is performed on a primary substation or any its cables or secondary substations. Switch events can have a large range of possible lengths, from a few minutes, to multiple months, depending on how fast the underlying problem is resolved. Both of these, measurement errors, which we will call anomalies, and switch events, need to be filtered out to get an accurate estimate of the load profile. This more accurate load profile can be used directly for more optimized usage of the grid. It can then be used to find the true minimum and maximum load of the substation under normal operating conditions, so the redundancy capacity can be added separately. The task of detecting which parts

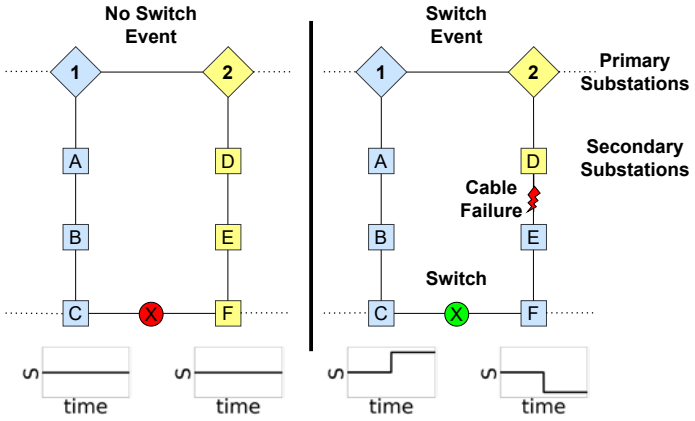


Figure 8: An illustration of a switch event. The diamonds numbered 1 and 2 indicate two primary substations. The squares named A-F indicate secondary substations. The circle named X indicates a switch. Solid lines indicates connections between substations, and dashed lines indicate connections to other substations outside of the figure. The primary substation and the secondary substations powered by it are colored light blue or light yellow for primary substations 1 and 2 respectively. When a switch event occurs, for example due to cable failure between secondary substations D and E, power will be supplied from primary substation 1 rather than 2. This is indicated by the switch X color change from red to green. This leads to a temporary increase in apparent power measured at 1 and a decrease at 2, as illustrated with the apparent power measurements as a function of time at the bottom of the figure.

of a load measurement represent the normal situation can be seen as a time series segmentation task.

Traditionally, these load measurement are manually segmented and annotated by domain experts within the Alliander organization. This procedure is however extremely time consuming, and produces annotations with a high percentage of label noise. This label noise is of little concern in determining the minimum and maximum loads, as the annotation procedure is generally optimized in such a way that these loads are as accurate as possible. The resulting annotation can however not be used for other purposes such as year-round usage insights. An automated annotation procedure, capable of producing high-quality annotations, can therefore both free up valuable domain expert time, as well as allow for detailed studies on load measurement data previously unfeasible. Because economically and logistically vital decisions are made based on these load estimates, it is extremely important that any machine learning model automating this procedure is highly interpretable.

For smart grids, change point and anomaly detection algorithms have been broadly applied. Zhang, Wu, and Boulet [46] describe multiple time series anomaly detection algorithms used in various applications across smart grids. They differentiate between point, contextual, and pattern anomalies. Thomas, Kurupath, and Nair [36] describe a novel method tailored to detecting islanding. In addition to islanding, their method, which uses k-means clustering and empirical mode decomposition, is able to detect load switch events and other faults. Neagoe et al. [26] used change point detection in order to detect multi-year patterns in hydropower generation in Romania. Wang and Ahn [41] combine a regression-based anomaly detection model with SVM, kNN and a cross-entropy loss function to detect anomalies in a Tanzanian solar power plant. Wang, Yao, and Papaefthymiou [42] present methodology for simultaneously performing load forecasting and semi-supervised anomaly detection and apply it to the UCI electric load dataset. They report higher specificity and sensitivity than neural network-based methods, but find similar F1 scores. For shorter time series, Rajabi et al. [30] have compared various clustering methods for load pattern segmentation.

Load estimation has been performed across a variety of use cases within power grid operation and study. Heslop, MacGill, and Fletcher [14] have estimated maximum photovoltaic generation for residential low voltage feeders. Mendes, Paiva, and Batista [24] estimate the variability of the load, using graph signal processing, on data with a high level of distributed generation, similar to the situation in the Netherlands. Langevin, Cheriet, and Gagnon [19] employ variational autoencoders for short-term forecasting of the load within households. Kara et al. [16] use multiple linear regression to disaggregate the solar generation in regular feeder measurements in order to get separate estimates for generation and production. Asefi et al. [1] perform a combination of statistical modeling, anomaly detection and classification in order to estimates states and identify false data injection.

However, to the best of our knowledge, for time series, of a year or longer in length, where event lengths vary substantially, no studies have been conducted on load estimation through means of automated segmentation.

4.2 MATERIALS AND METHODS

4.2.1 *Data*

In this study we optimize and evaluate our algorithms for anomaly and switch event detection on a total of 180 primary substation load measurements. Most primary substation measurements span a full year in length, and provide measurements at regular 15-minute intervals of the apparent power S, which we will refer to as load. We

calculate S from the active power P and the reactive power Q and assign it the sign of P, $S = sign(P)\sqrt{P^2 + Q^2}$. In some cases, the measurement equipment of a primary substation does not allow for accurate measurements of both P and Q. In these cases, S is calculated from $S = \sqrt{3}VI$. The $\sqrt{3}$ term originates from the fact that in a 3-phase system the phase voltage rather than the line voltage is used.

At each 15-minute interval over which the load is measured, we also calculate the so-called bottom-up load B throughout the subgrid supplied by that specific primary substation. The bottom-up load is an estimation of the load over a certain primary substation, but not a direct measurement, like S. Thus, the bottom-up load is directly related to the actual load measurement. The bottom-up is traditionally used by distribution system operators to get an estimate of load on the grid on places where no measurements are available. This bottom-up estimate tries to reconstruct the total load based on telemetry measurements from bulk consumers, from aggregated smaller scale measurements, and from average profiles based on smart meters at consumers' homes and some smaller bulk consumers. In the latter machine learning is used to estimate the load profiles of those consumers that do not have a smart meter, or have not consented to have their smart meter data read [13]. In order to acquire the final bottom-up load, Alliander uses the SunDance algorithm [6] for disaggregation of net consumption and generation, k-means clustering [22, 23] for generating load profiles, and XGBoost [7] for determining which clustered load profiles should be used instead of missing smart meter measurements [13]. More details regarding the bottom-up generation methodology can be found in [13]. Most often, the bottom-up load is fairly accurate. Most failure cases are not caused by the algorithm, but rather by incorrect grid-topology data, causing consumers to be wrongfully included or excluded.

In order to discern between a primary substation connection that is net consuming or net producing, the load measurements S are given a sign based on P. If a primary substation connection is consuming more than it is producing, the load measurement is assigned a positive sign. A negative sign therefore means that the primary substation is net producing. However, not all primary substations are outfitted with measurement equipment to determine whether the primary substation is net producing or consuming: they just measure the absolute current I, thus the sign needs to be corrected later on. The bottom-up load, in contrast to the actual load, can always be measured in the negative due to being based on more recent, lower-level, measuring equipment. We will use this property in correcting the sign of the load measurement. Bottom-up load measurements are furthermore based on P measurements and load estimates so they never have a missing sign. A typical primary substation measurement time series, consisting out of the load (S), the bottom-up load, and an illustrative

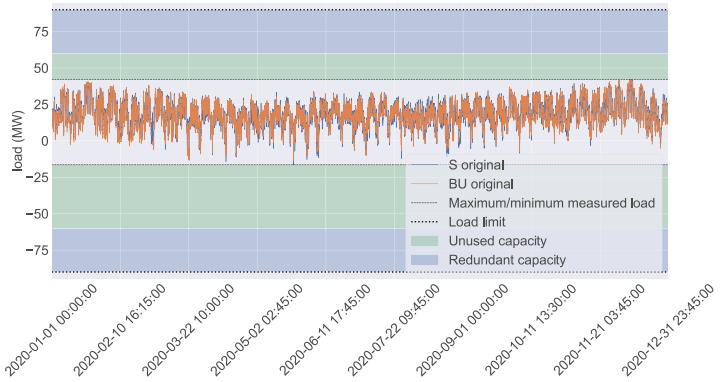


Figure 9: A plot of the measured load (S) and the bottom-up load (B) as measured or estimated over the entire year for station 005. The S measurement is visualized in blue, and the bottom-up load is visualized in orange. The minimum and maximum load estimates are shown by the dashed lines. The load limit of the primary substation is shown by the dotted line. The green and blue areas indicate the unused and redundant capacity, these are fictitious and only shown for illustrative purposes.

example of the needed minimum and maximum capacity is shown in Figure 9, where it should be noted that this measurement does not contain anomalies or switch events. In this figure we can additionally see the minimum and maximum loads (S) that are vitally important for the planning of grid expansion. On top of the maximum or minimum load, there should be enough redundant capacity in order to allow for rerouting in case of grid failures. All remaining capacity is unused, and quantification of this unused capacity is essential for determining whether the capacity of a station should be expanded, or whether there is room for additional customers. Should anomalies or switch events be present, we expect to see these minimum and maximum values be inflated or deflated, leading to inaccurate estimates of the unused capacity of a primary substation, thereby interfering with power grid expansion planning. An example of wrong capacity estimates leading to unused capacity can be found in Figure 10.

To summarize, at each 15-minute interval t for each station i we measure the load s_t^i and estimate the bottom-up load b_t^i. We do this for a full year of measurements, yielding for each station i a load vector $\mathbf{s}^i \in S$, and a bottom-up load vector $\mathbf{b}^i \in B$, where S and B respectively now denote the full collections of load and bottom-up vectors as measured for each of the 180 measured stations. In order to allow for evaluation of automatic segmentation and anomaly detection algorithms, each 15-minute measurement y_t^i has additionally been labeled, leading to a detailed labeled segmentation of each time series for each station i, $\mathbf{y}^i \in Y$, which we will treat as our gold stan-

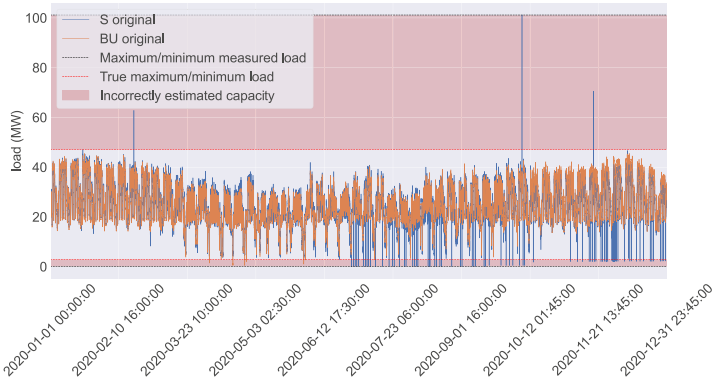


Figure 10: A plot of the measured load (S) and the bottom-up load (B) as measured or estimated over the entire year for station 010. The S measurement is visualized in blue, and the bottom-up load is visualized in orange. The minimum and maximum load estimates are shown by the black dashed lines. The load limit of the primary substation is shown by the dotted line. The true minimum and maximum load limits are shown by the red dashed lines. The capacity that would be incorrectly included in the estimate is shown by the opaque red boxes.

dard. Possible values for this label are: 0 (no anomaly or switch event); 1 (anomaly or switch event); 5 (the labeler is uncertain whether this should be label 0 or 1). For simplicity, the dependence of vectors $\mathbf{s}$, $\mathbf{b}$ and $\mathbf{y}$ on index of the station i will be omitted throughout the remainder of this chapter, and will only be mentioned explicitly when necessary.

The time series of all primary substations show great variation, both within a primary substation, but also between primary substations. The average load for a primary substation can range from 100's to 10,000's of kilowatts. In addition, not all bottom-up loads are equally accurate for each primary substation. This variability leads to a challenging segmentation task.

4.2.1.1 *Event-length Categories*

We use a variety of measures to judge the quality of the time series segmentation, see Section 4.2.4.1. Generally, these measures are calculated over all individual time points. This can be done when the segments are of somewhat similar length. In this use case however, anomalies are very short events, while switch events might be very long. This disparity in event length is illustrated in Figure 11. From this figure, we can clearly observe that long events are rare, but make up the majority of measured data labelled as 1, while short events/anomalies are much more frequent, but only make up a small

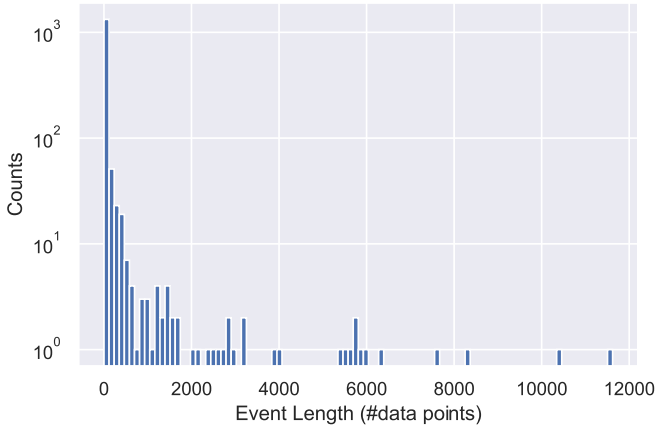


Figure 11: Histogram of the length of the events and anomalies over all datasets. Note that the y-axis is log-scaled due to the frequency of short events. A year typically consists out of 35040 15-minute interval measurements.

part of the label 1 data. In order to alleviate this problem while optimizing and evaluating our methods, we divide the anomaly/switch events in 4 categories based on their length based on how many samples they consists of, and calculate all measures, see Section 4.2.4.1 for each category. The 4 categories of event lengths are defined as: 1 up to and including 24 samples, 25 up to including 288 samples, 289 up to and including 4032 samples, and 4033 samples and longer. From here on out we will refer to these, for sake of clarity, as their equivalents in time units: 15 minutes to 6 hours, 6 hours to 3 days, 3 to 42 days, and 42 days or longer. Roughly speaking the categories contain the following types of events: 15 minutes to 6 hours generally contains measurement errors, 6 hours to 3 days contains longer measurement errors and very short switch events that are easily resolved, 3 to 42 days contains short switch events, as well as more complex rerouting, and 42 days or longer contains switch events caused by for example complex rerouting, long-term maintenance, and grid expansion. It should be noted that this categorization is not set in stone, and may be chosen slightly differently depending on the prevalence of event types in other applications.

4.2.2 *Preprocessing*

In order to make each time series suitable for further analysis, several preprocessing steps have been conducted. First, missing S measurements and corresponding bottom-up loads are removed from each

time series. Missing data can occur when there is a communication error in the system. Communication errors are typically characterized by repeated measurements or bottom-up loads. To correct for the discrepancies between the load measurement and the bottom-up load, we perform a linear regression to better match the two. The multiplication term corrects for multiplicative mismatches caused by over- or underestimating the amount of load of the customers and the grid losses that depend on the load (copper-losses), while the additive baseline correction corrects for constant grid losses, or iron losses, and, in the case of a current measurement, for the constant reactive power caused mainly by the capacitance of the cables. The linear regression is done on a subset of the time series, specifically by excluding everything outside user-defined quantiles $q_{lower}\%$ and $q_{upper}\%$. This is done to prevent that any anomalies or switch events steer the linear regression. Lastly, we perform a sign correction on S measurements as some measurement equipment cannot measure load signs. We list the preprocessing steps as pseudocode in Algorithm 1 in the order in which they are applied to each individual station measurement, i.e., all $\mathbf{s} \in S$ and $\mathbf{b} \in B$. Note that $\mathbf{s}^i$ and $\mathbf{b}^i$ are vectors of equal length. The used functions "bottomUpMissing" and "repeatedMeasurements" are described in Algorithm 2 and Algorithm 3 respectively.

As one can note, this procedure has several user-defined hyperparameters, which can be optimized. These are specifically the range of the quantiles, $q_{lower}\%$ and $q_{upper}\%$, used in the filtering procedure, as well as the number of $\mathbf{s}$ measurements, r, that have to be identical in order to be classified as missing due to a communication error. Based on manual observations of the preprocessing procedure, we have selected $q_{lower}\%$ and $q_{upper}\%$ to be 10% and 90% respectively, and r to be 5.

The resulting difference vector $\boldsymbol{\delta}$ is calculated for each primary substation, yielding the set of difference vectors Δ, and then used for further analysis. $\boldsymbol{\delta}$ now represents the error between the actual measurement and the bottom-up load. Effectively we now have a feature vector where the variation that can be explained from the bottom-up load has been removed from the S measurement. This difference vector is used as input for the various segmentation and anomaly detection algorithms we apply, all listed in section 4.2.3.

4.2.3 *Algorithms and Optimization*

In this research, we compare several methods for detecting anomalies and switch events. Each of these takes the difference vector between the measurement and the bottom-up load $\boldsymbol{\delta}^i \in \Delta$ as input data. The output of each base method is a vector of unbounded scores for each primary substation z^i, thus yielding a set of score vectors Z. These

Algorithm 1 Preprocessing procedure

Input: Measurement $\mathbf{s}$ and bottom-up load $\mathbf{b}$
Hyperparameters: The maximum number of repeated measurements r, and the quantile boundaries used for scaling before the linear fit $q_{lower}\%$, $q_{upper}\%$
Output: Difference vector $\boldsymbol{\delta}$

for $i \leftarrow 1$ to n **do** ▷ Remove missing and repeated measurements
    if bottomUpMissing(b_i) **or** repeatedMeasurements(s, i, r) **then**
        Remove s_i from $\mathbf{s}$ and b_i from $\mathbf{b}$ ▷ See Algorithm 2 and 3.
    end if
end for
$\boldsymbol{\delta}_{temp} \leftarrow \mathbf{s} - \mathbf{b}$ ▷ Calculate the temporary difference vector
$q_{min} \leftarrow \text{quantile}(\mathbf{s}, q_{lower}\%)$
$q_{max} \leftarrow \text{quantile}(\mathbf{s}, q_{upper}\%)$
$\mathbf{s}_{filtered} \leftarrow s_i \in \mathbf{s}$ where $s_i > q_{min}$ and $s_i < q_{max}$
$\mathbf{b}_{filtered} \leftarrow$ corresponding elements $\mathbf{b} \in B$
$\mathbf{s}_{filtered} = m\mathbf{b}_{filtered} + c$ ▷ Fit linear model to find slope m and offset c
$\mathbf{b}_{scaled} \leftarrow m\mathbf{b} + c$ ▷ Rescale the bottom-up to match S
▷ Correct the sign of $\mathbf{s}$ if the minimum of s is positive while the minimum of b is negative.
if $\min(\mathbf{s}) \geqslant 0$ **and** $\min(\mathbf{b}_{scaled}) < 0$ **then**
    $\mathbf{s}_{signed} \leftarrow \mathbf{s} \circ \text{sign}(\mathbf{b}_{scaled})$
else
    $\mathbf{s}_{signed} \leftarrow \mathbf{s}$
end if
$\boldsymbol{\delta} \leftarrow \mathbf{s}_{signed} - \mathbf{b}_{scaled}$ ▷ Calculate the difference vector

Algorithm 2 bottomUpMissing

Input: A single bottom-up measurement b_i
Output: Boolean β indicating whether the bottom-up load measurement b_i is missing

$$\beta \leftarrow \begin{cases} \text{True}, & \text{if } b_i = \text{NaN} \\ \text{False}, & \text{otherwise} \end{cases}$$

Algorithm 3 repeatedMeasurements

Input: Measurement $\mathbf{s}$ and index i
Hyperparameters: The maximum number of repeated measurements r
Output: Boolean β indicating whether the maximum allowed number of repeated measurements and adjacent to s_i is exceeded

$\beta \leftarrow$ False
$c \leftarrow 0$ ▷ Repeated measurement counter
$max\_c \leftarrow 0$
for $j \leftarrow \max(i-r, 0)$ to $\min(i+r, n)$ **do** ▷ n is the length of the vector $\mathbf{s}$
    if $s_j = s_i$ **then**
        $max\_c \leftarrow \max(c, max\_c)$
    else
        $c \leftarrow 0$
    end if
end for
if $max\_c \geqslant r$ **then**
    $\beta \leftarrow$ True
end if

scores are subsequently converted to predicted binary label vectors for each primary substation $\tilde{\mathbf{y}}^i$, yielding a set of predicted label vectors $\tilde{Y}$. This is done by thresholding the scores, on which more details can be found in section 4.2.4.3.

We specifically make use of unsupervised anomaly detection algorithms. Unsupervised algorithms do not learn from labeled data, but only consider the measurements. We have annotated a fairly large number of 180 yearly primary substation measurements, but recognize that anomalies and switch events are very rare and heterogeneous events. Because of this, we applied unsupervised methods of detection. We then assume that some higher level hyperparameters of the unsupervised procedure, specifically the thresholds used for acquiring labels, will generalize towards unobserved types of anomalies and switch events, we will evaluate this assumption on the test set.

We have applied 3 base detectors, specifically statistical process control, isolation forest, and binary segmentation, and describe them in section 4.2.3.1. We compare these detectors separately, but also ensemble them by combining them in several ways. We compare a naive ensemble method, a distinct optimization criterion ensemble, and a sequential ensemble, which we describe in section 4.2.3.2.

4.2.3.1 *Base Detectors*

STATISTICAL PROCESS CONTROL Under the assumption that the data is stationary we can use classical statistical process control, or SPC, methods [27] in order to detect anomalies and switch events. In traditional SPC one assumes a process which is stationary within a chosen time frame or segment. Then, the user defines certain lower and upper control limits, typically 2 or 3 standard deviations away from the mean of the segment. In our case, the control limits will be optimized as either symmetric or asymmetric thresholds, rather than using the traditional statistical approach. Additionally, due to the known presence of anomalies in the data, we also look at the distance to the median, rather than to the mean. We will explicitly refer to the optimized control limit as "threshold" from here on out. When a time point falls outside of these control limits, it is flagged as out-of-control, in this case meaning anomalous. How SPC is applied to each difference vector $\delta \in \Delta$ resulting from preprocessing is described in Algorithm 4

Algorithm 4 Statistical process control

1: **Input:** Difference vector δ
2: **Hyperparameters:** Quantile boundaries for scaling $q_{\text{lower}}\%$, $q_{\text{upper}}\%$
3: **Output**: Score vector z
4:
5: $m \leftarrow \text{median}(\delta)$ ▷ Calculate median
6: $d \leftarrow \text{interquantileDistance}(\delta, q_{\text{lower}}\%, q_{\text{upper}}\%)$ ▷ Calculate interquantile distance
7: $z \leftarrow (\delta - m)/d$

It should be noted that the relevant hyperparameters here are the quantiles that are chosen for the interquantile range. These can be distinct from the quantiles hyperparameters used in the preprocessing procedure.

ISOLATION FOREST One of the most commonly used machine learning methods for anomaly detection is the isolation forest, or IF [21]. The isolation forest is known as one of the best state-of-the-art anomaly detectors on real-valued static data [4]. An isolation forest on one-dimensional data effectively produces a density estimate by randomly splitting subsets of the data. We consider two distinct ways of applying isolation forests on the data: one where we apply a single forest per station difference vector, see Algorithm 5, and one where we scale and concatenate all difference vectors for training the isolation forest and apply that isolation forest on each difference vector, see Algorithm 6. Note that in these algorithms, we use several high-level functions. "fitPredictIsolationForest" fits an isolation forest with

$n_{estimators}$ trees on the input δ and returns the anomaly scores $\tilde{z}$ on the same input. "fitIsolationForest" fits an isolation forest with $n_{estimators}$ trees on the input δ and returns the fitted model γ. "predictIsolationForest" returns the anomaly scores $\tilde{z}^i$ calculated over δ^i_{scaled} given an already fitted isolation forest γ. Each of these functions is implemented as part of the scikit-learn library [29].

Algorithm 5 Isolation forest per station

Input: Difference vector δ
Hyperparameters: The number of trees $n_{estimators}$
Output: Score vector z

$\tilde{z} \leftarrow \text{fitPredictIsolationForest}(\delta, n_{estimators})$
$z \leftarrow -\tilde{z} + 1$ ▷ Rescale so higher score means more anomalous

Algorithm 6 Single isolation forest over all stations

Input: Difference vectors $\delta^i \in \Delta$ for each station
Hyperparameters: The number of trees $n_{estimators}$, quantile boundaries for scaling q_{lower}%, q_{upper}%
Output: Score vectors $z^i \in Z$ for each station

for $\delta^i \in \Delta$ **do**
    $m \leftarrow \text{median}(\delta^i)$
    $d \leftarrow \text{interquantileDistance}(\delta^i, q_{lower}\%, q_{upper}\%)$
    $\delta^i_{scaled} \leftarrow (\delta^i - m)/d$
end for
$\Delta_{scaled} \leftarrow \{\delta^i_{scaled}, ..., \delta^n_{scaled}\}$
$\gamma \leftarrow \text{fitIsolationForest}(\Delta_{scaled}, n_{estimators})$
for $\delta^i_{scaled} \in \Delta_{scaled}$ **do**
    $\tilde{z}^i \leftarrow \text{predictIsolationForest}(\gamma, \delta^i_{scaled})$
    $z^i \leftarrow -\tilde{z}^i + 1$ ▷ Rescale so higher score means more anomalous
end for

In either procedure, we need to set the hyperparameters of the isolation forest. In the case where a single isolation forest is applied, we also consider the quantile boundaries for scaling. Furthermore, the choice between fitting a single forest or a forest per station is treated as a hyperparameter. We perform rescaling of the scores so that the algorithms deems the most anomalous samples to have the highest scores. These definitions differ between different algorithms in literature, but we chose this definition to be analogous to SPC, allowing for a similar threshold optimization strategy.

BINARY SEGMENTATION Binary segmentation is a change point detection algorithm able to find multiple change points in a given time series [2, 32]. We use binary segmentation as a state-of-the-art change point algorithm because it is found to be one of the best performing algorithms on real-world univariate time series data [9, 31, 39]. It finds change points by recursively partitioning the time series into two parts, forming a binary tree. A split occurs at the optimal break point. This break point is found by first calculating the cost c_{total} of the entire segment using a chosen cost function. Then the costs of the two subsegments, c_{left} and c_{right}, are calculated for each possible break point using the same cost function. The optimal break point is then found by selecting the one for which the gain, $g = c_{total} - c_{left} - c_{right}$ is maximized. This procedure is repeated until the gain for a split is below a user-defined threshold called the penalty p. In our experiments, we consider two penalties, a linear and a L1 penalty [39], the scaling of which depends on a user defined β parameter.

We use binary segmentation in order to generate scores by performing the following procedure explained in Algorithm 7. The "findReferenceValue" function is described in more detail in Algorithm 8. The "findBreakpointsBinarySegmentation" is a call to the high-level "ruptures" Python library [37] which takes the scaled vector $\tilde{\boldsymbol{z}}$ and segments it according to the well-known binary segmentation algorithm with hyperparameters β, C, l, and j.

Algorithm 7 Binary segmentation

Input: Difference vector $\boldsymbol{\delta}$
Hyperparameters: Cost function C, cost function weight β, minimum segment size l, the jump size j, quantile boundaries for scaling q_{lower}%, q_{upper}%, and the reference point strategy *reference_point*
Output: Score vector $\boldsymbol{z}$

$m \leftarrow \text{median}(\boldsymbol{\delta})$
$d \leftarrow \text{interquantileDistance}(\boldsymbol{\delta}, q_{lower}\%, q_{upper}\%)$
$\tilde{\boldsymbol{z}} \leftarrow (\boldsymbol{\delta} - m)/d$

$\mathbf{b} \leftarrow \text{findBreakpointsBinarySegmentation}(\tilde{\boldsymbol{z}}, \beta, C, l, j)$
$r \leftarrow \text{findReferenceValue}(\tilde{\boldsymbol{z}}, \mathbf{b}, \textit{reference\_point})$
$b_{begin} = 1$ ▷ indexing begins at 1
for $b_{end} \in \mathbf{b}$ **do**
    $\mathbf{t} \leftarrow (\tilde{\boldsymbol{z}}_i)_{b_{begin} \leqslant i \leqslant b_{end}}$ ▷ Get segment
    $(\boldsymbol{z}_i)_{b_{begin} \leqslant i \leqslant b_{end}} \leftarrow \text{mean}(\mathbf{t}) - r$ ▷ Get difference between segment and reference value
    $b_{begin} = b_{end}$
end for

Algorithm 8 findReferenceValue

Input: Scaled difference vector $\tilde{\mathbf{z}}$, and breakpoints resulting from binary segmentation $\mathbf{b}$
Hyperparameters: Reference point strategy *reference_point*
Output: Reference point value r

if *reference_point* = *"mean"* **then**
    $r = \text{mean}(\tilde{\mathbf{z}})$
else if *reference_point* = *"median"* **then**
    $r = \text{median}(\tilde{\mathbf{z}})$
else if *reference_point* = *"longest_mean"* **then**
    $s_{max} = 0$
    $b_{begin} = 1$ ▷ indexing begins at 1
    for $b_{end} \in \mathbf{b}$ **do**
        $\mathbf{t} \leftarrow (\tilde{\mathbf{z}}_i)_{b_{begin} \leqslant i \leqslant b_{end}}$ ▷ Get segment
        $s_i \leftarrow b_{end} - b_{begin}$ ▷ Get segment size
        if $s_i > s_{max}$ **then**
            $s_{max} = s_i$
            $r = \text{mean}(\mathbf{t})$
        end if
        $b_{begin} = b_{end}$
    end for
else if *reference_point* = *"longest_median"* **then**
    $s_{max} = 0$
    $b_{begin} = 1$ ▷ indexing begins at 1
    for $b_{end} \in \mathbf{b}$ **do**
        $\mathbf{t} \leftarrow (\tilde{\mathbf{z}}_i)_{b_{begin} \leqslant i \leqslant b_{end}}$ ▷ Get segment
        $s_i \leftarrow b_{end} - b_{begin}$ ▷ Get segment size
        if $s_i > s_{max}$ **then**
            $s_{max} = s_i$
            $r = \text{median}(\mathbf{t})$
        end if
        $b_{begin} = b_{end}$
    end for
end if

This procedure has a large number of optimizable hyperparameters resulting from the binary segmentation algorithm, the strategy for determining the reference point, and with which quantiles scaling should be applied. We have compared 4 different strategies for determining the reference point: 1) comparing to the mean of the entire scaled station difference vector (*"mean"*), 2) comparing to the median of the entire scaled station difference vector(*"median"*), 3) comparing to the mean of the longest segment (*"longest_mean"*), and 4) comparing to the median of the longest segment (*"longest_median"*).

4.2.3.2 *Ensembles*

The different base detection algorithms have various strengths and weaknesses. Specifically, binary segmentation is good at detecting long events, while SPC and IF are good at detecting shorter events. Binary segmentation more easily detects long events as it can compare the distance of an entire segment to the distance of a different segment, thus being more sensitive to small changes over a long time period. SPC and IF consider each time point individually, thus only detecting large changes without considering the time component. In order to leverage the strengths of multiple complementary methods, we employ different ensembling techniques to combine base detection algorithms. We specifically compare naive ensembling through combining predictions directly, combining algorithms optimized on detecting events of different lengths, and sequential ensembles, where we apply binary segmentation and apply SPC or IF only on the "normal" segments. A more detailed description of each ensembling strategy can be found below. For all ensembles, we look at a combination of a change point detector, binary segmentation, and an anomaly detector, which is either SPC or IF. As both SPC and IF are designed to find singular anomalies, we see them as complementary to binary segmentation, and will specifically compare SPC to IF.

NAIVE ENSEMBLES The simplest way of combining base detection algorithm predictions is by what we call naive ensembling. In naive ensembling, we simply take the predictions of each base detection algorithm, and combine them using an OR operation, meaning when either algorithm predicts an anomaly or switch event, so does the ensembled prediction.

DIFFERENT OPTIMIZATION CRITERION ENSEMBLES As noted earlier, it is known that binary segmentation is good at detecting long events, while SPC and IF are good at detecting shorter events. By simply combining predictions as in naive ensembling, the individual strengths of the sub-models are not leveraged to their fullest extent. We can instead optimize both detection models on different criteria. In this case, we have optimized binary segmentation to detect longer events in the "3 to 42 days", and "42 days and longer" length categories, whereas we optimize either SPC or IF on the "15 minutes to 6 hours", and "6 hours to 3 days" length categories. Then, like in naive ensembling, we combine the predictions of both separately optimized algorithms using an OR operation. Due to the category specific optimization we expect this method to find fewer false positives than naive ensembling and have a higher recall. For clarity's sake, we will abbreviate this type of ensemble as DOC, Different Optimization Criterion, ensembles.

SEQUENTIAL ENSEMBLES Sequential ensembles follow the same idea of using differently optimized detectors. In a sequential ensemble we first apply binary segmentation optimized on the "3 to 42 days", and "42 days and longer" length categories. Then, all segments that were not classified as switch events are passed to a second detection algorithms, either SPC or IF, to detect shorter events and anomalies. The second method is again optimized for detecting the shorter event categories "15 minutes to 6 hours", and "6 hours to 3 days". Due to the further specificity of this method, we expect to find fewer false positives, and to have a higher recall, especially on the shorter events which are no longer being masked by long switch events.

4.2.4 *Evaluation and Optimization*

4.2.4.1 *Evaluation Metrics*

Typically, time series segmentation is evaluated using precision, recall, the ROC/AUC, and the Fβ score [9, 31]. In this research, we focus on a weighted approach of the precision, recall, and the Fβ score. Rather than calculating them directly on a sample-to-sample basis, we calculate them for each of the 4 defined length categories. We do this so we can accurately detect events across the entire spectrum of lengths, as the number of measurements of events in each category increases greatly for increasing event lengths. For each of the length categories, we calculate the precision, recall, and the F1.5 score. We use $\beta = 1.5$ to give a higher importance to the recall term, as the potential impact of a false negative is higher than that of a false positive in power grid expansion planning. When we calculate a score for a length category, we do not include timestamps where either the assigned label is 5 ("uncertain"), or where the assigned label is 1 for any of the other categories. In order to get an estimate of the overall performance of the model, we average these measures over the four categories. The metric used to optimize any (hyper)parameter is the average F1.5 score over all 4 length categories. We explicitly choose to study the precision, recall, and the Fβ scores. In the practical use case we describe, the predictions are always binary, even though they are based on real-valued scores. As we explicitly threshold the scores to labels in order to filter our data, the precision, recall, and the Fβ score most closely illustrate the performance trade-off that is being made by thresholding. While we do not explicitly study the ranking of the anomalies and switch events, as is commonly done by the ROC/AUC metric, we do provide an additional plot plot of the performance in terms of the ROC/AUC in the appendix of this chapter in Figure 17.

Table 6: The distribution of event lengths over the train, test, and validation splits. The event count indicates how many events within that category are in a dataset. The label "1" count indicates how many separate time points belong to the anomaly/switch event class per dataset.

	Dataset	15m-6h	6h-3d	3d-42d	42d and longer
Event count	Train	338	136	24	4
	Validation	203	173	25	4
	Test	444	99	23	4
	All	985	408	72	12
Label 1 count	Train	1506	6290	28386	28212
	Validation	1971	13974	25075	30904
	Test	2262	6790	27389	24992
	All	5739	27054	80850	84108

4.2.4.2 *Validation*

In order to optimize thresholds, hyperparameters and to compare models, we split our original dataset consisting out of 180 stations into 3 equal parts of 60 stations each, creating a training, validation, and test dataset. This splitting procedure was done in a stratified manner such that the three splits have a more or less equal distribution of event lengths. The stations were divided by aiming to have an equal number of events for each category present in each dataset. The computational complexity of evaluating all possible combinations, like is done in available cross-validation software, is too high. We have therefore used a greedy approach to produce a stratified split of the data, where stations were divided starting with the longest, and least frequent, event length category, and ending with the shortest and most frequent event length category. All stations with no events were subsequently divided among the datasets in order to create 3 equally large datasets. The number of events, as well as the label "1" counts, can be found in Table 6.

From this table we can observe that the splitting of events in the longer categories is fairly balanced, whereas there is some imbalance in the shorter event categories. Even though the data is split in a stratified manner, such an imbalance can occur when single stations have a lot of events. This imbalance is most notable in the validation set, which has fewer "15 minutes to 6 hours" events, but more "6 hours to 3 days" events. Because these two categories are optimized on together, even within ensembles, we argue that this imbalance does not affect our conclusions.

After splitting, the train set is used to optimize the thresholds and train the isolation forest or just optimize the thresholds. The validation set is used to select the best performing hyperparameters for each method. Lastly, the test set is used to compare methods using only the best performing hyperparameters as evaluated on the validation set.

To get an estimate of the reliability of each method, we perform bootstrapping [8]. We do so by resampling the test set stations with replacement 10,000 times. In this way, we acquire both a bootstrapped mean and standard deviation for the precision, recall, and F1.5 metrics.

4.2.4.3 *Threshold Optimization*

All methods we have applied produce scores $z \in Z$. These can either be positives scores, where a higher value indicates a higher likelihood for a sample to be an anomaly or switch event according to the model, or scores centered around 0, where a greater distance to 0 indicates a higher likelihood. Of the presented methods, isolation forest has a purely positive score, while the statistical process control and binary segmentation scores are centered around 0. These scores are thresholded to yield a label prediction vector $\tilde{y} \in \tilde{Y}$. This is done by applying a one-sided or symmetrical approach, $\tilde{y} = \text{thresholdScores}(\text{abs}(z), \theta^{\text{symmetrical}})$, or a two-sided or asymmetrical approach, $\tilde{y} = \text{thresholdScores}(z, \theta^{\text{lower}}, \theta^{\text{upper}})$. The symmetrical approach can be used on both positive and zero-centered scores, whereas the asymmetrical approach can only be used on the zero-centered scores. Pseudocode for the one-sided approach can be found in Algorithm 9, and for the two-sided approach in Algorithm 10.

Algorithm 9 thresholdScores (one-sided)

1: **Input:** A score vector z, where a higher z_i indicates a higher likelihood of being an anomaly/switch event
2: **Hyperparameters:** A threshold $\theta^{\text{symmetrical}}$
3: **Output:** Predicted label vector $\tilde{y}$
4: **for** $z_i \in z$ **do**
5:

$$\tilde{y}_i \leftarrow \begin{cases} 1, & \text{if } z_i \geqslant \theta^{\text{symmetrical}} \\ 0, & \text{otherwise} \end{cases}$$

6: **end for**

The thresholds are optimized by selecting those thresholds for which the average of the F1.5 over all 4 segment length categories is highest. This can be done efficiently by calculating the F1.5 score for each possible threshold value for all distinct segment length categories, and

Algorithm 10 thresholdScores (two-sided)

1: **Input:** A score vector z, where a high or low z_i indicates a higher likelihood of being an anomaly/switch event
2: **Hyperparameters:** A lower threshold θ^{lower}, and upper threshold θ^{upper}
3: **Output:** Predicted label vector $\tilde{y}$
4: **for** $z_i \in z$ **do**
5:

$$\tilde{y}_i \leftarrow \begin{cases} 1, & \text{if } z_i \geqslant \theta^{\text{upper}} \\ 1, & \text{else if } z_i < \theta^{\text{lower}} \\ 0, & \text{otherwise} \end{cases}$$

6: **end for**

then averaging these profiles. An illustration of this selection procedure for the symmetrical approach can be found in Figure 12. In this figure we visualize the procedure for the SPC model with optimal hyperparameters. We further discuss the performance of this model in Section 4.3. We can formalize this optimization as finding that threshold θ, or those thresholds $\theta^{\text{lower}}, \theta^{\text{upper}}$ that maximize the average F1.5 score

$$\theta_{\text{optimal}} = \arg\max_{\theta} \text{F1.5}_{\text{average}}(Y, \text{thresholdScores}(\text{abs}(Z), \theta))$$

in the symmetrical case, or

$$\theta^{\text{lower}}_{\text{optimal}}, \theta^{\text{upper}}_{\text{optimal}} = \arg\max_{\theta^{\text{lower}}, \theta^{\text{upper}}} \text{F1.5}_{\text{average}}(Y, \text{thresholdScores}(Z, \theta^{\text{lower}}, \theta^{\text{upper}}))$$

in the two-sided case.

Whether to use one- or two-sided optimization is treated as a hyperparameter in the evaluation and selection process.

4.2.5 *Implementation and Reproducibility*

Our analysis has been done in the Python programming language, specifically version 3.10.0. Many of our calculations rely on NumPy [11] and Pandas [35]. We furthermore make use of the Ruptures[37] package for binary segmentation, and use the Scikit-learn [29] package for scaling procedures, as well as applying the isolation forest. Visualizations were made using the Seaborn [43] and Matplotlib [15] packages. In order to reproduce all our experiments, we have provided access to a public GitHub repository[1]. This reproduces the data splitting procedure, all experiments, including hyperparameter optimization, as

1 The Git repository can be found at: https://github.com/RoelBouman/StormPhase2

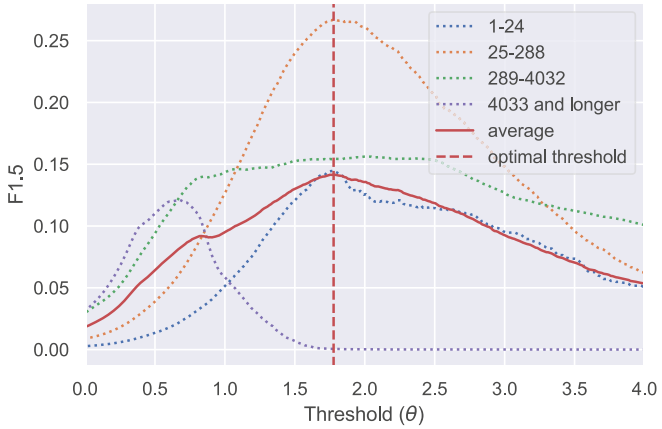


Figure 12: Plot of the one-sided threshold optimization procedure. The F1.5 score, on the y-axis, as a function of the threshold, on the x-axis, is shown for all four distinct segment length categories, as well as their average. The red vertical line indicates the selected threshold which maximizes the F1.5 score on the average. As an example the optimization is visualized for an SPC model with optimal hyperparameters.

well as producing all figures and tables in this chapter. All data is provided through Alliander[2]. An overview of all evaluated, as well as optimal, hyperparameters can be found in this chapter's appendix in Tables 12, 13 and 14.

4.3 RESULTS

After evaluating each method on the validation set, we have selected for each method and ensembling combination the model with the best performance on the validation set. Each of these models was then evaluated on the test set to get an estimate of how well optimized models perform on unseen data. The resulting performance in terms of the F1.5 is visualized in Figure 13. From this figure we can observe the performance of the three base learners, as well as the effectiveness of the ensembling strategies when applied to different combinations.

We find that IF and SPC perform similarly across the board. This is partially to be expected, as they are designed to find short anomalies/events that are rare compared to normal data. IF, however is able to more easily model multimodal distributions and detect anomalies in the presence of multimodality. Since IF does not noticeably outperform SPC, it seems that it can not leverage this advantage. This

2 The data repository can be found at https://www.liander.nl/over-ons/open-data

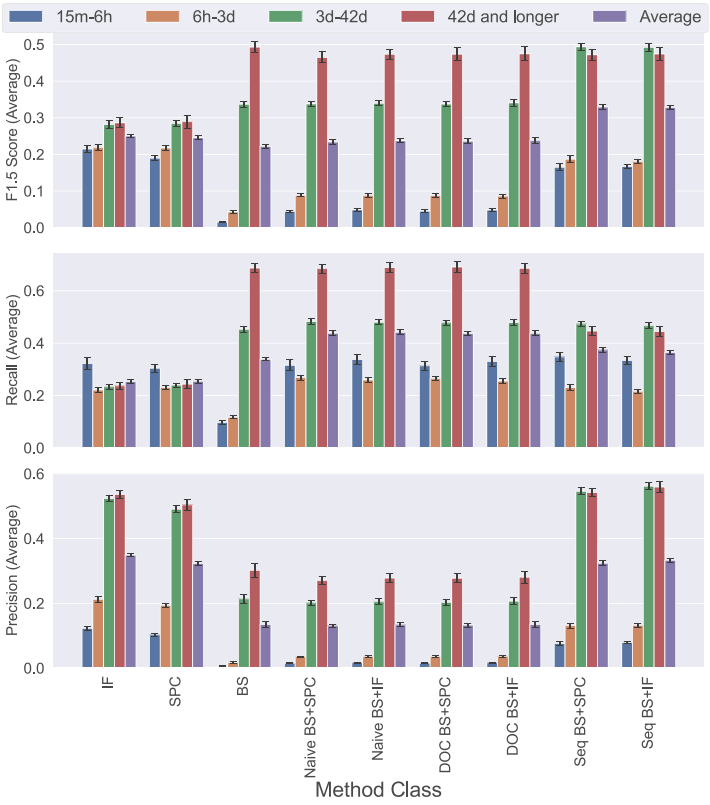


Figure 13: From top to bottom: bar plots of the results of each method per length category for the F1.5, recall, and precision respectively. The height of each bar indicates the average score over the bootstrap iterations. The error bars indicate the standard deviation resulting from the bootstrap resampling.

is further corroborated by visualization of typical scaled data, which is mostly unimodal. both IF and SPC are able to detect short anomalies/events better than binary segmentation, achieving a F1.5 score of approximately 0.2 for both the "15 minutes to 6 hours", and "6 hours to 3 days" event length categories. Perhaps surprisingly, these methods are able to detect some of the longer switch events as well, achieving just below 0.3 F1.5 scores. The relatively good performance on these longer events can be easily explained. Many of the longer switch events have a larger distance to the δ median than normal data. SPC is therefore able to detect these events based on this distance, while IF identifies them because these switch events occupy a lower density region where they are not masked by normal data. We find that even when ensembled in any of the three tested manners, IF offers no significant performance boost over SPC. Due to the ease of interpretability of SPC, it is preferable over IF.

Binary segmentation is found to be much better at detecting longer events than either SPC or IF. The longer the event, the better binary segmentation will perform, as can be seen from the performance on the longest category, which nears a F1.5 score of 0.5. It is also clear that binary segmentation fails to detect most short events/anomalies, achieving near random baseline F1.5 scores.

The results from the naive and different optimization criterion (DOC) ensembles are less obvious. Both methods perform similarly, and the different optimization criterion addition does not significantly improve overall model performance. Perhaps surprisingly, the F1.5 score for the two shorter event length categories is only marginally better than that of binary segmentation. Indeed, ensembling binary segmentation with a detector for shorter events barely increases the F1.5 score. This can be explained when we consider the two terms making up the Fβ score: precision and recall. When ensembling using an OR operation, we expect the recall to always increase, as we will only classify more samples in category "1". The precision, however, will generally decrease, as more false positives will be found. Indeed, when looking at the performance in terms of recall and precision, which we have visualized in Figure 13, we can readily observe that the recall only increases or stays the same when ensembling. From this we conclude that the number of false positives introduced by OR ensembling causes simple ensemble methods to underperform.

Sequential ensembles seemingly do not suffer from this increase in false positives as much. We can see that sequential ensembles nearly match the performance of the individual SPC and IF detectors in the single categories where they perform best, while the performance in the "3 to 42 days" category is significantly better than any base detector or other ensemble. The performance in the longest category is comparable to that of the other ensembles. Sequential ensembles indeed outperform any base detector or other ensembling method.

While intuitively one might think the performance increase in the shorter categories should be similar to other ensembling strategies, it seems that due to the sequential nature, we can optimize more on precision than is the case in other ensembles. This can also be observed in Figure 13, where we see that the recall of the sequential models is lower in the longest category, while being similar in the other three. This means that all performance increases from naive or DOC ensembles to sequential can be attributed to a higher precision, which is confirmed by Figure 13.

To further study the behaviour of one of the sequential ensembles, specifically the combination of BS and SPC, we have visualized one of the station difference vectors together with the predictions, thresholds, and reference points from both components of the sequential ensemble in Figure 14. From this figure we can observe several qualities, as well as failings, of the sequential ensembling approach. Most prominent is the correct classification of the switch event on the far right of the figure. Binary segmentation was able to correctly classify this. Also correctly detected are the 3 shorter events/anomalies to the left of the figure, which SPC has detected due to their large negative contribution. However, some mistakes are made by the method. Specifically some false positives arise in the middle of the figure, where some points fall just outside the detection boundaries. Then, on the right in the second segment, there is a mix of true positives and false negatives. Due to the variability of the signal, only a few time points within this event are accurately detected by SPC. Lastly, one could argue that the second segment in its entirety should have been classified as a switch event. Yet, this was neither done by the domain expert, nor by the binary segmentation algorithm. Upon closer inspection, we found that the information in the load measurement and the bottom-up load is insufficient to fully determine whether this was a switch event. From this, and other observations on similar stations, we are led to conclude that further improvements can be made by including more metadata in future endeavors to improve our algorithms. This is further reinforced by the performance of all methods, which is relatively low across the board, even on the train data. This indicates that the problem is hard to learn, though it generalizes fairly well. Additionally, with these visualizations, we show that these models, even when ensembled, are exceedingly interpretable, solidifying their use in applications of societal importance.

In order to gain more insight into how the proposed filtering approach works for automatically acquiring load estimates, we have plotted the ground truth load estimates against the predictions or unfiltered load estimates in Figure 15 for the maximum load estimates, and in Figure 16 for the minimum load estimates. In these figures, a perfect prediction would lead to all points lying on the diagonal. As can be seen, when we do not apply any filtering approach on the data,

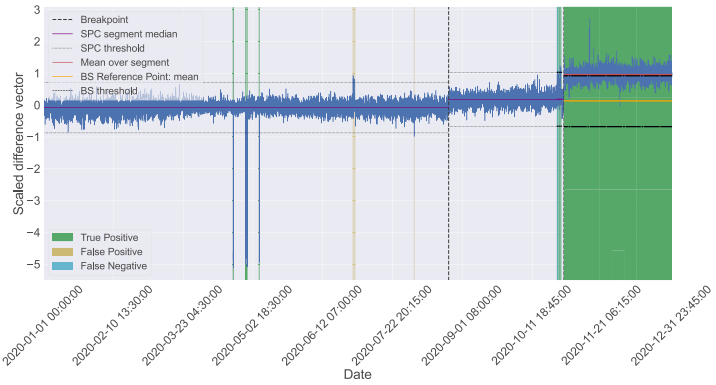


Figure 14: Plot of the results of the best sequential BS+SPC model on station "042" contained in the test set. The blue line indicates the inputted difference vector δ. The thresholds found by the initial BS pass are indicated by vertical dashed lines. The SPC segment medians and the BS overall mean used to calculate difference with the reference point are indicated by purple and orange lines. The boundaries for classification are indicated by either dotted or dashed lines for BS and SPC respectively. True positives, false positives and false negatives are visualized by changing the background color to green, yellow, or blue respectively.

the minimum and maximum load values are highly inflated, leading to a 300% increase at worst. A lot of potential grid capacity is unused when the measurements remain unfiltered and are being considered normal behavior. When we apply binary segmentation, we correctly predict some of the minimum and maximum loads, but still miss many while at the same time making the mistake of vastly underestimating the minimum and maximum load in a single case. With statistical process control the filtering procedure is much better, but we still have the worst case scenario of a 300% increase in maximum load prediction. By combining both using sequential ensembling, we leverage the strengths of either method, reducing the worst case scenario to an approximately 200% increase, and having only a single additional underestimation in the minimum load estimates. To further zoom in on the performance, we can see how well the method performs within certain error margins. The maximum load predictions are perfect in 75.00%, and within a 10% margin in 88.33% of all cases. The minimum load predictions are perfect in 86.96% and within a 10% error margin in 91.30% of all cases.

Due to the high performance of our ensemble method for load estimation it has been adopted for use within Alliander. Previously, all filtering and load estimates were done by hand. This was a time-consuming process, taking a month of time for several full-time employees. This was done once a year to acquire the load estimates for

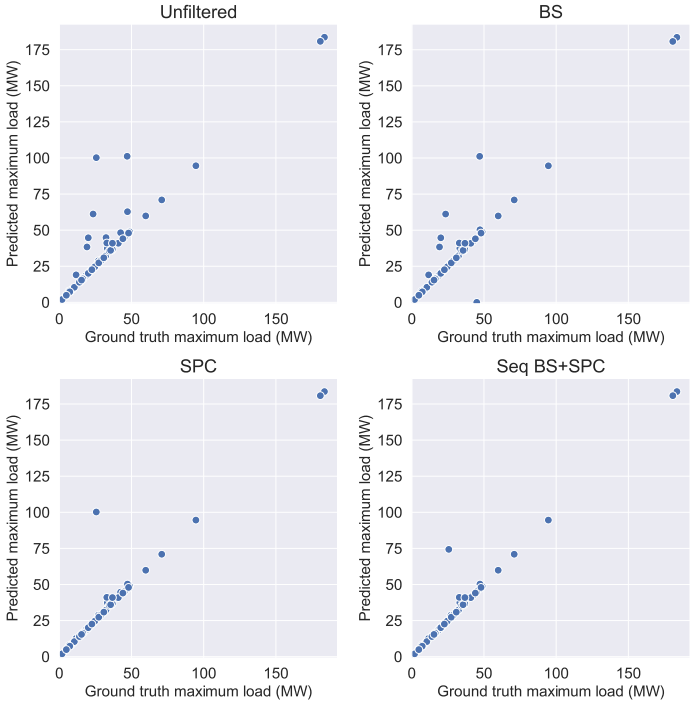


Figure 15: Scatter plots where the ground truth maximum load estimate (in kW on x-axis) is plotted against the predicted maximum load estimate (in kW on y-axis). From top-left to bottom-right are shown the estimates resulting from: no filtering, the best Binary Segmentation (BS) model, the best Statistical Process Control (SPC) model, and the best Sequential Binary Segmentation + Statistical Process Control ensemble. When a point is above the $y = x$ line, too few points are filtered out, when the point is below, too many points are filtered out.

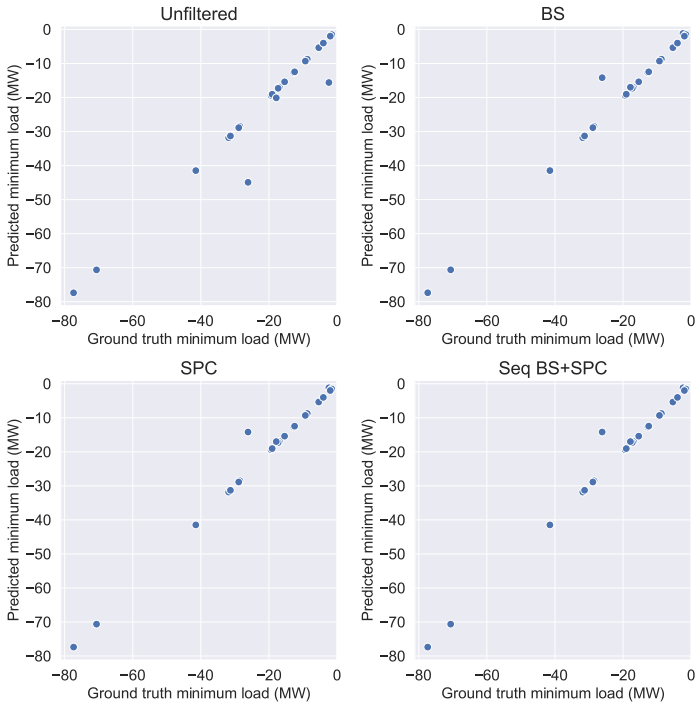


Figure 16: Scatter plots where the ground truth minimum load estimate (in kW on x-axis) is plotted against the predicted minimum load estimate (in kW on y-axis). From top-left to bottom-right the figure shows the estimates resulting from: no filtering, the best Binary Segmentation (BS) model, the best Statistical Process Control (SPC) model, and the best Sequential Binary Segmentation + Statistical Process Control ensemble. Minimum load estimates are only shown for those stations that have measurements with a negative sign (23/60). When a point is above the $y = x$ line, too many points are filtered out, when the point is below, too few points are filtered out.

future planning and operations management. Earlier variations of the presented methodology have been in use since 2021, fully replacing the manual, time-consuming, process. In addition to replacing time-consuming manual labor of domain experts whose expertise is better utilized elsewhere, our methodology allows for easier updating of load estimates, which can now be generated on-the-fly. We hope that by open sourcing our methodology, code, and data other distribution system operators can benefit similarly.

4.4 DISCUSSION

This chapter showcases a novel combination of change point detection and anomaly detection algorithms for acquiring better load estimates. While the considered algorithms demonstrate good performance, we envision several possible improvements of the proposed method in the future

Firstly, in some cases, just looking at the difference between load measurement and bottom-up load is not enough to determine the ground truth. Additional information or metadata is needed for accurate segmentation, for example one could use a separate detector for seeing whether the load or bottom-up is incorrect, and incorporating this as a rule-based detector. We estimate that we could improve performance by using additional rule-based detectors on top of our current method.

Secondly, the datasets have a somewhat skewed distribution of events, meaning not all categories are equally represented in each dataset, as can be glanced from Table 6. Due to this imbalance, as well as the general heterogeneity of events, we observe that the datasets are not as indistinct as they ideally would be. By labeling more stations we might be able to alleviate this problem. As labeling data is costly, the gains of this procedure might however not be worth the investment. Specifically over 500 primary substations are measured, which is unfeasible to manually label. Furthermore, events and anomalies are rare, and only a finite number of these occurs during any given year, which means that data acquisition and labeling for several more years might be needed to get fully balanced data.

Our current method relies heavily on the bottom-up loads which are used to attain the difference vector δ. The accuracy of the bottom-up load varies between stations, leading to a high variability between stations in terms of variance. We correct for this using the robust scaling procedure, but it remains a critical step in the total analysis pipeline. When the bottom-up load is not available, for example due to anomalous situations in the grid, or when wanting to apply this method on old historical data, our methodology cannot be directly applied.

The current analysis considers one year of measurements for one station at a time. This is in line with the labeling procedure used by the grid operator prior to this research, where the data was labeled by hand during the first month of the year. As load estimates are used in long-term decision making, generally the speed of acquisition is not that important. However, should the availability of more recent load estimates be prioritized, the binary segmentation algorithm, which only works on static data, will need to be replaced by an online change point detection algorithm.

The methods employed for the change point and anomaly detection are all highly interpretable and fairly easily comprehensible. In recent years, machine learning has been widely applied in high-performance applications on time series. Examples are LSTM's for anomaly detection [20], and Meta's (formerly Facebook's) Prophet [34] for simultaneous anomaly and change point detection. While promising, these methods would not be readily usable on this application. These methods require far more data than the unsupervised methods used in this research, something which is unfeasible due to time of data acquisition. While these methods can be trained on the unlabeled time series, we still need labels for hyperparameter optimization. Using the model with the best predictions doesn't guarantee the best model for event detection, as an overparametrized model might learn to predict anomalies. Furthermore it has been shown that in many cases simple solutions work better than complex ones for time series anomaly detection [45].

We show that our automatic filtering and load estimate procedure works in most cases, but there are still some failures cases. Future work could include studying these failure cases and devising detection methods for this purpose. Increasing the amount of data is expected to further increase performance. Furthermore, incorporating information of previous yearly measurements might lead to additional robustness of the filtering procedure. For example, a rule-based system could detect large differences in load estimates between years on top of the presented models, while still allowing for the full interpretability.

It should be noted that while figures like Figure 14 can be interpreted fairly easily, it is not immediately clear why the thresholds are chosen like they are. One could change the thresholds for this figure and achieve a better filtering. When interpreting these figures, one should always note that the thresholds are optimized to find the best F1.5 score over all stations, rather than the single figure shown.

4.5 CONCLUSION

Based on our research we present interpretable methodology for the automatic filtering of anomalies and switch events from load measurements in order to establish more accurate load estimates.

We posit that using unsupervised methods, with supervised optimization of hyperparameters and the threshold parameters, based on the F1.5 score, allows for robust, well-generalizing, performance on unseen data.

We show that without filtering, a lot of grid capacity is left unused. In our experiments on unseen test data, comprised of 60 individual station measurements, we only observe a single severe failure case in both the automatic minimum and maximum load estimate predictions. Of all estimated predictions, approximately 90% fall within a 10% error margin.

By having compared different methods and ensembling strategies we find that a combination of the well-known binary segmentation algorithm for change point detection and the tried statistical process control method for anomaly detection works best. The best ensembling strategy is a sequential ensemble, where the anomaly detector is applied after first segmenting the time series based on the obtained change points. The proposed methods are highly interpretable, a distinct advantage when this methodology is used in critical infrastructure planning. This high interpretability is a direct result of each underlying model being interpretable. SPC and BS are both simple, yet effective mathematical models. The strategies used for optimization are similarly straightforward, and can be visualized. Depending on underlying needs or when business priorities change the chosen threshold(s) for the algorithms can be adjusted based on the easily translatable precision and recall measures using figures such as Figure 12.

We finally identify possible steps for further improvement of the presented methodology. Incorporating additional data, either for further optimization, or as a historical reference, are potential avenues for improvement. Furthermore, the currently identified failure modes might be caught by using interpretable rule-based classification without losing the initial performance of the current algorithms.

ACKNOWLEDGMENTS AND DISCLOSURE OF FUNDING

The research reported in this chapter has been partly funded by the NWO grant NWA.1160.18.238 (PrimaVera); as well as BMK, BMDW, and the State of Upper Austria in the frame of the SCCH competence center INTEGRATE [(FFG grant no. 892418)] part of the FFG COMET Competence Centers for Excellent Technologies Programme; the Alliander Research Centre for Digital Technologies; and finally

by a Radboud Interdisciplinary Research Platform Green IT voucher. We also want to thank Ayan Wasame for her efforts in studying label quality, and Evander van Wolfswinkel, Gijs van Paridon and Jari Immerzeel for their efforts in (re)labeling the data.

REFERENCES

[1] S. Asefi, M. Mitrovic, D. Ćetenović, V. Levi, E. Gryazina, and V. Terzija. "Anomaly detection and classification in power system state estimation: Combining model-based and data-driven methods." In: *Sustainable Energy, Grids and Networks* 35 (2023), p. 101116.

[2] J. Bai. "Estimating multiple breaks one at a time." In: *Econometric Theory* 13.3 (1997), pp. 315–352.

[3] J. N. Betge, B. Droste, J. Heres, and S. H. Tindemans. "Efficient Assessment of Electricity Distribution Network Adequacy with the Cross-Entropy Method." In: *2021 IEEE Madrid PowerTech*. IEEE. 2021, pp. 1–6.

[4] R. Bouman, Z. Bukhsh, and T. Heskes. "Unsupervised Anomaly Detection Algorithms on Real-world Data: How Many Do We Need?" In: *Journal of Machine Learning Research* 25.105 (2024), pp. 1–34. URL: http://jmlr.org/papers/v25/23-0570.html.

[5] A. S. Brouwer, T. Kuramochi, M. van den Broek, and A. Faaij. "Fulfilling the electricity demand of electric vehicles in the long term future: An evaluation of centralized and decentralized power supply systems." In: *Applied Energy* 107 (2013), pp. 33–51.

[6] D. Chen and D. Irwin. "Sundance: Black-box behind-the-meter solar disaggregation." In: *Proceedings of the Eighth International Conference on Future Energy Systems*. 2017, pp. 45–55.

[7] T. Chen and C. Guestrin. "XGBoost: A scalable tree boosting system." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 785–794.

[8] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[9] A. Ermshaus, P. Schäfer, and U. Leser. "ClaSP: Parameter-free time series segmentation." In: *Data Mining and Knowledge Discovery* (2023).

[10] S. Á. Funke, F. Sprei, T. Gnann, and P. Plötz. "How much charging infrastructure do electric vehicles need? A review of the evidence and international comparison." In: *Transportation research part D: transport and environment* 77 (2019), pp. 224–242.

[11] C. R. Harris et al. "Array programming with NumPy." In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[12] M. Hekkenberg, R. Benders, H. Moll, and A. S. Uiterkamp. "Indications for a changing electricity demand pattern: The temperature dependence of electricity demand in the Netherlands." In: *Energy Policy* 37.4 (2009), pp. 1542–1551.

[13] J. Heres, M. van Braak, W. de Swart, and E. Gerritse. "Creating bottom up load profiles using disaggregation, clustering and supervised machine learning on large smart meter dataset." In: *27th International Conference on Electricity Distribution (CIRED 2023)*. Vol. 2023. IET. 2023, pp. 3503–3507.

[14] S. Heslop, I. MacGill, and J. Fletcher. "Maximum PV generation estimation method for residential low voltage feeders." In: *Sustainable Energy, Grids and Networks* 7 (2016), pp. 58–69.

[15] J. D. Hunter. "Matplotlib: A 2D graphics environment." In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[16] E. C. Kara, C. M. Roberts, M. Tabone, L. Alvarez, D. S. Callaway, and E. M. Stewart. "Disaggregating solar generation from feeder-level measurements." In: *Sustainable Energy, Grids and Networks* 13 (2018), pp. 112–121.

[17] A. Kieft, R. Harmsen, and M. P. Hekkert. "Heat pumps in the existing Dutch housing stock: An assessment of its Technological Innovation System." In: *Sustainable Energy Technologies and Assessments* 44 (2021), p. 101064.

[18] E. Laes, L. Gorissen, and F. Nevens. "A comparison of energy transition governance in Germany, the Netherlands and the United Kingdom." In: *Sustainability* 6.3 (2014), pp. 1129–1152.

[19] A. Langevin, M. Cheriet, and G. Gagnon. "Efficient deep generative model for short-term household load forecasting using non-intrusive load monitoring." In: *Sustainable Energy, Grids and Networks* 34 (2023), p. 101006.

[20] B. Lindemann, B. Maschler, N. Sahlab, and M. Weyrich. "A survey on anomaly detection for technical systems using LSTM networks." In: *Computers in Industry* 131 (2021), p. 103498.

[21] F. T. Liu, K. M. Ting, and Z.-H. Zhou. "Isolation forest." In: *Eighth IEEE International Conference on Data Mining*. IEEE. 2008, pp. 413–422.

[22] S. Lloyd. "Least squares quantization in PCM." In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137.

[23] J. MacQueen et al. "Some methods for classification and analysis of multivariate observations." In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. Oakland, CA, USA. 1967, pp. 281–297.

[24] M. A. Mendes, M. H. M. Paiva, and O.-e. E. Batista. "Signal processing on graphs for estimating load current variability in feeders with high integration of distributed generation." In: *Sustainable Energy, Grids and Networks* 34 (2023), p. 101032.

[25] R. Moss, E. Tzimas, H. Kara, P. Willis, and J.-k. Kooroshy. *Critical metals in strategic energy technologies. Assessing rare metals as supply-chain bottlenecks in low-carbon energy technologies*. Tech. rep. Institute for Energy and Transport IET, 2011.

[26] A. Neagoe, E. Tică, F. Popa, and B. Popa. "Change point detection in recent hydropower generation in Romania." In: *IOP Conference Series: Materials Science and Engineering*. Vol. 1032. IOP Publishing. 2021, p. 012045.

[27] J. S. Oakland. *Statistical process control*. Routledge, 2007.

[28] G. A. Pagani and M. Aiello. "Towards decentralization: A topological investigation of the medium and low voltage grids." In: *IEEE Transactions on Smart Grid* 2.3 (2011), pp. 538–547.

[29] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[30] A. Rajabi, M. Eskandari, M. J. Ghadi, L. Li, J. Zhang, and P. Siano. "A comparative study of clustering techniques for electrical load pattern segmentation." In: *Renewable and Sustainable Energy Reviews* 120 (2020), p. 109628.

[31] P. Schäfer, A. Ermshaus, and U. Leser. "ClaSP - Time Series Segmentation." In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. CIKM '21. Virtual Event, Queensland, Australia: Association for Computing Machinery, 2021, pp. 1578–1587. ISBN: 9781450384469. DOI: 10.1145/3459637.3482240. URL: https://doi.org/10.1145/3459637.3482240.

[32] A. J. Scott and M. Knott. "A cluster analysis method for grouping means in the analysis of variance." In: *Biometrics* (1974), pp. 507–512.

[33] M. Sufyan, N. A. Rahim, M. M. Aman, C. K. Tan, and S. R. S. Raihan. "Sizing and applications of battery energy storage technologies in smart grid system: A review." In: *Journal of Renewable and Sustainable Energy* 11.1 (2019).

[34] S. J. Taylor and B. Letham. "Forecasting at scale." In: *The American Statistician* 72.1 (2018), pp. 37–45.

[35] The Pandas development team. *pandas-dev/pandas: Pandas*. Version v2.1.3. Nov. 2023. DOI: 10.5281/zenodo.10107975. URL: https://doi.org/10.5281/zenodo.10107975.

[36] S. R. Thomas, V. Kurupath, and U. Nair. "A passive islanding detection method based on K-means clustering and EMD of reactive power signal." In: *Sustainable Energy, Grids and Networks* 23 (2020), p. 100377.

[37] C. Truong, L. Oudre, and N. Vayatis. "Ruptures: Change point detection in Python." In: *arXiv preprint arXiv:1801.00826* (2018).

[38] United Nations. *Theme Report on Energy Transition—Towards the Achievement of SDG 7 and Net-Zero Emissions*. 2021.

[39] G. J. Van den Burg and C. K. Williams. "An evaluation of change point detection algorithms." In: *arXiv preprint arXiv:2003.06222* (2020).

[40] L. Wang, Z. Qin, T. Slangen, P. Bauer, and T. Van Wijk. "Grid impact of electric vehicle fast charging stations: Trends, standards, issues and mitigation measures-an overview." In: *IEEE Open Journal of Power Electronics* 2 (2021), pp. 56–74.

[41] X. Wang and S.-H. Ahn. "Real-time prediction and anomaly detection of electrical load in a residential community." In: *Applied Energy* 259 (2020), p. 114145.

[42] X. Wang, Z. Yao, and M. Papaefthymiou. "A real-time electrical load forecasting and unsupervised anomaly detection framework." In: *Applied Energy* 330 (2023), p. 120279.

[43] M. L. Waskom. "Seaborn: Statistical data visualization." In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

[44] M. van Werven, J. de Joode, and M. Scheepers. *To an optimal electricity supply system*. Tech. rep. ECN, Tech. Rep. ECN-C–06-005, 2006.

[45] R. Wu and E. J. Keogh. "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress." In: *IEEE Transactions on Knowledge and Data Engineering* 35.3 (2021), pp. 2421–2429.

[46] J. E. Zhang, D. Wu, and B. Boulet. "Time series anomaly detection for smart grids: A survey." In: *2021 IEEE Electrical Power and Energy Conference (EPEC)*. IEEE. 2021, pp. 125–130.

[47] R. Zuijderduin, O. Chevtchenko, J. Smit, G. Aanhaanen, I. Melnik, and A. Geschiere. "Integration of HTS cables in the future grid of the Netherlands." In: *Physics Procedia* 36 (2012), pp. 890–893.

5

DISCUSSION AND OUTLOOK

In this thesis we discussed some of the challenges and applications of anomaly detection. Many detailed topic specific points of discussion were elaborated upon in Chapter 2, 3, and 4. In this section we instead take more of bird's-eye view on the field of anomaly detection and what we identify as important future directions to advance the field.

5.1 REGARDING THE EVALUATION OF NEW UNSUPERVISED ALGORITHMS

In Chapter 2 we perform the largest comparison of unsupervised anomaly detection algorithms. While benchmark studies like this provide useful insights at the time of writing, they are insufficiently robust for future developments. To put it more simply: it is easy to cheat on existing benchmarks, and this is doubly true for unsupervised anomaly detection. This can be problematic, as it is hard to assess which methods are pushing the state-of-the-art, and which are only muddying the waters. Systematic progress of the field can be severely hampered by a lack of good evaluation.

In supervised tasks we typically have benchmark datasets, take, for example, MNIST or CIFAR, which are already split in a train and test set. Typically, authors of new methods report their performance and open source their code, both for training and testing. Sometimes performance can be reported higher than would be realistic, for example by cherry picking the model with the best performance on the test set. With available code, this is however decently verifiable.

In unsupervised learning this becomes much harder. There are no distinct train of test steps, there is just a single evaluation round. In this evaluation, we can either average across a sensible range of hyperparameters, as we have done in Chapter 2, or by evaluating the out-of-the-box hyperparameter settings. If the benchmark datasets are available however by a developer of algorithms, the range of hyperparameters, or the out-of-the-box hyperparameters, can be influenced by the performance of the algorithm on the benchmark. Either knowingly or unknowingly, the benchmark can be overfitted on when new methods are developed after the benchmark.

Solving this problem completely is not trivial, and of course even a more robust evaluation system might still be abused. As an outlook to the future we want to offer two ways in which the evaluation problem might at least be alleviated:

5.1.1 *Constructing Benchmarks with Distinct Datasets Belonging to Train and Test Categories*

One of the most common assumptions of unsupervised and semi-supervised anomaly detection is that anomalies are heterogeneous and rare. As such, they cannot typically be explicitly modeled or learned. Because of this, we cannot generally divide a dataset in a train and test partition, unless we have a sufficiently large dataset with enough anomalies. In practical use cases, acquiring even a subset of labeled data might even be possible only after initial anomaly detection. Optimizing hyperparameters on a train set will likely improve performance [3] for some algorithms, but is likely insufficient. The best thing we can strive for is finding a good set of out-of-the-box hyperparameters that perform decently enough across a variety of datasets.

Finding this good set of hyperparameters can likely be done by having a large collection of datasets, and constructing a train and test set by splitting across the datasets rather than across the samples of each dataset. This approach has not been adopted widely, and instead many researchers opt to show "oracle" performance by optimizing on the entire dataset. A rare example of optimizing on different datasets to acquire hyperparameter settings for a test dataset is found in the work of Yoon, Noh, and Park [4]. This is however research in the computer vision domain. Because of the heterogeneity of especially tabular data, we need a large collection of multiple datasets, rather than a few large datasets. This would still allow us to construct a sufficiently representative benchmark.

In Chapter 2 we were able to barely identify the cluster of "local" anomalies from the total collection. These types of anomalies seem rarer in practice. Moreover, some properties of anomalies could not even be identified in the collection we have assembled. If we want to really go towards finding estimates of good out-of-the-box hyperparameters, we will need many more datasets.

5.1.2 *Regular Contests for Anomaly Detection*

Machine learning contests are regularly held across a variety of fields. Competitions continuously push the state of the art. For anomaly detection, barely any competitions have been held. One of the most recent prominent competitions in the field has been the VAND2.0 challenge at CVPR 2024 as part of the "VAND 2.0: Visual Anomaly and Novelty Detection" workshop. Yet, even this challenge was held with the long-existing MVTec AD [1] and MVTec LOCO AD [2] datasets rather than new datasets.

The establishment of new contests, focusing on various parts of anomaly detection, might be a valuable force in driving advancements in the field.

5.2 WHAT DO WE ACTUALLY KNOW ABOUT METHODS?

In Chapter 3 we aim to study the way autoencoders work in detail. One of the more remarkable findings of the literature review accompanying this study was how pervasive the prevailing theory - that autoencoders do not reconstruct anomalies well - was. Even though several authors have remarked upon this through the years, this assumptions kept being propagated. Of all the proposed theories of why failure can then happen, only a few substantiated their theories with experiments. It seems that as a field, we are much more busy with engineering, rather than science. Falsifying or proving theories seems to be second to just acquiring higher performance on some test set. This can be considered reminiscent of the infamous talk of Ali Rahimi at NeurIPS 2017[1], where he famously called machine learning to now be "alchemy". With the recent advances in, and expectations of, reproducibility we are already progressing along the right lines. Yet, to really move beyond the "alchemy" of it all, we need to spend more time scientifically studying machine learning, not just the methods, but really every part of machine learning.

REFERENCES

[1] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger. "The MVTec anomaly detection dataset: a comprehensive real-world dataset for unsupervised anomaly detection." In: *International Journal of Computer Vision* 129.4 (2021), pp. 1038–1059.

[2] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger. "Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization." In: *International Journal of Computer Vision* 130.4 (2022), pp. 947–969.

[3] J. Soenen, E. Van Wolputte, L. Perini, V. Vercruyssen, W. Meert, J. Davis, and H. Blockeel. "The effect of hyperparameter tuning on the Comparative evaluation of unsupervised anomaly detection methods." In: *Knowledge Discovery and Data Mining Workshop on Outlier Detection and Description*. 2021, pp. 1–9.

[4] S. Yoon, Y.-K. Noh, and F. Park. "Autoencoding under normalization constraints." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12087–12097.

1 https://www.youtube.com/watch?v=x7psGHgatGM

APPENDIX TO: UNSUPERVISED ANOMALY DETECTION ALGORITHMS ON REAL-WORLD DATA: HOW MANY DO WE NEED?

AUC SCORES FOR EACH ALGORITHM-DATA SET COMBINATION

Table 7 contains the first half of the combinations and Table 8 contains the second half of the combinations.

NEMENYI POST-HOC ANALYSIS RESULTS

Table 9 shows the results for all data sets, Table 10 shows the results for the local data sets, and Table 11 shows the results for the global data sets.

	ionosphere	yeast	spambase	hrss	glass	seismic-bumps	donors	pageblocks	internetads	wilt	breastw	shuttle	pen-global	skin	satimage-2	fault	pima	mammography	smtp	hepatitis	mi-f	cover	parkinson	pen-local	arrhythmia
MCD	0.95	0.61	0.49	0.59	0.75	0.73	0.91	0.93	0.75	0.86	0.99	1.00	0.91	0.82	0.99	0.51	0.68	0.81	0.95	0.79	0.38	0.84	0.62	0.74	0.72
OCSVM	0.84	0.58	0.52	0.57	0.65	0.71	0.87	0.93	0.69	0.44	0.98	0.98	0.87	0.55	0.99	0.47	0.63	0.83	0.95	0.76	0.49	0.93	0.45	0.47	0.80
ODIN	0.85	0.55	0.53	0.57	0.64	0.55	0.28	0.58	0.57	0.66	0.51	0.50	0.66	0.33	0.55	0.59	0.56	0.44	0.30	0.67	0.51	0.52	0.43	0.88	0.63
SO-GAAL	0.78	0.58	0.44	0.57	0.77	0.57	0.60	0.88	0.45	0.44	0.48	0.58	0.76	0.66	0.72	0.39	0.33	0.61	0.66	0.72	0.58	0.60	0.30	0.05	0.70
LODA	0.47	0.48	0.59	0.58	0.22	0.71	0.74	0.57	0.60	0.49	0.92	0.67	0.72	0.54	0.98	0.43	0.58	0.81	0.76	0.55	0.80	0.91	0.76	0.64	0.51
DynamicHBOS	0.21	0.58	0.70	0.57	0.60	0.75	0.90	0.71	0.50	0.45	0.98	0.99	0.71	0.61	0.98	0.55	0.63	0.86	0.95	0.78	0.34	0.79	0.68	0.66	0.86
COPOD	0.80	0.62	0.68	0.60	0.64	0.71	0.82	0.88	0.68	0.34	0.99	0.99	0.79	0.47	0.97	0.46	0.65	0.91	0.91	0.80	0.66	0.88	0.54	0.52	0.80
kNN	0.92	0.60	0.65	0.55	0.82	0.74	0.67	0.95	0.73	0.69	0.98	0.82	0.98	0.67	0.95	0.66	0.71	0.83	0.91	0.79	0.39	0.79	0.64	0.98	0.80
kth-NN	0.89	0.60	0.67	0.55	0.79	0.73	0.67	0.95	0.71	0.69	0.98	0.84	0.98	0.67	0.97	0.64	0.71	0.83	0.92	0.81	0.39	0.79	0.60	0.98	0.80
PCA	0.80	0.59	0.55	0.57	0.53	0.64	0.77	0.90	0.61	0.31	0.82	0.99	0.78	0.44	0.97	0.50	0.59	0.84	0.87	0.76	0.83	0.93	0.31	0.34	0.78
gen2out	0.76	0.60	0.66	0.57	0.73	0.71	0.86	0.91	0.56	0.49	0.99	0.99	0.92	0.65	0.99	0.53	0.65	0.86	0.94	0.70	0.70	0.92	0.53	0.66	0.78
GMM	0.74	0.57	0.58	0.60	0.76	0.66	0.75	0.83	0.61	0.78	0.96	0.90	0.92	0.74	0.75	0.65	0.59	0.85	0.94	0.35	0.45	0.90	0.61	0.95	0.34
DeepSVDD	0.75	0.50	0.47	0.55	0.48	0.59	0.52	0.67	0.68	0.50	0.71	0.49	0.64	0.50	0.75	0.50	0.53	0.53	0.78	0.65	0.41	0.47	0.52	0.72	0.67
SOD	0.89	0.53	0.54	0.53	0.74	0.71	0.77	0.76	0.53	0.59	0.94	0.74	0.81	0.61	0.78	0.64	0.58	0.78	0.88	0.57	0.37	0.63	0.70	0.92	0.76
KDE	0.93	0.61	0.60	0.55	0.77	0.74	0.88	0.94	0.66	0.50	0.98	0.93	0.96	0.62	0.99	0.63	0.70	0.83	0.95	0.78	0.36	0.92	0.62	0.87	0.67
CBLOF	0.72	0.50	0.59	0.54	0.52	0.45	0.43	0.63	0.56	0.57	0.29	0.98	0.58	0.49	0.33	0.59	0.42	0.63	0.40	0.36	0.56	0.83	0.45	0.67	0.48
INNE	0.90	0.60	0.59	0.56	0.70	0.71	0.79	0.96	0.69	0.56	0.77	0.98	0.89	0.53	1.00	0.53	0.67	0.72	0.94	0.71	0.52	0.95	0.47	0.74	0.73
AE	0.85	0.60	0.55	0.58	0.60	0.67	0.76	0.92	0.61	0.34	0.97	0.99	0.90	0.60	0.98	0.53	0.66	0.88	0.82	0.72	0.83	0.92	0.35	0.55	0.77
LOF	0.90	0.54	0.50	0.56	0.79	0.54	0.57	0.67	0.63	0.65	0.40	0.58	0.72	0.57	0.56	0.57	0.61	0.74	0.48	0.72	0.55	0.51	0.54	0.98	0.62
IF	0.86	0.61	0.62	0.59	0.69	0.67	0.77	0.90	0.68	0.45	0.99	1.00	0.93	0.67	0.99	0.58	0.68	0.86	0.91	0.70	0.78	0.89	0.50	0.86	0.81
ECOD	0.74	0.56	0.64	0.59	0.62	0.68	0.89	0.91	0.68	0.39	0.99	1.00	0.77	0.49	0.97	0.47	0.59	0.91	0.88	0.74	0.56	0.93	0.38	0.45	0.81
ABOD	0.93	0.60	0.52	0.57	0.81	0.74	0.25	0.94	0.74	0.69	0.98	0.79	0.98	0.25	0.95	0.68	0.70	0.55	0.94	0.79	0.44	0.80	0.63	0.96	0.82
ensemble-LOF	0.90	0.53	0.51	0.56	0.80	0.52	0.68	0.71	0.68	0.64	0.31	0.62	0.81	0.68	0.67	0.56	0.61	0.78	0.83	0.64	0.59	0.50	0.54	0.99	0.62
ae-DeepSVDD	0.71	0.54	0.41	0.52	0.58	0.51	0.28	0.66	0.63	0.41	0.72	0.46	0.66	0.49	0.60	0.51	0.55	0.36	0.71	0.60	0.30	0.46	0.44	0.63	0.63
u-CBLOF	0.87	0.42	0.51	0.57	0.80	0.70	0.81	0.92	0.69	0.53	0.97	0.99	0.90	0.67	1.00	0.56	0.66	0.79	0.90	0.72	0.60	0.89	0.57	0.81	0.79
LUNAR	0.93	0.58	0.51	0.58	0.82	0.72	0.67	0.72	0.65	0.43	0.98	0.65	0.91	0.66	0.89	0.71	0.70	0.83	0.90	0.65	0.41	0.73	0.51	0.90	0.79
HBOS	0.77	0.59	0.64	0.56	0.71	0.71	0.81	0.75	0.68	0.36	0.99	0.99	0.77	0.60	0.98	0.59	0.70	0.83	0.83	0.78	0.40	0.64	0.52	0.73	0.86
VAE	0.86	0.60	0.56	0.57	0.59	0.67	0.79	0.92	0.61	0.33	0.96	0.79	0.85	0.55	0.87	0.55	0.66	0.84	0.82	0.72	0.51	0.94	0.42	0.65	0.77
COF	0.88	0.55	0.47	0.54	0.78	0.55	0.68	0.59	0.66	0.65	0.43	0.56	0.64	0.70	0.54	0.57	0.60	0.75	0.42	0.57	0.52	0.51	0.60	0.95	0.66
LMDD	0.74	0.53	0.50	0.50	0.59	0.71	0.90	0.76	0.68	0.40	0.66	0.99	0.75	0.43	0.48	0.40	0.64	0.78	0.86	0.83	0.56	0.91	0.59	0.45	0.86
ALAD	0.57	0.51	0.56	0.52	0.55	0.64	0.60	0.58	0.61	0.43	0.83	0.58	0.62	0.72	0.62	0.41	0.54	0.72	0.31	0.66	0.74	0.42	0.45	0.33	0.67
beta-VAE	0.79	0.60	0.55	0.57	0.59	0.66	0.82	0.91	0.61	0.33	0.95	0.99	0.81	0.52	0.98	0.49	0.66	0.89	0.82	0.74	0.80	0.93	0.36	0.44	0.77
EIF	0.88	0.61	0.67	0.58	0.70	0.71	0.83	0.91	0.69	0.49	0.98	1.00	0.94	0.71	1.00	0.52	0.69	0.84	0.94	0.75	0.79	0.89	0.52	0.79	0.82

Table 7: The AUC values for the first half of the algorithm-data set combinations.

	musk	wbc2	vertebral	musit	pendigits	stamps	nasa	wbc	optdigits	waveform	landsat	cardio	satellite	yeast6	campaign	speech	letter	wine	aloi	mf-v	vowels	thyroid	annthyroid	http	magic.gamma
MCD	1.00	0.99	0.39	0.83	0.84	0.84	0.61	0.93	0.39	0.59	0.56	0.82	0.76	0.67	0.77	0.50	0.81	0.77	0.54	0.58	0.91	0.99	0.93	1.00	0.74
OCSVM	0.87	0.98	0.38	0.70	0.93	0.88	0.52	0.94	0.51	0.65	0.38	0.88	0.63	0.72	0.80	0.47	0.61	0.86	0.54	0.62	0.71	0.99	0.94	0.99	0.67
ODIN	0.55	0.79	0.50	0.62	0.51	0.59	0.56	0.79	0.51	0.66	0.49	0.57	0.49	0.45	0.56	0.65	0.86	0.66	0.74	0.53	0.87	0.57	0.50	0.92	0.65
SO-GAAL	0.90	0.83	0.67	0.58	0.90	0.71	0.51	0.41	0.50	0.41	0.46	0.80	0.55	0.56	0.56	0.47	0.43	0.20	0.50	0.53	0.04	0.93	0.86	0.66	0.57
LODA	0.99	0.98	0.32	0.82	0.91	0.92	0.57	0.96	0.42	0.70	0.39	0.64	0.62	0.71	0.40	0.55	0.55	0.32	0.53	0.72	0.72	0.93	0.55	0.97	0.69
DynamicHBOS	1.00	0.99	0.27	0.44	0.88	0.89	0.65	0.95	0.83	0.69	0.59	0.84	0.77	0.67	0.83	0.47	0.63	0.92	0.47	0.51	0.71	0.99	0.89	0.94	0.72
COPOD	0.95	0.99	0.33	0.77	0.90	0.93	0.54	0.96	0.68	0.73	0.42	0.92	0.63	0.81	0.78	0.49	0.56	0.87	0.51	0.67	0.50	0.94	0.78	0.99	0.68
kNN	0.54	0.99	0.35	0.81	0.79	0.86	0.65	0.94	0.45	0.74	0.60	0.86	0.70	0.64	0.80	0.51	0.85	0.81	0.60	0.59	0.97	0.99	0.92	0.15	0.79
kth-NN	0.63	0.99	0.34	0.82	0.82	0.89	0.65	0.94	0.48	0.74	0.61	0.84	0.71	0.67	0.81	0.48	0.81	0.90	0.58	0.60	0.96	0.99	0.92	0.15	0.78
PCA	1.00	0.99	0.49	0.85	0.92	0.85	0.46	0.92	0.48	0.60	0.40	0.95	0.63	0.71	0.74	0.47	0.52	0.81	0.55	0.74	0.64	0.96	0.69	1.00	0.65
gamaut	1.00	1.00	0.39	0.78	0.95	0.90	0.53	0.94	0.63	0.63	0.48	0.93	0.72	0.74	0.70	0.46	0.65	0.78	0.50	0.71	0.71	0.99	0.87	1.00	0.70
GMM	0.76	0.39	0.34	0.67	0.76	0.74	0.62	0.38	0.45	0.69	0.59	0.64	0.67	0.49	0.77	0.57	0.89	0.17	0.53	0.60	0.95	0.91	0.87	0.20	0.86
DeepSVDD	0.73	0.92	0.50	0.63	0.58	0.69	0.46	0.85	0.55	0.45	0.60	0.73	0.59	0.65	0.65	0.51	0.63	0.72	0.50	0.48	0.64	0.70	0.63	0.41	0.48
SOD	0.45	0.96	0.59	0.71	0.66	0.74	0.59	0.93	0.52	0.62	0.57	0.68	0.58	0.59	0.73	0.56	0.89	0.47	0.66	0.57	0.93	0.96	0.81	0.26	0.75
KDE	0.08	0.97	0.33	0.73	0.96	0.87	0.64	0.90	0.41	0.76	0.60	0.81	0.78	0.72	0.84	0.43	0.88	0.77	0.59	0.59	0.89	0.98	0.94	0.99	0.68
CBLOF	0.61	0.23	0.53	0.64	0.45	0.32	0.45	0.35	0.48	0.69	0.55	0.53	0.62	0.39	0.51	0.52	0.59	0.31	0.54	0.55	0.81	0.24	0.37	0.42	0.49
INNE	0.99	0.91	0.40	0.82	0.87	0.82	0.59	0.93	0.55	0.74	0.54	0.89	0.75	0.69	0.81	0.47	0.68	0.82	0.53	0.64	0.89	0.98	0.92	1.00	0.71
AE	1.00	0.98	0.36	0.85	0.93	0.88	0.50	0.93	0.51	0.64	0.37	0.93	0.60	0.67	0.73	0.47	0.55	0.80	0.55	0.74	0.76	0.91	0.65	1.00	0.69
LOF	0.54	0.75	0.47	0.60	0.51	0.64	0.56	0.92	0.49	0.72	0.54	0.53	0.54	0.48	0.55	0.53	0.87	0.76	0.74	0.55	0.93	0.59	0.48	0.37	0.69
IF	1.00	1.00	0.36	0.81	0.95	0.90	0.57	0.94	0.72	0.72	0.48	0.93	0.70	0.73	0.72	0.47	0.64	0.80	0.54	0.77	0.77	0.98	0.82	1.00	0.73
ECOD	0.96	0.99	0.42	0.75	0.91	0.88	0.44	0.90	0.60	0.72	0.37	0.94	0.75	0.70	0.77	0.49	0.57	0.73	0.53	0.65	0.59	0.98	0.79	0.98	0.64
ABOD	0.18	0.99	0.36	0.81	0.77	0.85	0.63	0.93	0.46	0.70	0.58	0.76	0.66	0.61	0.79	0.57	0.82	0.76	0.61	0.59	0.96	0.98	0.91	0.97	0.86
ensemble-LOF	0.63	0.92	0.45	0.63	0.53	0.70	0.55	0.94	0.51	0.72	0.55	0.59	0.58	0.48	0.46	0.55	0.87	0.88	0.75	0.59	0.94	0.71	0.50	0.14	0.70
sb-DeepSVDD	0.64	0.86	0.43	0.60	0.47	0.67	0.52	0.77	0.47	0.47	0.54	0.76	0.52	0.57	0.61	0.50	0.53	0.60	0.51	0.37	0.58	0.76	0.61	0.43	0.45
u-CBLOF	0.85	0.99	0.42	0.82	0.91	0.78	0.44	0.93	0.52	0.71	0.57	0.84	0.77	0.62	0.80	0.47	0.70	0.59	0.54	0.59	0.86	0.99	0.91	1.00	0.70
LUNAR	0.73	0.97	0.36	0.76	0.72	0.71	0.54	0.93	0.44	0.74	0.59	0.60	0.66	0.61	0.67	0.46	0.75	0.69	0.71	0.62	0.86	0.93	0.71	0.16	0.81
HBOS	1.00	0.99	0.36	0.35	0.93	0.91	0.49	0.96	0.87	0.68	0.58	0.80	0.76	0.75	0.78	0.46	0.60	0.91	0.50	0.59	0.66	0.97	0.68	0.97	0.71
VAE	0.80	0.97	0.38	0.84	0.92	0.88	0.54	0.87	0.47	0.66	0.51	0.87	0.63	0.77	0.72	0.47	0.63	0.77	0.55	0.64	0.70	0.92	0.67	1.00	0.67
COF	0.50	0.61	0.47	0.61	0.51	0.52	0.54	0.81	0.43	0.71	0.53	0.50	0.53	0.41	0.50	0.54	0.86	0.41	0.77	0.54	0.90	0.51	0.47	0.12	0.65
LMDD	0.97	1.00	0.36	0.75	0.94	0.89	0.49	0.72	0.59	0.56	0.45	0.71	0.42	0.63	0.73	0.48	0.52	0.86	0.50	0.60	0.63	0.99	0.91	1.00	0.63
ALAD	0.56	0.65	0.50	0.51	0.61	0.53	0.49	0.65	0.70	0.54	0.43	0.71	0.54	0.62	0.67	0.49	0.50	0.81	0.50	0.56	0.59	0.43	0.51	0.91	0.57
beta-VAE	0.86	0.99	0.37	0.85	0.94	0.90	0.49	0.93	0.51	0.65	0.39	0.95	0.60	0.77	0.73	0.47	0.52	0.81	0.55	0.74	0.62	0.96	0.67	1.00	0.67
EIF	1.00	1.00	0.35	0.81	0.95	0.89	0.59	0.95	0.66	0.73	0.50	0.93	0.72	0.73	0.76	0.47	0.65	0.85	0.53	0.78	0.81	0.99	0.90	0.99	0.72

Table 8: The AUC values for the second half of the algorithm-data set combinations.

	PCA	LMDD	SO-GAAL	IF	LOF	EIF	ALAD	kNN	LUNAR	KDE	COF	MCD	GMM	u-CBLOF	INNE	ECOD	beta-VAE	AE	LODA	ODIN	HBOS	SOD	ensemble-LOF	DynamicHBOS	VAE	COPOD	OCSVM	DeepSVDD	genzout	ABOD	CBLOF	kth-NN	sb-DeepSVDD
PCA	1.0	0.9	0.188	0.9	**0.012**	**0.045**	0.155	0.9	0.482	0.835	0.167	0.9	0.9	0.736	0.9	0.9	0.9	0.9	0.9	0.616	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.216	0.651	0.747	**0.035**	0.958	**0.001**
LMDD	0.9	1.0	0.581	**0.031**	0.9	**0.001**	**0.024**	0.9	0.245	**0.024**	0.9	0.125	0.9	**0.026**	0.9	0.658	0.326	0.9	0.9	0.9	0.9	0.9	0.9	0.888	0.9	0.613	0.9	0.616	0.245	0.338	0.204	**0.007**	**0.035**
SO-GAAL	0.188	0.581	1.0	**0.001**	0.9	**0.001**	0.9	**0.001**	**0.032**	**0.001**	0.9	**0.001**	**0.048**	**0.001**	**0.001**	**0.006**	**0.024**	**0.007**	0.877	0.9	**0.001**	0.238	0.553	**0.001**	0.074	**0.001**	**0.001**	0.9	**0.001**	0.9	0.9	**0.001**	0.9
IF	0.188	**0.031**	**0.001**	1.0	**0.001**	0.9	**0.001**	0.9	0.574	0.9	**0.001**	0.9	0.496	0.9	0.9	0.852	0.63	0.849	**0.006**	**0.001**	0.9	0.146	**0.036**	0.9	0.401	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
LOF	0.9	0.9	0.9	**0.001**	1.0	**0.001**	0.9	**0.001**	0.9	**0.002**	0.9	**0.001**	0.9	0.062	0.9	0.718	0.9	0.722	0.9	0.9	0.33	0.9	0.9	0.171	0.9	**0.049**	0.434	0.9	**0.007**	**0.012**	0.9	**0.001**	0.383
EIF	**0.012**	**0.001**	**0.001**	0.9	**0.001**	1.0	**0.001**	0.9	0.088	0.9	**0.001**	0.9	0.062	0.9	**0.001**	**0.001**	0.274	0.271	0.116	0.271	**0.001**	**0.001**	**0.008**	**0.001**	0.845	**0.04**	0.9	0.574	**0.001**	0.9	**0.001**	0.9	**0.001**
ALAD	**0.045**	0.245	0.9	**0.001**	0.9	**0.001**	1.0	**0.001**	**0.005**	**0.001**	0.9	**0.001**	0.442	**0.008**	**0.001**	**0.001**	**0.004**	**0.001**	0.56	0.9	**0.001**	0.062	0.221	**0.001**	**0.014**	**0.001**	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.9
kNN	0.155	**0.024**	**0.001**	0.9	**0.001**	0.9	**0.001**	1.0	0.525	0.9	**0.001**	0.9	0.442	0.9	0.9	0.9	0.9	0.9	0.581	0.799	**0.004**	**0.001**	0.9	0.119	**0.028**	0.9	0.342	0.9	**0.001**	0.9	0.9	0.9	**0.001**
LUNAR	0.9	0.9	**0.032**	0.574	0.9	0.088	**0.005**	0.525	1.0	0.863	0.451	0.542	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.221	0.9	0.9	0.9	0.9	0.9	0.9	**0.039**	0.9	0.9	**0.004**	0.284	**0.001**
KDE	0.482	0.125	**0.001**	0.9	**0.002**	0.9	**0.001**	0.9	0.863	1.0	**0.001**	0.9	0.785	0.9	0.9	0.9	0.9	**0.03**	0.9	**0.001**	0.9	0.409	0.141	0.9	0.701	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
COF	0.835	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.451	**0.001**	1.0	**0.001**	0.532	**0.003**	**0.001**	0.175	0.383	0.178	0.9	0.9	**0.032**	0.898	0.9	**0.011**	0.616	**0.002**	0.05	0.9	**0.001**	**0.001**	0.9	**0.001**	0.9
MCD	0.167	**0.026**	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.542	0.9	**0.001**	1.0	0.462	0.9	0.9	0.821	0.599	0.817	**0.005**	**0.001**	0.9	0.127	**0.031**	0.9	0.362	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
GMM	0.9	0.9	**0.048**	0.496	0.9	0.062	**0.008**	0.442	0.9	0.785	0.532	0.462	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.291	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.958	0.9	0.9	**0.006**	0.216	**0.001**
u-CBLOF	0.9	0.658	**0.001**	0.9	0.062	0.9	**0.001**	0.9	0.9	0.9	**0.003**	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.342	**0.001**	0.9	0.687	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
INNE	0.736	0.326	**0.001**	0.9	**0.011**	0.9	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	1.0	0.9	0.9	0.9	0.105	**0.001**	0.9	0.673	0.358	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
ECOD	0.9	0.9	**0.001**	0.9	0.33	0.662	**0.001**	0.9	0.9	0.9	0.175	0.821	0.9	**0.032**	0.9	1.0	0.9	0.9	0.9	0.775	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.584	**0.001**
beta-VAE	0.9	0.9	**0.024**	0.63	0.9	0.116	**0.004**	0.581	0.9	0.9	0.383	0.599	0.9	0.9	0.9	0.673	0.9	0.9	1.0	0.9	0.9	0.68	0.9	1.0	0.9	0.9	0.9	0.271	0.588	0.683	**0.048**	**0.042**	**0.002**
AE	0.9	0.9	**0.007**	0.849	0.722	0.271	**0.001**	0.799	0.9	0.9	0.178	0.817	0.9	0.9	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.641	0.9	0.588	0.271	0.371	0.185	**0.008**	**0.013**
LODA	0.9	0.9	0.877	**0.006**	0.9	**0.001**	0.56	**0.004**	0.9	0.9	**0.03**	0.9	**0.005**	0.9	0.9	0.9	0.9	0.9	1.0	0.9	**0.001**	0.775	0.9	0.9	0.295	0.863	0.9	0.071	0.11	0.51	**0.001**	0.074	
ODIN	0.616	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.221	**0.001**	0.9	**0.001**	0.291	**0.001**	**0.001**	0.068	0.178	0.069	0.9	1.0	**0.009**	0.68	0.9	**0.003**	0.383	**0.001**	**0.015**	0.9	**0.001**	**0.001**	0.9	**0.001**	0.9
HBOS	0.9	0.9	**0.001**	0.9	0.33	0.662	**0.001**	0.9	0.9	1.0	**0.001**	0.9	0.9	**0.032**	0.9	0.9	0.9	0.9	0.9	0.775	**0.009**	1.0	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
SOD	0.9	0.9	0.238	0.146	0.9	**0.008**	0.062	0.119	0.9	0.9	0.499	0.898	0.127	0.9	0.9	0.673	0.9	0.9	0.9	0.9	0.9	0.68	0.9	1.0	0.9	0.9	0.9	0.271	0.588	0.683	**0.048**	**0.042**	**0.002**
ensemble-LOF	0.9	0.9	0.553	**0.036**	0.9	**0.001**	0.221	**0.028**	0.9	0.9	0.141	0.9	**0.031**	0.9	0.9	0.358	0.9	0.9	0.9	0.9	1.0	0.9	0.9	1.0	0.9	0.9	0.641	0.9	0.588	0.271	0.371	0.185	**0.008**	**0.013**
DynamicHBOS	0.9	0.888	**0.001**	0.9	0.171	0.845	**0.001**	0.9	0.9	0.9	0.9	**0.011**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.591	**0.003**	0.9	0.9	0.9	1.0	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**
VAE	0.9	0.9	0.074	0.401	0.9	**0.04**	**0.014**	0.342	0.9	0.9	0.701	0.616	0.362	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.383	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.687	0.87	0.9	**0.01**	0.155	**0.001**
COPOD	0.9	0.613	**0.001**	0.9	**0.049**	0.9	**0.001**	0.9	0.9	0.9	0.9	**0.002**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.295	**0.001**	0.9	0.9	0.641	0.9	0.9	1.0	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
OCSVM	0.9	0.9	**0.001**	0.9	0.434	0.574	**0.001**	0.9	0.9	0.9	0.05	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.865	**0.015**	0.9	0.9	0.9	0.9	0.9	1.0	**0.001**	0.9	0.9	0.9	**0.001**	0.863	**0.001**
DeepSVDD	0.216	0.616	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	**0.039**	**0.001**	0.9	**0.001**	**0.058**	**0.001**	**0.001**	**0.008**	**0.029**	**0.008**	0.9	0.9	**0.001**	0.271	0.588	**0.001**	0.087	**0.001**	**0.001**	1.0	**0.001**	**0.001**	0.9	**0.001**	0.9	
genzout	0.651	0.245	**0.001**	0.9	**0.007**	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.071	**0.001**	0.9	0.588	0.271	0.9	0.87	0.9	0.9	**0.001**	1.0	0.9	0.9	**0.001**	
ABOD	0.747	0.338	**0.001**	0.9	**0.012**	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.11	**0.001**	0.9	0.683	0.371	0.9	0.9	0.9	0.9	0.9	1.0	**0.001**	0.9	**0.001**	
CBLOF	**0.035**	0.204	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	**0.004**	**0.001**	0.9	**0.001**	**0.006**	**0.001**	**0.001**	**0.001**	**0.003**	**0.001**	0.51	0.9	**0.001**	**0.048**	0.183	**0.001**	**0.01**	**0.001**	**0.001**	0.9	**0.001**	**0.001**	1.0	**0.001**	0.9	
kth-NN	0.958	**0.007**	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.284	0.9	**0.001**	0.9	0.216	0.9	0.9	0.954	0.342	0.581	**0.001**	0.9	**0.001**	**0.042**	**0.008**	0.9	0.155	0.9	0.863	**0.001**	0.9	0.9	**0.001**	1.0	**0.001**	
sb-DeepSVDD	**0.001**	**0.015**	0.9	**0.001**	0.383	**0.001**	0.9	**0.001**	**0.001**	**0.001**	0.9	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	0.074	0.9	**0.001**	**0.002**	**0.013**	**0.001**	**0.001**	**0.001**	0.9	**0.001**	**0.001**	0.9	**0.001**	1.0	

Table 9: The p-values from Nemenyi post-hoc analysis on all algorithm pairs based on all 52 data sets. P-values below 0.05 have been printed bold.

	PCA	LMDD	SO-GAAL	IF	LOF	EIF	ALAD	kNN	LUNAR	KDE	COF	MCD	GMM	u-CBLOF	INNE	ECOD	beta-VAE	AE	LODA	ODIN	HBOS	SOD	ensemble-LOF	DynamicHBOS	VAE	COPOD	OCSVM	DeepSVDD	genzout	ABOD	CBLOF	kth-NN	sh-DeepSVDD
PCA	1.0	0.9	0.9	0.129	**0.001**	**0.042**	0.9	**0.001**	**0.001**	**0.001**	**0.002**	**0.001**	**0.001**	**0.028**	0.064	0.9	0.9	0.9	0.9	0.9	0.407	0.804	**0.002**	**0.001**	0.78	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9
LMDD	0.9	1.0	0.9	0.436	**0.009**	0.193	0.9	**0.001**	**0.009**	**0.003**	**0.019**	**0.004**	**0.004**	0.144	0.263	0.9	0.9	0.9	0.9	0.9	0.755	0.9	**0.017**	**0.001**	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9
SO-GAAL	0.9	0.9	1.0	0.068	**0.001**	**0.019**	0.9	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.012**	0.03	0.9	0.9	0.9	0.9	0.9	0.252	0.659	**0.001**	**0.001**	0.635	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9
IF	0.129	0.436	0.068	1.0	0.9	0.9	0.144	0.263	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.463	0.497	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.755	0.9	0.695	0.335
LOF	**0.001**	**0.009**	**0.001**	0.9	1.0	0.9	**0.001**	0.9	**0.001**	**0.001**	0.9	0.9	0.9	**0.011**	**0.008**	0.144	0.392	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.526	0.31	0.241	0.129	0.538	0.9	**0.006**
EIF	**0.042**	0.193	**0.019**	0.9	0.9	1.0	**0.047**	0.526	0.9	0.9	0.9	0.9	0.9	0.212	0.175	0.731	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.864	0.707	0.9	0.888	0.9	0.136
ALAD	0.9	0.9	0.9	0.144	**0.001**	**0.047**	1.0	**0.001**	**0.001**	**0.001**	**0.003**	**0.001**	**0.001**	0.032	0.072	0.9	0.9	0.9	0.9	0.9	0.436	0.828	**0.002**	**0.001**	0.804	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9
kNN	**0.001**	**0.001**	**0.001**	0.263	0.9	0.526	**0.001**	1.0	0.9	0.9	0.9	0.9	0.9	0.598	0.336	0.9	0.9	**0.001**	**0.001**	**0.001**	0.072	0.9	0.9	0.9	**0.01**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	0.9	**0.001**	**0.001**
LUNAR	**0.001**	**0.009**	**0.001**	0.9	0.9	0.9	**0.001**	0.9	1.0	0.9	0.9	0.9	0.9	0.598	0.436	0.9	0.9	**0.008**	0.136	0.378	0.9	0.9	0.9	0.9	0.9	0.9	0.514	0.248	0.231	0.125	0.526	0.252	0.9	**0.005**
KDE	**0.001**	**0.003**	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	**0.003**	**0.003**	0.064	0.212	0.9	0.378	0.9	0.755	0.9	0.9	0.9	0.322	0.159	0.118	0.057	0.335	0.9	0.139	0.9	**0.002**
COF	**0.002**	**0.019**	**0.001**	0.9	0.9	0.9	**0.003**	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	**0.022**	**0.017**	0.231	0.526	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.647	0.45	0.363	0.212	0.659	0.9	0.392	0.9	**0.012**
MCD	**0.001**	**0.004**	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	**0.004**	**0.003**	0.076	0.241	0.9	0.792	0.9	0.9	0.9	0.816	0.363	0.184	0.136	0.668	0.378	0.9	0.152	0.9	**0.002**
GMM	**0.001**	**0.004**	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	**0.004**	**0.003**	0.076	0.241	0.9	0.792	0.9	0.9	0.9	0.816	0.363	0.184	0.136	0.668	0.378	0.9	0.152	0.9	**0.002**
u-CBLOF	**0.028**	0.144	**0.012**	0.9	0.9	0.9	**0.032**	0.598	0.9	0.9	0.9	0.9	0.9	1.0	0.9	0.159	0.129	0.699	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.864	0.653	0.9	0.816	0.9	0.1
INNE	0.064	0.263	**0.03**	0.9	0.9	0.9	0.072	0.436	0.9	0.9	0.9	0.9	0.9	1.0	0.286	0.241	0.816	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.792	0.9	0.9	0.9	0.852	0.193
ECOD	0.9	0.9	0.9	0.463	**0.011**	0.212	0.9	**0.001**	**0.01**	**0.003**	**0.022**	**0.004**	**0.004**	0.159	0.286	1.0	0.9	0.9	0.9	0.9	0.9	0.78	0.9	**0.019**	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
beta-VAE	0.9	0.9	0.9	0.497	**0.008**	0.175	0.9	**0.001**	**0.008**	**0.003**	**0.017**	**0.003**	**0.003**	0.129	0.241	0.9	1.0	0.9	0.9	0.9	0.731	0.9	**0.014**	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
AE	0.9	0.9	0.9	0.9	0.144	0.731	0.9	**0.001**	0.136	0.064	0.231	0.076	0.076	0.659	0.816	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	**0.032**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9
LODA	0.9	0.9	0.9	0.9	0.392	0.9	0.9	**0.001**	0.378	0.212	0.526	0.241	0.241	0.9	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.004**	0.9	0.9
ODIN	0.497	0.755	0.252	0.9	0.9	0.9	0.436	0.072	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.731	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.363	0.671
HBOS	0.804	0.9	0.659	0.9	0.9	0.9	0.828	**0.009**	0.9	0.755	0.9	0.792	0.792	0.9	0.9	0.9	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.622	0.9	0.9	0.9	0.081	0.9
SOD	**0.002**	**0.017**	**0.001**	0.9	0.9	0.9	**0.002**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.019**	**0.014**	0.212	0.502	0.9	0.9	1.0	0.9	0.9	0.622	0.9	0.9	0.622	0.421	0.335	0.193	0.635	0.9	0.363	0.9	**0.01**
ensemble-LOF	**0.001**	**0.001**	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	**0.032**	0.125	0.9	0.9	0.622	0.9	1.0	0.9	0.9	0.9	0.647	0.202	0.289	0.064	**0.028**	0.212	0.9	0.072	0.9	**0.001**
DynamicHBOS	0.78	0.9	0.635	0.9	0.9	0.9	0.804	**0.01**	0.9	0.78	0.9	0.816	0.816	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.647	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.118	0.9	0.089	0.9	0.9
VAE	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.514	0.322	0.647	0.365	0.365	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	1.0	0.9	0.9	0.9	0.9	0.9	**0.012**	0.9	**0.009**	0.9	0.9
COPOD	0.9	0.9	0.9	0.31	0.9	0.9	**0.001**	0.298	0.159	0.159	0.45	0.184	0.184	0.864	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.421	0.089	0.9	0.9	**0.004**	0.9	**0.003**	0.9
OCSVM	0.9	0.9	0.9	0.241	0.864	0.9	**0.001**	0.231	0.118	0.057	0.212	0.365	0.136	0.792	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	1.0	0.9	0.335	0.064	0.9	**0.002**	0.9	**0.002**	0.9
DeepSVDD	0.9	0.9	0.9	0.129	0.707	0.9	**0.001**	0.125	0.057	0.212	0.068	0.068	0.136	0.792	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	1.0	0.9	0.193	**0.028**	0.9	**0.001**	0.9	**0.001**	0.9
genzout	0.9	0.9	0.9	0.9	0.538	0.9	0.9	**0.001**	0.526	0.335	0.659	0.378	0.378	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.635	0.212	0.9	0.9	0.9	0.9	0.9
ABOD	**0.001**	**0.001**	**0.001**	0.755	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.013**	1.0	**0.003**	0.9
CBLOF	0.9	0.9	0.9	0.263	0.888	0.9	**0.001**	0.252	0.129	0.392	0.152	0.152	0.816	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.013**	0.9	**0.002**
kth-NN	**0.001**	**0.001**	**0.001**	0.695	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.002**	0.9	1.0	**0.001**
sh-DeepSVDD	0.9	0.9	0.9	0.335	**0.006**	0.136	0.9	**0.001**	**0.005**	**0.002**	**0.012**	**0.002**	**0.002**	0.1	0.193	0.9	0.9	0.9	0.9	0.9	0.671	0.9	**0.01**	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	0.9	1.0

Table 10: The p-values from Nemenyi post-hoc analysis on all algorithm pairs based on the 17 local data sets. P-values below 0.05 have been printed bold.

	PCA	LMDD	SO-GAAL	IF	LOF	EIF	ALAD	kNN	LUNAR	KDE	COF	MCD	GMM	u-CBLOF	INNE	ECOD	beta-VAE	AE	LODA	ODIN	HBOS	SOD	ensemble-LOF	DynamicHBOS	VAE	COPOD	OCSVM	DeepSVDD	genzout	ABOD	CBLOF	kth-NN	sb-DeepSVDD
PCA	1.0	0.9	**0.015**	0.9	**0.001**	**0.669**	**0.001**	0.9	0.731	0.9	**0.001**	0.9	0.599	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.118	**0.005**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
LMDD	0.9	1.0	0.23	0.634	**0.002**	**0.126**	**0.026**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.651	0.105	0.9	0.9	0.9	0.658	0.9	**0.009**	0.484	0.9	**0.001**	**0.001**
SO-GAAL	**0.015**	0.23	1.0	**0.001**	0.9	**0.001**	0.9	0.054	0.9	**0.004**	0.9	**0.001**	0.9	**0.001**	0.9	0.9	**0.001**	**0.001**	**0.001**	0.9	**0.005**	0.9	0.9	**0.001**	0.216	**0.001**	**0.001**	0.9	0.335	0.9	**0.002**	0.9
IF	0.9	0.634	**0.001**	1.0	**0.001**	0.9	**0.001**	0.9	**0.004**	0.9	**0.001**	0.9	**0.002**	0.9	0.9	0.9	0.9	0.9	**0.021**	**0.001**	0.9	**0.001**	**0.001**	0.9	0.651	0.9	0.9	**0.001**	0.9	0.524	**0.001**	0.9	**0.001**
LOF	**0.001**	**0.002**	0.9	**0.001**	1.0	**0.001**	0.9	**0.001**	0.519	**0.001**	0.9	**0.001**	0.651	**0.001**	0.9	0.9	0.9	**0.001**	0.223	0.9	**0.001**	0.9	0.9	**0.001**	**0.002**	**0.001**	**0.001**	0.9	**0.004**	0.9	**0.001**	0.9
EIF	0.669	0.126	**0.001**	0.9	**0.001**	1.0	**0.001**	0.424	**0.001**	**0.898**	**0.001**	0.9	**0.001**	0.546	0.9	0.9	0.9	0.9	**0.001**	**0.001**	0.9	0.9	**0.001**	**0.001**	0.136	0.9	0.9	**0.001**	0.9	**0.078**	**0.001**	0.9	**0.001**
ALAD	**0.001**	**0.026**	0.9	**0.001**	0.9	**0.001**	1.0	**0.003**	0.9	0.9	**0.001**	0.9	**0.001**	**0.002**	**0.001**	**0.001**	**0.001**	**0.001**	0.669	0.9	**0.001**	0.9	0.9	**0.001**	**0.023**	**0.001**	**0.001**	0.9	**0.001**	**0.046**	0.9	**0.001**	0.9
kNN	0.9	0.9	0.054	0.9	**0.001**	0.424	**0.003**	1.0	0.9	0.9	**0.001**	0.9	**0.836**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.295**	**0.019**	0.9	0.9	0.9	0.9	**0.001**	0.819	0.9	**0.001**	0.9	**0.001**
LUNAR	0.731	0.9	0.9	**0.004**	0.519	**0.001**	0.9	1.0	0.502	**0.096**	0.198	0.9	0.854	0.312	0.138	0.286	0.424	0.9	0.336	0.55	0.9	0.9	0.219	0.9	**0.004**	0.195	0.775	**0.001**	0.9	0.147	0.371	0.126
KDE	0.9	0.9	**0.004**	0.9	**0.001**	**0.858**	**0.001**	0.9	0.502	1.0	**0.001**	0.9	0.35	0.9	0.9	0.9	0.9	0.9	**0.784**	**0.001**	0.9	**0.039**	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
COF	**0.001**	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.056	**0.001**	1.0	**0.001**	0.105	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.012**	0.9	**0.001**	0.572	0.9	**0.001**	**0.001**	**0.001**	**0.001**	0.9	**0.001**	**0.001**	0.9	**0.001**	0.9
MCD	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.198	0.9	**0.001**	1.0	0.116	0.9	0.9	0.9	0.9	0.9	**0.488**	**0.001**	0.9	**0.007**	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
GMM	0.599	0.9	0.9	**0.002**	**0.051**	**0.001**	0.9	**0.836**	0.9	0.35	0.105	0.116	1.0	0.722	0.195	0.078	0.175	0.277	0.9	0.51	**0.408**	0.9	0.9	0.128	0.9	**0.002**	0.114	0.9	**0.001**	0.9	0.245	0.237	0.216
u-CBLOF	0.9	0.9	**0.03**	0.9	**0.001**	**0.546**	**0.002**	0.9	0.854	0.9	**0.001**	0.9	0.722	1.0	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.195**	**0.01**	0.9	0.9	0.9	**0.484**	0.493	**0.009**	0.9	**0.039**	0.669	**0.033**
INNE	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.312	0.9	**0.001**	0.9	0.195	0.9	1.0	0.9	0.9	0.9	0.616	**0.001**	0.9	**0.016**	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
ECOD	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.138	0.9	**0.001**	0.9	0.078	0.9	0.9	1.0	0.9	0.9	0.387	**0.001**	0.9	**0.004**	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
beta-VAE	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.286	0.9	**0.001**	0.9	0.175	0.9	0.9	0.9	1.0	0.9	0.59	**0.001**	0.9	**0.013**	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
AE	0.9	0.9	**0.002**	0.9	**0.001**	0.9	**0.001**	0.9	0.424	0.9	**0.001**	0.9	0.277	0.9	0.9	0.9	0.9	1.0	0.713	**0.001**	0.9	**0.027**	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
LODA	0.9	0.9	0.9	**0.021**	0.223	**0.001**	**0.669**	0.9	0.9	**0.784**	**0.012**	0.488	0.9	0.9	0.616	0.387	0.59	0.713	1.0	0.126	**0.832**	0.9	0.9	0.515	0.9	**0.022**	**0.484**	0.493	**0.009**	0.9	**0.039**	0.669	**0.933**
ODIN	**0.001**	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.36	**0.001**	0.9	**0.001**	0.51	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	0.126	1.0	**0.001**	0.9	0.9	**0.001**	**0.001**	**0.001**	0.9	**0.001**	**0.001**	0.9	**0.001**	0.9
HBOS	0.9	0.9	**0.005**	0.9	**0.001**	**0.85**	**0.001**	0.9	0.55	0.9	**0.001**	0.9	**0.048**	0.9	0.9	0.9	0.9	0.9	**0.832**	**0.001**	1.0	0.05	**0.001**	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**	0.9	**0.001**
SOD	0.118	**0.651**	0.9	**0.001**	0.9	**0.001**	0.9	0.295	0.9	**0.039**	**0.572**	**0.007**	0.9	**0.195**	**0.016**	**0.004**	**0.013**	**0.027**	0.9	0.9	0.9	0.05	1.0	0.9	**0.009**	**0.634**	**0.001**	**0.007**	0.9	**0.001**	0.762	0.784	**0.021**	0.748
ensemble-LOF	**0.005**	0.105	0.9	**0.001**	0.9	**0.001**	0.9	**0.019**	0.9	**0.001**	0.9	**0.001**	0.9	**0.01**	**0.001**	**0.001**	**0.001**	**0.001**	0.9	0.9	**0.001**	0.9	1.0	**0.001**	0.9	**0.096**	**0.001**	**0.001**	0.9	**0.001**	0.167	0.9	**0.001**	0.9
DynamicHBOS	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.219	0.9	**0.001**	0.9	0.128	0.9	0.9	0.9	0.9	0.9	0.9	0.515	**0.001**	0.9	**0.009**	**0.001**	1.0	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.001**
VAE	0.9	0.9	0.216	**0.001**	0.136	**0.023**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.022**	**0.001**	0.9	0.9	**0.001**	**0.001**	0.9	0.654	0.096	0.9	1.0	0.656	0.9	**0.005**	0.9	**0.001**
COPOD	0.9	**0.658**	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	**0.004**	0.9	**0.001**	0.9	**0.002**	0.9	0.9	0.9	0.9	0.9	0.9	**0.484**	**0.001**	0.9	0.9	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.9	**0.528**	**0.001**	0.9	**0.001**
OCSVM	0.9	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	0.195	0.9	**0.001**	0.9	0.114	0.9	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.007**	**0.001**	0.9	0.9	0.9	1.0	**0.001**	0.9	0.9	0.9	**0.001**	0.9	**0.001**
DeepSVDD	**0.001**	**0.009**	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	**0.775**	**0.001**	0.9	**0.001**	0.9	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	0.493	0.9	**0.001**	0.9	0.9	**0.001**	**0.008**	**0.001**	**0.001**	1.0	**0.001**	**0.018**	0.9	**0.001**	0.9
genzout	0.9	0.484	**0.001**	0.9	**0.001**	**0.819**	**0.001**	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	**0.009**	**0.001**	0.9	**0.001**	**0.001**	0.9	0.9	0.502	0.9	0.9	**0.001**	1.0	0.355	**0.001**	0.9	**0.001**
ABOD	0.9	0.9	0.335	0.534	**0.004**	**0.078**	**0.046**	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.762	0.167	0.9	0.9	0.528	0.9	**0.015**	0.355	1.0	**0.001**	0.9	**0.001**
CBLOF	**0.001**	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.447	**0.001**	0.9	**0.001**	0.9	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.239**	0.9	**0.001**	0.784	0.9	**0.001**	**0.001**	**0.001**	**0.001**	0.9	**0.001**	**0.001**	1.0	**0.001**	0.9
kth-NN	0.9	0.9	**0.002**	0.9	**0.001**	0.9	**0.001**	0.9	0.371	0.9	**0.001**	0.9	0.237	0.9	0.9	0.9	0.9	0.9	0.669	**0.001**	0.9	**0.021**	**0.001**	0.9	0.9	0.9	0.9	0.9	**0.001**	0.9	**0.001**	1.0	**0.001**
sb-DeepSVDD	**0.001**	**0.001**	0.9	**0.001**	0.9	**0.001**	0.9	**0.001**	0.126	**0.001**	0.9	**0.001**	0.216	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	0.933	0.9	**0.001**	0.748	0.9	**0.001**	**0.001**	**0.001**	**0.001**	0.9	**0.001**	0.9	**0.001**	**0.001**	1.0

Table 11: The p-values from Nemenyi post-hoc analysis on all algorithm pairs based on the 32 global data sets. P-values below 0.05 have been printed bold.

APPENDIX TO: AUTOENCODERS FOR ANOMALY DETECTION ARE UNRELIABLE

LINEAR NETWORKS WITH BIAS TERMS

Linear neural networks with a bias term, similar to those without a bias term, still exhibit out-of-bounds reconstruction that leads to zero reconstruction loss for certain anomalous data points.

Linear autoencoders with bias terms consist of a single linear encoding layer and a single linear decoding layer, each with an added bias term. Like for linear networks without a bias, all multi-layer networks can be reduced to a single layer autoencoder. At the global optimum the bias terms will recover the process of mean-centering. Note that a simplified version of this proof was presented by Bourlard and Kamp [1].

Let us now consider $\bar{\mathbf{x}} = \frac{1}{m}\mathbf{1}_m\mathbf{X}$, so the vector of length n where each element contains the corresponding column-wise mean of $\mathbf{X}$. We will prove that the reconstruction loss $\mathcal{L}_R(\mathbf{b}_{enc}, \mathbf{b}_{dec}; \mathbf{X}, \hat{\mathbf{X}})$ for fixed $\boldsymbol{W}_{enc}$, $\boldsymbol{W}_{dec}^T$ is minimized by $\mathbf{b}_{enc} = -\bar{\mathbf{x}}\boldsymbol{W}_{enc}$, and $\mathbf{b}_{dec} = \bar{\mathbf{x}}$.

First let us acknowledge that

$$\bar{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i,$$

and thus

$$\sum_{i=1}^{m}(\mathbf{x}_i - \bar{\mathbf{x}}) = \mathbf{0}.$$

We can then express the average reconstruction loss over the entire dataset as:

$$
\begin{aligned}
&\mathcal{L}_R(\mathbf{b}_{enc}, \mathbf{b}_{dec}; \mathbf{X}, \hat{\mathbf{X}}) \\
&= \frac{1}{mn} \sum_{i=1}^{m} |\mathbf{x}_i - \hat{\mathbf{x}}_i|^2 \\
&= \frac{1}{mn} \sum_{i=1}^{m} |\mathbf{x}_i - h(g(\mathbf{x}_i))|^2 \\
&= \frac{1}{mn} \sum_{i=1}^{m} |\mathbf{x}_i - ((\mathbf{x}_i \mathbf{W}_{enc} + \mathbf{b}_{enc})\mathbf{W}_{dec}^T + \mathbf{b}_{dec})|^2 \\
&= \frac{1}{mn} \sum_{i=1}^{m} |\mathbf{x}_i - \mathbf{x}_i \mathbf{W}_{enc}\mathbf{W}_{dec}^T - \mathbf{b}_{enc}\mathbf{W}_{dec}^T - \mathbf{b}_{dec}|^2 \\
&= \frac{1}{mn} \sum_{i=1}^{m} |\mathbf{x}_i (1 - \mathbf{W}_{enc}\mathbf{W}_{dec}^T) - \mathbf{b}_{enc}\mathbf{W}_{dec}^T - \mathbf{b}_{dec}|^2 \\
&= \frac{1}{mn} \sum_{i=1}^{m} |(\mathbf{x}_i - \bar{\mathbf{x}})(1 - \mathbf{W}_{enc}\mathbf{W}_{dec}^T)|^2 \\
&+ \frac{1}{mn} \sum_{i=1}^{m} |(\bar{\mathbf{x}} - \bar{\mathbf{x}}\mathbf{W}_{enc}\mathbf{W}_{dec}^T) - \mathbf{b}_{enc}\mathbf{W}_{dec}^T - \mathbf{b}_{dec}|^2.
\end{aligned}
$$

Notice that the left term is constant with respect to $\mathbf{b}_{dec}$ and $\mathbf{b}_{enc}$, and the right term is minimized when $\frac{1}{mn} \sum_{i=1}^{m} |(\bar{\mathbf{x}} - \bar{\mathbf{x}}\mathbf{W}_{enc}\mathbf{W}_{dec}^T) - \mathbf{b}_{enc}\mathbf{W}_{dec}^T - \mathbf{b}_{dec}|^2 = 0$. If we now substitute $\mathbf{b}_{enc} = -\bar{\mathbf{x}}\mathbf{W}_{enc}$, and $\mathbf{b}_{dec} = \bar{\mathbf{x}}$:

$$
\begin{aligned}
&\frac{1}{mn} \sum_{i=1}^{m} |(\bar{\mathbf{x}} - \bar{\mathbf{x}}\mathbf{W}_{enc}\mathbf{W}_{dec}^T) - \mathbf{b}_{enc}\mathbf{W}_{dec}^T - \mathbf{b}_{dec}|^2 = \\
&\frac{1}{mn} \sum_{i=1}^{m} |(\bar{\mathbf{x}} - \bar{\mathbf{x}}\mathbf{W}_{enc}\mathbf{W}_{dec}^T) + \bar{\mathbf{x}}\mathbf{W}_{enc}\mathbf{W}_{dec}^T - \bar{\mathbf{x}}|^2 = 0
\end{aligned}
$$

thereby showing that the optimal solution for the biases indeed recovers the process of mean centering.

Also note that the other term now mimics the reconstruction loss on the mean-centered data. This means that we find $\mathbf{V}_d$ by performing PCA not on $\mathbf{X}$, but on $(\mathbf{X} - \bar{\mathbf{X}})$. Again, we can use the same strategy $\mathbf{a} = \mathbf{c}\mathbf{V}_d^T$ to find adversarial anomalies.

REFERENCES

[1] H. Bourlard and Y. Kamp. "Auto-association by multilayer perceptrons and singular value decomposition." In: *Biological Cybernetics* 59.4 (1988), pp. 291–294.

APPENDIX TO: ACQUIRING BETTER LOAD ESTIMATES BY COMBINING ANOMALY AND CHANGE POINT DETECTION IN POWER GRID TIME SERIES MEASUREMENTS

EVALUATED AND BEST HYPERPARAMETERS

An overview of all evaluated hyperparameters can be found in Table 12. The best parameters resulting from optimization on the validation set for each method-ensembling method combination can be found in Table 13 and Table 14.

AUC-ROC PERFORMANCE OF EACH METHOD

To provide insight into the ranking of the anomalies, so without explicit thresholding, we provide an additional plot of the area under the curve of the receiver-operating characteristic (AUC-ROC) in Figure 17.

Table 12: Evaluated hyperparameters.

Method	Hyperparameter	Hyperparameter values
IF per station	$n_{estimators}$	[1000]
	Threshold strategy	[Symmetrical]
IF over all stations	$n_{estimators}$	[1000]
	(q_{lower}%, q_{upper}%)	[(10, 90), (15, 85), (20, 80)]
	Threshold strategy	[Symmetrical]
SPC	(q_{lower}%, q_{upper}%)	[(10, 90), (15, 85), (20, 80)]
	Threshold strategy	[Symmetrical, Asymmetrical]
BS	β	[0.005, 0.008, 0.015, 0.05, 0.08, 0.12]
	l	[150, 200, 288]
	j	[5, 10]
	(q_{lower}%, q_{upper}%)	[(10, 90), (15, 85), (20, 80)]
	C	[L1]
	reference_point	[mean, median, longest_median, longest_mean]
	Threshold strategy	[Symmetrical, Asymmetrical]

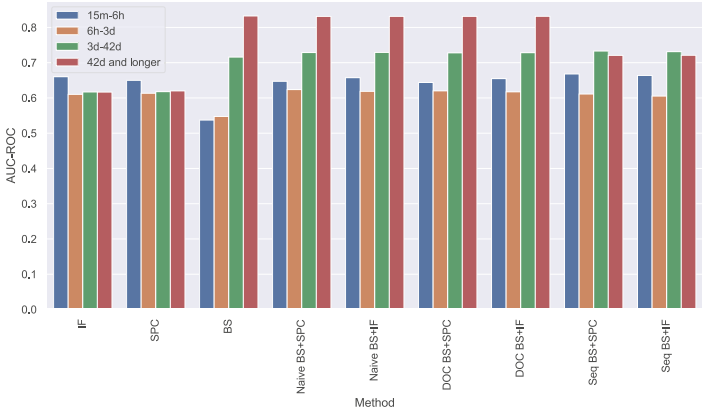


Figure 17: A bar plot of the results of each method per length category in terms of the area under the curve of the receiver-operating characteristic (AUC-ROC).

Table 13: The best parameters resulting from optimization on the validation set for each single method and within the naive ensembles.

Ensemble method	Combination	Method	Hyperparameter	Hyperparameter values
No ensemble	-	IF	$n_{estimators}$	1000
			(q_{lower}%, q_{upper}%)	(15, 85)
			Threshold strategy	Symmetrical
			Optimal threshold(s)	1.27
		SPC	(q_{lower}%, q_{upper}%)	(15, 85)
			Threshold strategy	Symmetrical
			Optimal threshold(s)	2.50
		BS	β	0.008
			j	10
			l	200
			C	L1
			(q_{lower}%, q_{upper}%)	(10, 90)
			reference_point	mean
			Threshold strategy	Asymmetrical
			Optimal threshold(s)	(-0.41, 0.66)
Naive	BS+SPC	BS	β	0.008
			j	10
			l	200
			C	L1
			(q_{lower}%, q_{upper}%)	(10, 90)
			reference_point	mean
			Threshold strategy	Asymmetrical
			Optimal threshold(s)	(-0.41, 0.66)
		SPC	(q_{lower}%, q_{upper}%)	(15, 85)
			Threshold strategy	Symmetrical
			Optimal threshold(s)	2.50
	BS+IF	BS	β	0.008
			j	10
			l	200
			C	L1
			(q_{lower}%, q_{upper}%)	(10, 90)
			reference_point	mean
			Threshold strategy	Asymmetrical
			Optimal threshold(s)	(-0.41, 0.66)
		IF	$n_{estimators}$	1000
			(q_{lower}%, q_{upper}%)	(15, 85)
			Threshold strategy	Symmetrical
			Optimal threshold(s)	1.27

Table 14: The best parameters resulting from optimization on the validation set for each method within the DOC and sequential ensembles.

Ensemble method	Combination	Method	Hyperparameter	Hyperparameter values
DOC	BS+SPC	BS	β	0.008
			j	10
			l	200
			C	L1
			(q_{lower}%, q_{upper}%)	(10, 90)
			reference_point	mean
			Threshold strategy	Asymmetrical
			Optimal threshold(s)	(-0.41, 0.66)
		SPC	(q_{lower}%, q_{upper}%)	(15, 85)
			Threshold strategy	Symmetrical
			Optimal threshold(s)	2.50
	BS+IF	BS	β	0.008
			j	10
			l	200
			C	L1
			(q_{lower}%, q_{upper}%)	(10, 90)
			reference_point	mean
			Threshold strategy	Asymmetrical
			Optimal threshold(s)	(-0.41, 0.66)
		IF	$n_{estimators}$	1000
			(q_{lower}%, q_{upper}%)	(15, 85)
			Threshold strategy	Symmetrical
			Optimal threshold(s)	1.27
Sequential	BS+SPC	BS	β	0.008
			j	10
			l	200
			C	L1
			(q_{lower}%, q_{upper}%)	(15, 85)
			reference_point	mean
			Threshold strategy	Asymmetrical
			Optimal threshold(s)	(-0.49, 0.84)
		SPC	(q_{lower}%, q_{upper}%)	(10, 90)
			Threshold strategy	Symmetrical
			Optimal threshold(s)	2.24
	BS+IF	BS	β	0.008
			j	5
			l	150
			C	L1
			(q_{lower}%, q_{upper}%)	(15, 85)
			reference_point	mean
			Threshold strategy	Asymmetrical
			Optimal threshold(s)	(-0.49, 0.84)
		IF	$n_{estimators}$	1000
			(q_{lower}%, q_{upper}%)	(10, 90)
			Threshold strategy	Symmetrical
			Optimal threshold(s)	1.28

RESEARCH DATA MANAGEMENT

This thesis research has been carried out under the research data management policy of the Institute for Computing and Information Science of Radboud University, The Netherlands.[1]

The following research datasets and software have been produced during this PhD research:

[1] R. Bouman. *Software and Data Related to: Unsupervised Anomaly Detection Algorithms on Real-world Data: How Many Do We Need?* Aug. 2024. DOI: 10.5281/zenodo.13305896. URL: https://doi.org/10.5281/zenodo.13305896.

[2] R. Bouman. *Software Related to: Autoencoders for Anomaly Detection are Unreliable*. Aug. 2024. DOI: 10.5281/zenodo.13308587. URL: https://doi.org/10.5281/zenodo.13308587.

[3] R. Bouman, L. Buise, and L. Schmeitz. *Software Related to: Acquiring Better Load Estimates by Combining Anomaly and Change Point Detection in Power Grid Time Series Measurements*. Version 1.0. Aug. 2024. DOI: 10.5281/zenodo.13308551. URL: https://doi.org/10.5281/zenodo.13308551.

The data used in Chapter 4 is intellectual property of Alliander N.V. but has been open sourced. It is available at Alliander's open data repository.

1 ru.nl/icis/research-data-management/, last accessed August 12th, 2024.

ABOUT THE AUTHOR

Roel Bouman was born on the 14th of December 1993 in Gendringen, The Netherlands. He acquired a double bachelor's degree in Chemistry and Molecular Life Sciences with a minor in Data Science in 2016. He then acquired a double master's degree in Chemistry and Computing Science, specializing in Chemometrics and Data Science respectively. During his master's degree, he participated in the TI-COAST MSc+ program for talents in the analytical sciences. After obtaining both degrees in 2020 he went on to pursue a PhD at the Radboud University, specializing in Machine Learning, under supervision of Tom Heskes in the Data Science group. Roel's specialty is Anomaly Detection, on which he has written several deep dive and application papers. He has collaborated with companies on topics like process control and predictive maintenance and with research institutes like TNO, Fraunhofer and the SCCH.

PUBLICATIONS BY THE AUTHOR

The following is a complete list of publications and preprints by the author: (as of writing date)

[1] R. Bouman, Z. Bukhsh, and T. Heskes. "Unsupervised Anomaly Detection Algorithms on Real-world Data: How Many Do We Need?" In: *Journal of Machine Learning Research* 25.105 (2024), pp. 1–34. URL: http://jmlr.org/papers/v25/23-0570.html.

[2] R. Bouman and T. Heskes. *Autoencoders for Anomaly Detection are Unreliable*. 2025. arXiv: 2501.13864 [cs.LG]. URL: https://arxiv.org/abs/2501.13864.

[3] R. Bouman, L. Schmeitz, L. Buise, J. Heres, Y. Shapovalova, and T. Heskes. "Acquiring better load estimates by combining anomaly and change point detection in power grid time-series measurements." In: *Sustainable Energy, Grids and Networks* 40 (2024), p. 101540. ISSN: 2352-4677. DOI: https://doi.org/10.1016/j.segan.2024.101540. URL: https://www.sciencedirect.com/science/article/pii/S2352467724002698.

[4] R. Folcarelli, S. Van Staveren, R. Bouman, B. Hilvering, G. H. Tinnevelt, G. Postma, O. F. Van Den Brink, L. M. Buydens, N. Vrisekoop, L. Koenderman, et al. "Automated flow cytometric identification of disease-specific cells by the ECLIPSE algorithm." In: *Scientific Reports* 8.1 (2018), p. 10907.

[5] G. van Kollenburg, R. Bouman, T. Offermans, J. Gerretzen, L. Buydens, H.-J. van Manen, and J. Jansen. "Process PLS: Incorporating substantive knowledge into the predictive modelling of multiblock, multistep, multidimensional and multicollinear process data." In: *Computers & Chemical Engineering* 154 (2021), p. 107466.

[6] G. H. van Kollenburg, J. van Es, J. Gerretzen, H. Lanters, R. Bouman, W. Koelewijn, A. N. Davies, L. M. Buydens, H.-J. van Manen, and J. J. Jansen. "Understanding chemical production processes by using PLS path model parameters as soft sensors." In: *Computers & Chemical Engineering* 139 (2020), p. 106841.

[7] R. Stribos, R. Bouman, L. Jimenez, M. Slot, and M. Stoelinga. "A comparison of anomaly detection algorithms with applications on recoater streaking in an additive manufacturing process." In: *Rapid Prototyping Journal* (2024).

ACKNOWLEDGMENTS

Writing acknowledgments is hard. If you know me, you know I can be scatterbrained at times, especially when it concerns lists. As such, let me preface this section with the following: if you are reading this and missing your name, it was unintentional and I will make it up to you by buying you a beverage of your choice!

Then onto the actual list.

First of all I would like to express my thanks to my supervisor Tom. When I was deliberating whether I wanted to pursue a PhD I knew one thing: I only want to do so with a supervisor who is not only capable, but whom I can also trust. I always felt able to share whatever I needed to share, be it personal or professional in nature. I feel like you really helped me grow into a much more capable researcher. I am glad to have more time to learn during my time as a postdoc.

None of the work I did, I did alone. I want to thank all my coauthors during my PhD, Jacco, Linda, Lisandro, Luco, Maaike, Mariëlle, Reinier, Yuliya, and Zaharah for collaborating with me and providing helpful input and great work.

Some people I want to single out for always being willing to discuss ideas and keeping me sharp. Alex, Charlotte, Gabriel, Janneke, Jelle, Marco and Twan: Thank you so much!

Only last year I had the pleasure of being able to go on a research visit to Austria, where I visited the Software Competence Centre Hagenberg to meet up with colleagues there. Florian, you were a very gracious host, thank you for the wonderful stay!

My time at the Data Science department has always been wonderful. What started off great in February 2020 saw a somewhat unfortunate interruption by this stupid global pandemic. Nonetheless, the (then) limited number of colleagues actively endeavored to keep social contacts up during what could have otherwise been a very isolated start of my PhD. Not only did we have a nice Discord channel for collective (digital) lunch, but we also had weekly groups to share our research insights or otherwise complain about whatever PhD life entailed at times. When we finally were able to meet in person again the group quickly grew. From that, we quickly started organizing tons of social activities of all kinds. I especially enjoyed all the board game nights we organized, and are still organizing, as DaS.

I was furthermore lucky to be part of *The Office*™ where the environment was always great and truly *gezellig*, even though at times it was hard to get actual work done when all of us were present. It was all a joy!

To all my wonderful colleagues, past and present, thank you for the wonderful time! (I really tried, but you really are too many to list in full)

So I have this habit of having too many hobbies. One of the hobbies I enjoy spending an exorbitant amount of time on is playing role-playing games (for the uninitiated, this means things like Dungeons & Dragons). I want to thank Glenn, Henk, and Jeff for spending so much time with me on all those lovely obscure, but mostly Swedish, games we have played throughout the years. Rarely have I laughed so much. Bas, Eva, Laura, Louis, and Sjors, thank you for sticking with me throughout the strange adventure on tropical islands and the Feywild West™, even when planning has been horrible when I was busy finishing my thesis. Charlotte, Erkan, Gabriel, Kasper, Mohanna, Parisa, Sjors, and Wouter, thank you for joining and sharing with me a bunch of wacky adventures throughout the Feywild, Arthis Ador, and Noctinem. Thanks especially for being patient with my convoluted plans and turning the game into a real estate simulator.

I have the pleasure of having a great number of wonderful friends, with whom I can always talk, cook, ferment, hike, climb, drink, listen to vinyl, discuss books, or play board games with. As the risk of missing someone would become too great, I just want to thank you all with a bit of a perhaps underwhelming blanket statement: you are wonderful, and I hope to have you all in my life for much longer.

Alex and Janneke, thank you for agreeing to be my paranymphs. Even though your duties have barely begun as I write this, I am sure you will do a great job.

Starting way before my PhD, but of course continuing throughout, my family has always been of great support to me. I am extremely lucky to still have two living grandparents, oma Wil en oma Marietje: thank you for all your love, support, and copious amounts of food and talks. I am also beyond lucky to have many aunts, uncles, and cousins whom I enjoy spending time with, I treasure each family gathering. Carel, Petra, Bas, Femke, and Rens you are like a second family to me, and I always feel fully at home and loved by all of you! To my parents, Julie and Reinier, and my brother Martijn: the support you have given me over the years is immense. Even when I was ridiculously busy, you were always there for me. In many ways my PhD journey was only possible through you.

Even though he obviously cannot read, I want to thank Moki for being the cuddliest, enthusiastic and most pet-deserving dog a man could want.

Laura, your patience and understanding is endless. You are the most wondrous person I ever had the blessing to encounter. Then, for some quirk of the universe, we ended up being together. I treasure you every day. All my love to you for the rest of my days.

- Roel, 04-02-2025

Radboud Universiteit