REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

ORDER STATISTICS BASED TRAINING AND SCORING ALGORITHMS FOR DEEP OUTLIER DETECTION

Ahmet Zahid BALCIOĞLU

MASTER OF SCIENCE THESIS
Department of Statistics
Program of Statistics

Supervisor
Asst. Prof. Erhan ÇENE

July, 2022

REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

ORDER STATISTICS BASED TRAINING AND SCORING ALGORITHMS FOR DEEP OUTLIER DETECTION

A thesis submitted by Ahmet Zahid BALCIOĞLU in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 19.07.2022 in Department of Statistics, Program of Statistics.

Asst. Prof. Erhan ÇENE
Yildiz Technical University
Supervisor

Approved By the Examining Committee

Asst. Prof. Erhan ÇENE, Supervisor
Yildiz Technical University _____

Assoc. Prof. Gülder KEMALBAY, Member
Yildiz Technical University _____

Asst. Prof. Arzu BAYGÜL EDEN, Member
Koç University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of Order Statistics Based Training and Scoring Algorithms for Deep Outlier Detection supervised by my supervisor, Asst. Prof. Erhan ÇENE. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Ahmet Zahid BALCIOĞLU

Signature

To my friend and colleague Oğuz Gürerk
and my dear mentor Sevinç Gökşen.

ACKNOWLEDGEMENTS

This work grew out of a research project with Oğuz Gürerk wherein we endeavoured to characterize the concept of an outlier in a close relation with the notion of a tail index. The definition introduced in Chapter 4 was originally developed as part of the project. The computational aspect which constitutes the heart of this dissertation arose naturally when I was investigating ways to use our definition in the multivariate case. Besides our continuous collaboration, I would like to thank Oğuz Gürerk for our valuable discussions and moral support.

I would like to express my gratitude for my advisor Erhan Çene who has been a source of guidance and reassurance. Throughout my master's, I was at ease to ask him about anything and learned a lot from our valuable discussions.

Finally, I would like to thank my dear aunt Hatice for her constructive criticism of the manuscript.

Ahmet Zahid BALCIOĞLU

TABLE OF CONTENTS

LIST OF ABBREVIATIONS vii

LIST OF FIGURES viii

LIST OF TABLES x

ABSTRACT xi

ÖZET xiii

1 INTRODUCTION 1
- 1.1 Literature Review . . . 1
- 1.2 Objective of the Thesis . . . 2
- 1.3 Hypothesis . . . 2

2 Outlier Analysis 4
- 2.1 What is an Outlier? . . . 4
- 2.2 Types of Outliers . . . 5
- 2.3 Outlier Detection Methods . . . 6
- 2.4 Domains of Outlier Detection . . . 8

3 Outlier Detection with Autoencoders 10
- 3.1 Introduction . . . 10
  - 3.1.1 Principal Component Analysis . . . 10
  - 3.1.2 Autoencoder Networks . . . 11
- 3.2 Autoencoders in Anomaly Detection . . . 13
- 3.3 Shortcomings of Autoencoders . . . 14

4 R-Statistics for Outlier Detection 15
- 4.1 Order Statistics in Outlier Detection . . . 15
- 4.2 The Definition of R-Statistic . . . 17
- 4.3 Application: Simulation Studies . . . 19
- 4.4 Elbow Detection . . . 19
  - 4.4.1 Kneedle algorithm . . . 21

4.5 Simulation Studies Revisited 21

5 Towards Robust Autoencoders with R-Statistic **23**
5.1 Order Statistic Augmented Loss 23
5.1.1 Investigating weight functions 23
5.2 Kappa Threshold Early Stopping 24

6 Case Studies **27**
6.1 Experiments on MNIST digit 27
6.2 Experiments on Musk 28
6.3 Results 29

7 RESULTS AND DISCUSSION **37**

REFERENCES **41**

A MISCELLANEOUS PLOTS **46**
A.1 Extra plots from MNIST digit 0 46
A.2 Extra plots from MNIST digit 1 47
A.3 Extra plots from MNIST digit 2 49
A.4 Extra plots from MNIST digit 3 50
A.5 Extra plots from MNIST digit 4 52
A.6 Extra plots from MNIST digit 5 53
A.7 Extra plots from MNIST digit 6 55
A.8 Extra plots from MNIST digit 7 56
A.9 Extra plots from MNIST digit 8 58
A.10 Extra plots from MNIST digit 9 59
A.11 Extra plots from Musk 61

PUBLICATIONS FROM THE THESIS **63**

LIST OF ABBREVIATIONS

APS	Advanced Photon Source
AUC	Area under the ROC Curve
CBLOF	Cluster Based Local Outlier Factor
CIFAR	Canadian Institute For Advanced Research
cdf	Cumulative distribution function
EVT	Extreme Value Theory
EWMA	Exponentially Weighted Moving Average
GAN	Generative Adversarial Networks
i.i.d.	Independent and identically distributed
k-NN	k-Nearest Neighbours
MSE	Mean Squared Error
PCA	Principal Components Analysis
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
pdf	Probability density function
UCI	University of California Irvine
VAE	Variational Autoencoder

LIST OF FIGURES

Figure 2.1 Examples of outliers [20] 6
Figure 3.1 Structure of an autoencoder with three hidden layers [46]. 12
Figure 4.1 Sorted samples of 1 000 000 taken from half-normal, half-Cauchy and χ^2 distributions. We show the order statistic from the samples on the left, and the corresponding $R = \frac{\sum_{i=1}^{n-k} X_{(i)}}{\sum_{i=n-k+1}^{n} X_{(i)}}$ for each point on the right. 20
Figure 4.2 Outlier κ-thresholds selected by the kneedle algorithm for the samples in Figure 4.1 22
Figure 5.1 Unaltered and 3 epoch aggregated κ-threshold values during the first 100 epochs of a training session. 26
Figure 6.1 Examples from MNIST data set [60]. 28
Figure 6.2 Outliers in the musk dataset. 29
Figure A.1 Outlier plots for the train set of MNIST digit 0. 46
Figure A.2 Outlier plots for the test set of MNIST digit 0. 46
Figure A.3 κ-threshold values during training of the algorithm for MNIST digit 0. 47
Figure A.4 ROC curves for the test set of MNIST digit 0. 47
Figure A.5 Outlier plots for the train set of MNIST digit 1. 47
Figure A.6 Outlier plots for the test set of MNIST digit 1. 48
Figure A.7 κ-threshold values during training of the algorithm for MNIST digit 1. 48
Figure A.8 ROC curves for the test set of MNIST digit 1. 48
Figure A.9 Outlier plots for the train set of MNIST digit 2. 49
Figure A.10 Outlier plots for the test set of MNIST digit 2. 49
Figure A.11 κ-threshold values during training of the algorithm for MNIST digit 2. 50
Figure A.12 ROC curves for the test set of MNIST digit 2. 50
Figure A.13 Outlier plots for the train set of MNIST digit 3. 50
Figure A.14 Outlier plots for the test set of MNIST digit 3. 51
Figure A.15 κ-threshold values during training of the algorithm for MNIST digit 3. 51

Figure A.16 ROC curves for the test set of MNIST digit 3. 51
Figure A.17 Outlier plots for the train set of MNIST digit 4. 52
Figure A.18 Outlier plots for the test set of MNIST digit 4. 52
Figure A.19 κ-threshold values during training of the algorithm for MNIST digit 4. 53
Figure A.20 ROC curves for the test set of MNIST digit 4. 53
Figure A.21 Outlier plots for the train set of MNIST digit 5. 53
Figure A.22 Outlier plots for the test set of MNIST digit 5. 54
Figure A.23 κ-threshold values during training of the algorithm for MNIST digit 5. 54
Figure A.24 ROC curves for the test set of MNIST digit 5. 54
Figure A.25 Outlier plots for the train set of MNIST digit 6. 55
Figure A.26 Outlier plots for the test set of MNIST digit 6. 55
Figure A.27 κ-threshold values during training of the algorithm for MNIST digit 6. 56
Figure A.28 ROC curves for the test set of MNIST digit 6. 56
Figure A.29 Outlier plots for the train set of MNIST digit 7. 56
Figure A.30 Outlier plots for the test set of MNIST digit 7. 57
Figure A.31 κ-threshold values during training of the algorithm for MNIST digit 7. 57
Figure A.32 ROC curves for the test set of MNIST digit 7. 57
Figure A.33 Outlier plots for the train set of MNIST digit 8. 58
Figure A.34 Outlier plots for the test set of MNIST digit 8. 58
Figure A.35 κ-threshold values during training of the algorithm for MNIST digit 8. 59
Figure A.36 ROC curves for the test set of MNIST digit 8. 59
Figure A.37 Outlier plots for the train set of MNIST digit 9. 59
Figure A.38 Outlier plots for the test set of MNIST digit 9. 60
Figure A.39 κ-threshold values during training of the algorithm for MNIST digit 9. 60
Figure A.40 ROC curves for the test set of MNIST digit 9. 60
Figure A.41 Outlier plots for the train set of Musk. 61
Figure A.42 Outlier plots for the test set of Musk. 61
Figure A.43 κ-threshold values during training of the algorithm for Musk. . . . 62
Figure A.44 ROC curves for the test set of Musk. 62

LIST OF TABLES

Table 6.1 Training results for the digit 0. 29
Table 6.2 Test results for the digit 0. 30
Table 6.3 Training results for the digit 1. 30
Table 6.4 Test results for the digit 1. 30
Table 6.5 Training results for the digit 2. 31
Table 6.6 Test results for the digit 2. 31
Table 6.7 Training results for the digit 3. 31
Table 6.8 Test results for the digit 3. 32
Table 6.9 Training results for the digit 4. 32
Table 6.10 Test results for the digit 4. 32
Table 6.11 Training results for the digit 5. 33
Table 6.12 Test results for the digit 5. 33
Table 6.13 Training results for the digit 6. 33
Table 6.14 Test results for the digit 6. 34
Table 6.15 Training results for the digit 7. 34
Table 6.16 Test results for the digit 7. 34
Table 6.17 Training results for the digit 8. 35
Table 6.18 Test results for the digit 8. 35
Table 6.19 Training results for the digit 9. 35
Table 6.20 Test results for the digit 9. 36
Table 6.21 Training results for musk. 36
Table 6.22 Test results for musk. 36
Table 7.1 AUC (left) and AP (right) results on MNIST provided by Merrill and Iskandarian [39]. 39
Table 7.2 AUC and $F1$ results on MNIST test sets provided by An and Cho [29]. 39

ABSTRACT

Order Statistics Based Training and Scoring Algorithms for Deep Outlier Detection

Ahmet Zahid BALCIOĞLU

Department of Statistics
Master of Science Thesis

Supervisor: Asst. Prof. Erhan ÇENE

Autoencoder networks are among the most popular deep learning methods for outlier detection. They work as an unsupervised algorithm that aims to reconstruct original instances through learning a lower dimensional feature space. An important problem with autoencoders is that due to their complexity, they can generalize to learn outliers just as well as normal instances. This is also exacerbated by the fact that outliers can act as points of leverage during training. In this work, we propose to solve this problem by introducing an order statistics based definition of an outlier. Using our definition we quantify the cut-off point between outliers and normal data in a new statistic. We discuss the proposed statistic and demonstrate its effectiveness through simulation studies. Furthermore, we use kneecap detection on our statistic to automatically find a cut-off point for outliers. We incorporate our statistic to autoencoder training and show that with our proposed statistic, autoencoder model no longer needs the percentage of anomalies as a priori to estimate the cut-off point. We also introduce an order statistics augmented loss function, which creates a more robust model by penalizing the leverage effect of outliers in the loss signal. Finally, we introduce an early stopping algorithm specifically for outlier detection. We show that training with our algorithms show a significant improvement when compared to conventional autoencoder training on image and chemical datasets.

Keywords: Outlier detection, order statistics, anomaly detection, autoencoder, robust statistics

ÖZET

Derin Aykırı Değer Analizi için Sıra İstatistik Esaslı Optimizasyon ve Puanlama Algoritmaları

Ahmet Zahid BALCIOĞLU

İstatistik Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Dr.Öğr.Üyesi Erhan ÇENE

Otomatik kodlayıcı ağlar, aykırı değer tespiti için en meşhur derin öğrenme yöntemleri arasındadır. Daha düşük boyutlu bir özellik uzayını öğrenerek orijinal örnekleri yeniden yapılandırmayı amaçlayan denetimsiz bir algoritma olarak çalışırlar. Otomatik kodlayıcıların aykırı değer tespitinde önemli bir eksiği, karmaşıklıkları nedeniyle eğitim esnasında aykırı değerleri normal örnekler gibi öğrenebilmesidir. Bu durum aykırı değerlerin eğitim sırasında kaldıraç noktaları gibi davranabilmesi gerçeğiyle daha da kötüleşir. Bu çalışmada, aykırı değerlerin sıra istatistiğine dayalı bir tanımını sunarak bu sorunu çözmeyi öneriyoruz. Aykırı değerler ve normal veriler arasındaki ayrım noktasını tanımladığımız yeni istatistikle sayısal olarak belirliyoruz. Önerilen istatistiği tartışıp, benzetim çalışmaları ile etkinliğini gösteriyoruz. Ayrıca, aykırı değerler için otomatik olarak bir ayrım noktası bulmak adına istatistiğimizi diz kapağı (kneecap) tespit yöntemiyle birleştiriyoruz. İstatistiğimizi otomatik kodlayıcı eğitimine dahil edip bu şekilde ayrım noktasını tahmin etmek için aykırı değer yüzdesinin önsel olarak bilinme ihtiyacını ortadan kaldırıyoruz. Ayrıca kayıp fonksiyonundaki kaldıraç etkisi yüksek aykırı değerleri cezalandırarak daha dayanıklı bir model oluşturan, sıra istatistik tabanlı arttırılmış kayıp fonksiyonunu alan yazına kazandırıyoruz. Son olarak, özellikle aykırı değer tespiti için geliştirilmiş bir erken durdurma algoritması öneriyoruz. Algoritmamızı görüntü ve kimyasal veri kümeleri üzerinde eğiterek, geleneksel otomatik kodlayıcılara göre kayda değer bir gelişim gösterdiğini ortaya koyuyoruz.

Anahtar Kelimeler: Aykırı değer analizi, sıra istatistikleri, otomatik kodlayıcı, otokodlayıcı

1
INTRODUCTION

1.1 Literature Review

Outlier Analysis concerns itself in the study of points and patterns in the data that deviate from the rest to the point of arousing suspicions that "it was generated by a different mechanism" [1]. Such points, which are called outliers, are a common occurrence in many real-life datasets. The study of outliers is a growing research field with increasing contributions from probability, statistics, and machine learning literature.

Outlier Analysis have many applications in physics and astronomy where it has been used in particle accelerators [2] and in the discovery of new stars [3]. There are also numerous applications in medicine where outlier analysis can help with medical diagnosis [4]. In network intrusion detection [5], a detected outlier could mean a hacked computer sending out sensitive data [6]. Likewise in fraud detection [7], an outlier could mean a hacked credit card, or a stolen identity. In other applications such as in sensor data, outlier analysis is useful for providing information about potentially faulty components [8, 9].

Order statistics are among the most important mathematical tools for outlier analysis. They play a crucial part in quantifying extreme or tail-events. In many cases the extreme event studied can be of great interest to the public. One of the most famous examples is in Netherlands, where 40% of the country lies below the sea-level. For safety, these regions have to be protected sea-dykes. The question is how tall the dykes should be in order to provide sufficient protection from flooding, while remaining cost efficient. In order to answer this question, statisticians look at the past data of storms in the region and determine the height of the dykes based on the maximum possible height of an incoming wave [10].

With the advent of deep learning, autoencoders were introduced to the outlier analysis literature [11]. Autoencoders have become among the most popular outlier detection

methods due to their ability to address problems in high-dimensional and complex data [12]. They are also highly versatile and can easily adapt to different types of data such as time series, and spatial data.

Despite their recent success, autoencoders are far from a universal solution to all outlier detection problems [12]. They also suffer greatly in the presence of too many outliers in the training data. In order to ameliorate some of these deficiencies Zhou and Paffenroth proposed a robust autoencoder for outlier detection [13] inspired by robust principal component analysis. Likewise a more robust variational autoencoder variant, β-VAE [14], was developed to create a more reliable variational autoencoder for outlier detection.

1.2 Objective of the Thesis

Robust methods that are successful in high dimensional data remains a challenge for outlier detection. Though we have named a few recent advances, these methods still rely on a priori knowledge such as the percentage of outliers in the data. Our goal is to incorporate order statistics into autoencoder training in order to create a robust outlier detection method and remain successful in high-dimensional data. Furthermore, we wish to eliminate the need for any a priori by finding a natural cut-off point using order statistics.

Main contributions of our work are as follows:

1. A robust training and scoring methods using order statistics.
2. An order statistic augmented loss function.
3. An early stopping algorithm specifically for autoencoder training.
4. An open-source python library [15] containing implementations of our algorithms.

1.3 Hypothesis

Let $X_1, X_2, \ldots, X_n$ be independent and identically distributed (i.i.d.) non-negative random variables with cumulative distribution function F. Let $X_{(1)}, \ldots, X_{(n)}$ be the corresponding (increasing) order statistics. We treat the problem of outliers in a way

that the number of κ-outliers is defined via moving blocks as [16]

$$\mathcal{O}_n := n - \min\left\{ i : \sum_{j=1}^{i} X_{(j)} < \kappa \sum_{j=i+1}^{n} X_{(j)} \right\}. \tag{1.1}$$

Using this definition, we can parametrize outliers in data by the κ variable. We can assign outlier scores to each data point in autoencoder training and use it to augment the loss function. Finally, by selecting κ values automatically during each iteration, we propose an early stopping algorithm specifically for autoencoders. We posit that these changes will result in a more robust and more successful autoencoders for outlier detection.

2
Outlier Analysis

2.1 What is an Outlier?

Formally, detecting outliers have been studied in the statistics community as early as the 19^{th} century [17]. Even earlier Daniel Bernoulli, writing in 1777 about the combination of astronomical observations, said:

Is it right to hold that the several observations are of the same weight or moment, or equally prone to any and every error? ... Is there everywhere the same probability? Such an assertion would be quite absurd, which is undoubtedly the reason why astronomers prefer to reject completely observations which they judge to be too wide of the truth, while retaining the rest and, indeed, assigning to them the same reliability. ... I see no way of drawing a dividing line between those that are to be utterly rejected and those that are to be wholly retained; it may even happen that the rejected observation is the one that would have supplied the best correction to the others. Nevertheless, I do not condemn in every case the principle of rejecting one or other of the observations, indeed I approve it, whenever in the course of observation an accident occurs which in itself raises an immediate scruple in the mind of the observer, before he has considered the event and compared it with the other observations. If there is no such reason for dissatisfaction I think each and every observation should be admitted whatever its quality, as long as the observer is conscious that he has taken every care.[18]

250 years old advice of Bernoulli still sounds all too familiar to a statistician today. The reason his advice has remained relevant is rooted in the problem of defining an outlier. Today there are numerous mathematical formalisms. Cluster based Local Outlier Factor CBLOF [19] for instance, first divides the data into clusters and assigns outlier scores to each point based on their distance to clusters. Autoencoders, in contrast, have no assumption of clusters and treat each point individually. Each definition has its advantages and there is no consensus as to what the right notion ought to be.

There are a few good reasons for this diversity. For one the notion of outlier could

be subjective and relative, and dependent upon the field of study. Even the smallest deviations from the expected outcome could be alarming for a medical statistician, while the students of econometric and time-series data come across greater deviations by the hour. Whereas in network intrusion detection, it is entirely possible that data would have more "outliers" compared to normal observations.

In this work we will stray away from an all-encompassing definition of an outlier, but rather focus on a definition for the one dimensional case in terms of order statistics. For the multivariate case, instead of extending our definition and thus falling into the same trap, we can work with whichever definition describes the problem best, as our definition is innately compatible with any definition which produces outlier scores. Another advantage is that we can treat data independent of its distribution and in a sense approach the outlier problem from the perspective of investigating data points "that arouse suspicions that it was generated by a different mechanism" [1].

2.2 Types of Outliers

The nature of the outlier being studied is important when selecting an outlier detection method. Outliers are studied in three categories of point anomalies, contextual anomalies, and collective anomalies.

Point anomalies are data points which are singled out and distant with respect to the rest of the data. Point anomalies are the most basic type of anomaly and is the focus of the majority of outlier research. For example in Figure 2.1 the points O_1 and O_2 are isolated and distant from the rest. They are good examples of point outliers.

Contextual anomalies, or conditional anomalies are data points which are anomalous in a specific context. A point is considered a contextual anomaly if its value is an outlier conditioned on one or multiple variables. In general, conditional outliers occur within time series, and spatial data. We refer the reader to [21] for a more detailed account of conditional anomaly detection.

Collective anomalies are groups of data points which when viewed together seem to be outliers. The key difference is that individual points are not outliers by themselves or within any context, but the group itself is an outlier. For example in Figure 2.1 the points in O_3 are potential candidates for collective outliers.

We refer the reader to [22] and [23] for a treatment of contextual data within sequential and spatial datasets respectively. It should be noted that collective outliers only occur in datasets in which the observations are related, where as contextual

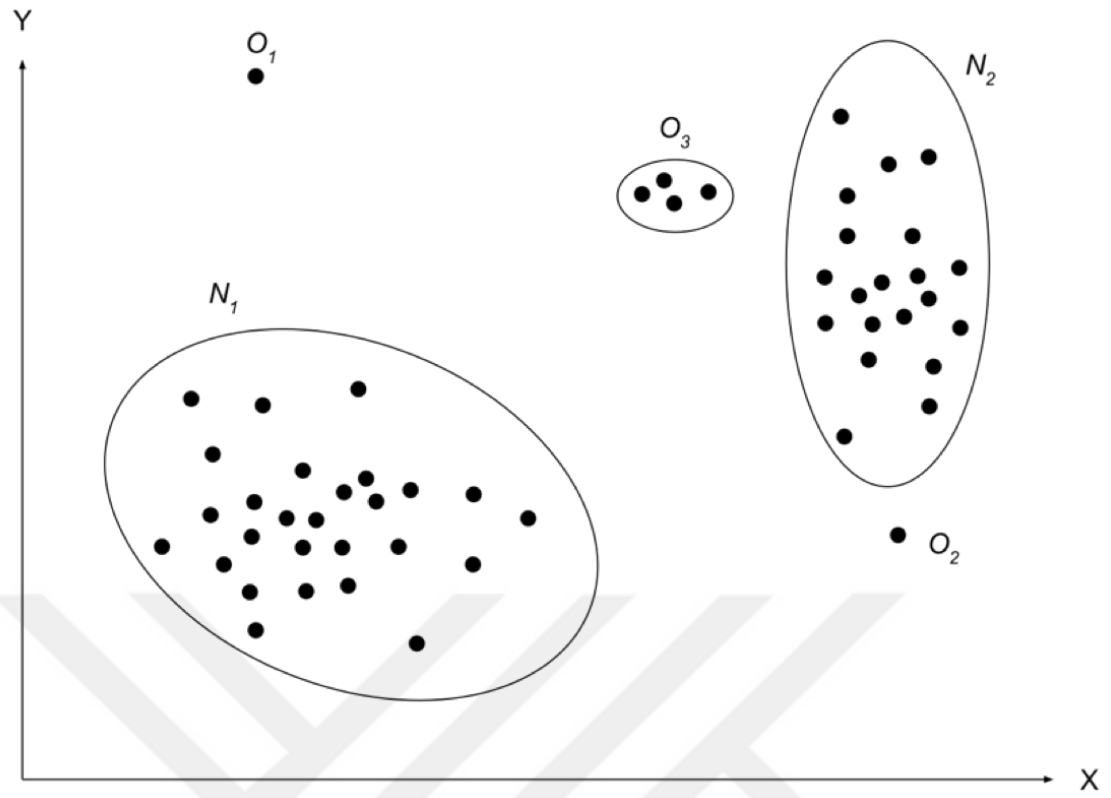


Figure 2.1 Examples of outliers [20]

outliers depend on the existence of contextual variables within the data. Finally, point outliers are a possible feature of any dataset [24].

Autoencoders are generally studied for point and contextual outliers. Due to the availability of different types of architectures, different datasets are studied with different autoencoders. For example Kieu et al. [25] used sparsely-connected RNNs for time series outlier detection. Lu et al. [26] used a deep architecture combining denoising autoencoders with RNNs for outlier detection in sequence data, and Xu et al. [27] used a combination of denoising autoencoder with one-class support vector machines (SVMs) for outlier detection in video data.

2.3 Outlier Detection Methods

Until recently, statistical and probabilistic models dominated the field. However, currently outlier detection is a very popular subject with a lot of machine learning and deep learning studies dedicated. Some of the traditional methods draw inspirations from extreme value analysis, probabilistic mixture modelling, and statistical models such as linear regression.

The goal of statistical and probabilistic outlier detection methods is to model the data

after a known probability distribution [11]. The points that are assigned the lowest probability in the model are then labelled to be the outliers. Linear models, likelihood based models, and spectral models [24] are among classical statistical methods for outlier detection. There are also various hypothesis tests developed for the cases in which the underlying outlier generating model is known [28].

The line between machine learning based methods and statistical methods can be blurred, as much of machine learning methods are statistics based. Informally machine learning methods have been categorized as proximity-based, and deviation based methods [11].

Proximity based methods assume that outlier points are isolated from the rest of the data. Clustering, density based methods, and distance based methods are in this category of outlier detection methods. Clustering based algorithms work by grouping the data into large clusters, next outlier scores are assigned to each point based on its distance to cluster centroids.

Density based methods look for data points that lie in sparse regions of the data. Points that are in relatively sparse locations are then considered to be outliers. Distance based methods use algorithms such as k-NN to establish a measure of distance between a data point and the rest of the data. The points which are far away compared to the rest are defined to be the outliers. One-class SVMs are also among the distance based methods. One-class SVMs are semi-supervised methods which first learn a region for the normal data. For inference, points that do not fall within the learned region are considered to be outliers.

A good example for deviation based methods could be principal component analysis (PCA) or autoencoders. The goal in deviation based methods is to reduce the data into a more compact form using dimension reduction. Each point is then assigned an outlier score, sometimes called the reconstruction error, based on how well it fits with the new representation.

Autoencoders are among the earliest deep learning algorithms studied for outlier detection. Different types of autoencoders such as classical dimension reduction autoencoders [11], variational autoencoders [29] have been used. More recently generative adversarial networks (GANs) have also been studied in the context of outlier detection [30], there are also semi-supervised one-class models that work similar to one-class SVMs [12].

In one of the earliest studies on variational autoencoders for outlier detection, An and Cho [29] studied the MNIST digits [31] dataset. In their experiments, they created

a dataset for each digit and sampled from the rest for outliers, resulting in a total of 10 datasets. For adversarial networks Liu et al. studied generative adversarial active learning for outlier detection, where they proposed two models with single and multiple generators [30]. For testing they used Annthyroid, SpamBase, Waveform and WDBC from the popular UCI repository [32]. There are also more recent probabilistic models such as Empirical Cumulative distribution based Outlier Detection (ECOD) [33] which relies on the empirical cumulative distribution function. They also report results on UCI datasets.

Outlier detection with autoencoders has also found its way to different disciplines. Particularly in physics autoencoders are used from astronomy to particle accelerators. For instance, Chan et al. used a convolutional autoencoder for analyzing the periodic pulsations of stars in order to find stars which have unique physical properties [3]. Edelen and Cook used an autoencoder consisting of 5 layers in order to detect magnet faults in an Advanced Photon Source (APS) storage system [2].

2.4 Domains of Outlier Detection

Outlier detection has many real-life applications in network intrusion detection [34], fraud detection [35], sensor network fault detection [8], medical diagnosis [4], industrial fault detection [24], image processing [36] and numerous other fields. One of the practical aspects of outlier detection is that it can lead to important and actionable information. For instance, in network intrusion detection an outlier could mean a hacked computer sending out sensitive data [6]. Outliers in fraud detection may indicate credit card or identity theft [7], or anomalous observations from spacecraft may indicate faulty components [9].

Fraud detection deals with commercial activities such as credit card fraud detection, insider trading detection, fraudulent insurance claim detection [24]. Hence time series and tabular data are the most common. An example credit card fraud detection dataset is the transactions of European cardholders in September 2013 [35].

Outlier detection in medicine usually deals with patient records. Medical outliers can also be studied in the context of image detection in radiography results [36], or in time series data with electrocardiograms [24]. Some example datasets include Annthyroid [32], which is an outlier detection dataset consisting of records of thyroid patients with a total of 3772 examples for training and 3428 for testing, and Chexpert [37] dataset of radiographs which has a total of 224,316 chest radiographs from 65,240 patients.

Outlier detection has many applications in image processing. Popular image dataset like MNIST [31] and CIFAR10 [38] are used as benchmarks in various studies [29, 39, 40]. There also various use cases from health to satellite imagery. Videos are also a potential data type, motion detection and surveillance videos are just some of the example areas [24, 41].

Sensor network fault detection and industrial fault detection focuses on industrial processes. Outlier detection systems can be important to the industrial plants as they can predict structural defects, and alert against potential faults due to wear and tear. Network intrusion detection is concerned with malicious attacks into computer systems. One problem with network intrusion detection is that outlier rates tend to be unusually high. We refer the reader [5], [42], and [34] for a review of methods in network intrusion detection. For an example dataset see KDD Cup [32].

3
Outlier Detection with Autoencoders

3.1 Introduction

Deep neural networks are generally non-convex and a non-linear statistical models. A typical neural network is a series of linear transformations which are composed in different ways. It is also possible that some non-linear functions are also integrated into the network. These functions are generally referred to as activation functions and contain no learnable parameters [43]. One of the simplest possible neural networks, which is a linear transformation from real numbers to real numbers is called the simple linear regression in the statistics literature. A similar network with an activation function, sigmoid or inverse logit function, is known in statistics as logistic regression.

Depending on the task different kinds of neural networks are possible. Convolutional neural networks (CNNs) incorporate discrete convolutions in themselves which makes them particularly effective against image data. Recurrent neural networks (RNNs) treat multiple data points repetitively with the same linear layer, which proves successful in studying sequential data such as time series or text data.

Autoencoders are special kinds of deep learning methods that can be used for dimension reduction. As the number of units in the layers decreases, the network is forced to learn more efficient representations of the data. By stacking up layers in a deep autoencoder, the network can progressively learn new representations of the data leading to a better reconstruction quality.

3.1.1 Principal Component Analysis

Principal component analysis (PCA) is among the most popular multivariate statistical methods. It was formalized by Hotelling in 1933, but its origins can be traced back to Cauchy and Pearson [44]. PCA takes a dataset $\mathcal{X} = (X_1, \ldots, X_p)$ with p columns, where X_i's can be correlated. The goal is to transform the dataset $\mathcal{X}$ to a new dataset $\mathcal{Y} = (Y_1, \ldots, Y_k)$, with $p > k$, while most of the information is preserved. The columns

of $\mathcal{Y}$ are orthogonal and are called the principal components.

Definition 3.1 (PCA [45])**.** Let $\mathcal{X}$ be a p-dimensional random vector with mean μ and covariance matrix $\Sigma = (\sigma_{ij})$. Let $\lambda_1 \geq \cdots \geq \lambda_p$ be the ordered characteristic roots of Σ and $\gamma_1, \ldots, \gamma_p$ the corresponding characteristic vectors satisfying $\gamma_i' \gamma_j = \delta_{ij}, i, j = 1, \ldots, p$. Then the i-th principal component of $\mathcal{X}$ is defined by

$$Y_i = \gamma_i'(\mathcal{X} - \mu), \quad i = 1, \ldots, p. \tag{3.1}$$

The characteristic roots λ_i and vectors γ_i of Σ are defined by solutions of the equations

$$\Sigma \gamma_i = \lambda_i \gamma_i, \quad i = 1, \ldots, p \tag{3.2}$$

where the γ_i are normalized as

$$\gamma_i' \gamma_i = 1, \quad \gamma_i' \gamma_j = 0, \quad i \neq j \tag{3.3}$$

Let

$$\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_p), \quad \Gamma = (\gamma_1, \ldots, \gamma_p). \tag{3.4}$$

Then,

$$\Sigma = \Gamma \Lambda \Gamma' = \lambda_1 \gamma_1 \gamma_1' + \cdots + \lambda_p \gamma_p \gamma_p'. \tag{3.5}$$

If λ_i is simple, the i-th principal component is defined uniquely except for the multiplication by -1. If some of the λ_i are equal, the corresponding principal components are not defined uniquely but can be given by any orthonormal basis of the latent subspace belonging to that same root.

3.1.2 Autoencoder Networks

An autoencoder is a neural network that is trained to copy its input to its output. It consists of two symmetric parts, with at least a hidden layer in between. The first part, also known as the encoding, or encoder network, tries to learn some low-dimensional feature representation space for the dataset. The second part, likewise called the decoding, or decoder network, tries to recover the original data instances from the low-dimensional representation [43]. A diagram of an autoencoder with three hidden layers is given in Figure 3.1. The network is trained to minimize the recovery error, which is most commonly calculated as the L_2 distance between an observation and its recovered value. The basic formulation is given as follows.

Definition 3.2 (Autoencoder Network [12])**.** The learning task for an autoencoder network, consisting of an encoder network ϕ_e and a decoder network ϕ_d, with the corresponding parameters Θ_e and Θ_d, is

$$\begin{aligned} &\text{for} && \mathbf{z} = \phi_e(\mathbf{x};\Theta_e), \hat{\mathbf{x}} = \phi_d(\mathbf{z};\Theta_d) \\ &\text{minimize} && \textstyle\sum_{\mathbf{x}\in X} \mathcal{L}(\mathbf{x} - \phi_d(\phi_e(\mathbf{x};\Theta_e);\Theta_d)), \end{aligned} \tag{3.6}$$

where $\mathcal{L}$ is the loss function. The optimum parameters for the network are

$$\{\Theta_e^*, \Theta_d^*\} = \arg\min_{\Theta_e,\Theta_d} \sum_{\mathbf{x}\in X} \mathcal{L}(\mathbf{x} - \phi_d(\phi_e(\mathbf{x};\Theta_e);\Theta_d)). \tag{3.7}$$

If we choose the most commonly used L_2 distance for our loss function we have

$$\left\|\mathbf{x} - \phi_d\left(\phi_e\left(\mathbf{x};\Theta_e^*\right);\Theta_d^*\right)\right\|^2. \tag{3.8}$$

Autoencoder networks behave analogous to PCA as they both try to learn a lower dimensional representation of the dataset. In the case of PCA, the lower dimensional representation is reached using singular value decomposition. In the case of autoencoders, the model is trained to optimize over a set of parameters for the chosen loss function and the lower dimensional representation is learned as the hidden layer of the network.

Compared to PCA, autoencoders are more complicated models that have the advantage of learning non-linear representations of the data. In fact when the decoder is linear and the loss is chosen as L_2, an autoencoder learns to span the same subspace as PCA [43]. One key disadvantage of autoencoders is that one can not choose the number of dimensions for the hidden layer after training.

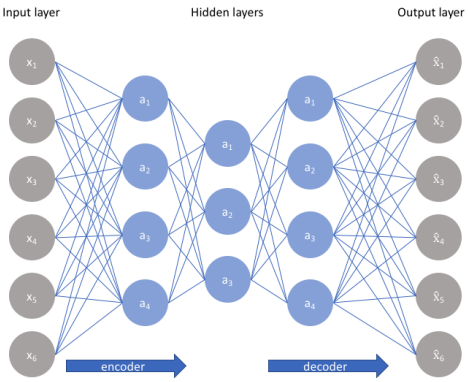


Figure 3.1 Structure of an autoencoder with three hidden layers [46].

3.2 Autoencoders in Anomaly Detection

PCA is one of the classical methods used for outlier detection in multivariate data. All methods using PCA rely on outliers having very little correlation with the principal components. Typically they perform PCA, or robust PCA on the entire data, once data is reduced to a lower number of dimensions, outliers can be identified as data points that do not conform to the rest of the data. For example Shyu et al. used a method in which they assigned an anomaly scores to each point based on its distance from the principal components [47].

Autoencoder networks have become one of the most popular models for outlier detection. Their ability to learn lower dimensional representations of the data in an unsupervised manner, and providing recovery errors for each data point makes them very attractive for outlier detection. We have already mentioned that autoencoder networks share a good deal of similarities with PCA. They are also similar as outlier detection methods as they both depend on the lower dimensional representation to distinguish outliers within the data.

Autoencoders learn a lower dimensional manifold for the dataset. After training, the boundaries of this manifold can be characterized by a threshold value which corresponds to the error values for the data. Points which are not included within the boundaries are candidates to be outliers. Thus, a key assumption in autoencoder models is that anomalous data instances will be more difficult to recover compared to normal instances, and have a higher reconstruction error. In practice, many methods use the reconstruction error directly as the outlier score and label the points above a certain threshold as outliers. We summarize the most elementary algorithm for autoencoder outlier detection in Algorithm 1.

Algorithm 1: Outlier Detection with Autoencoders

Data: $\mathcal{X}$, training data

Train the autoencoder using Equation (3.8) for optimum parameters Θ_e^* and Θ_d^*

Calculate the reconstruction error for each data point $\mathbf{x} \in \mathcal{X}$:

$$\mathcal{L}(\mathbf{x}) = \left\| \mathbf{x} - \phi_d\left(\phi_e\left(\mathbf{x}; \Theta_e^*\right); \Theta_d^*\right) \right\|^2. \tag{3.9}$$

Select a threshold T based on the calculated error values

Observations above T are labelled as outliers:

$$\mathcal{O} = \{x \in \mathcal{X} : \|x - \hat{x}\| \geq T\}. \tag{3.10}$$

3.3 Shortcomings of Autoencoders

One of the key problems of outlier detection is dealing with high-dimensional data, in which most machine learning models fail to produce satisfactory results due to curse of dimensionality [12]. Autoencoders tend be effective against high-dimensional data due to their complex and non-linear nature. However, there are a few key points which their complexity and non-linearity becomes a hindrance.

First, their complexity allows them to generalize to outliers just as well as to the rest of the data, effectively memorizing the dataset [39]. This is also exacerbated by non-linearity, which makes it difficult to determine when to stop the training process. Moreover, even at the beginning of the training process data points with most errors tend to be outliers [39], which makes them behave as potential points of leverage due to their disproportionate contribution to the model early on.

Second, most outlier detection methods using autoencoders use the model residuals as outlier scores. Loss function having a double purpose in training, and scoring introduces some bias to the data. It also makes the model dependent upon the choice of loss function. Another problem with the scoring method is that usually there is no clear cut-off point with which to separate the outliers. As a result most studies either assume a percentage of contamination as a priori, or use another method on top of the autoencoder to determine the cut-off point.

4
R-Statistics for Outlier Detection

If the random variables $X_1, \ldots, X_n$ are arranged in order of magnitude and then written as

$$X_{(1)} \leq \cdots \leq X_{(n)}, \tag{4.1}$$

we call $X_{(i)}$ the i-th order statistic. The study of order statistics is concerned with properties and applications of these ordered random variables, as well as functions involving them [48].

In this chapter, we will endeavour to give a definition of an outlier based on the order statistics of an i.i.d. sample. Then we will demonstrate the efficacy of our definition in a set of simulation studies.

4.1 Order Statistics in Outlier Detection

Order statistics are an important part of statistical study of outliers. They are used in outlier generating models, and for hypothesis testing for potential outliers [28]. They are also crucial for related fields such as extreme value theory (EVT) [49], and for developing robust statistical methods that remain effective in the presence of outliers [18].

Identified outliers model is among the most classical models for outlier generation, in which observations $X_1, \ldots, X_n$ are not i.i.d. but some $k \in \{1, \ldots, n\}$ come from a separate distribution. We will consider the simplistic exponential case, and refer the reader to [50] for a more detailed account.

Definition 4.1. (Identified Outliers Model [28]).
$X = \left\{X_i, \ldots, X_{n-k} : 1 - e^{-x/\theta}; \theta > 0\right\}$, k is known and fixed, for simplicity let's suppose $k = 1$, $X_1, \ldots, X_{n-k}$ are independent and the index of the contaminant is also known. If we assume further that the distribution function of the contaminant is

$$G(x) = F\left(b^{-1}x\right) = 1 - e^{-(b\theta)^{-1}x}, \quad x \in \mathbb{R} \tag{4.2}$$

for some $b \geq 1$, then, without loss of any generality, the joint distribution function of $X_1, \ldots, X_n$ is given by

$$F_{(X_1,\ldots,X_n)}(x_1,\ldots,x_n) = \left[\prod_{i=1}^{n-1}\left(1-e^{-x_i/\theta}\right)\right]\left(1-e^{-x_n/b\theta}\right) \tag{4.3}$$

for $x > 0$, $\theta > 0$, $b \geq 1$.

Order statistics slippage outliers model is another outlier generating model. First a set of i.i.d. random variables are observed, then the last k order statistics are increased by some amount $a > 0$ or multiplied by a $b > 1$. For the case of the exponential distribution the joint distribution function is given below.

Definition 4.2 (Order statistics slippage model [28])**.** Let $X_1, \ldots, X_n$ be independent and exponentially distributed random variables with the rate parameter θ. Suppose that the top k observations are multiplied by $b > 1$, then the joint distribution function would be given by

$$\begin{aligned} &F_{(X_{1:n},\ldots,X_{n:n})}(x_1,\ldots,x_n) \\ &\quad = n!\left[\prod_{i=1}^{n-k}\left(1-e^{-x_i/\theta}\right)\right]\left[\prod_{i=n-k+1}^{n}\left(1-e^{-x_i/b\theta}\right)\right]\cdot \mathbb{1}_{\mathbf{R}^n_{\leq\cdot b;n-k+1}}(x_1,\ldots,x_n). \end{aligned} \tag{4.4}$$

In EVT, extreme order statistics are used to study extreme events as a Poisson process. Sometimes called the first theorem of extreme value theory, Fisher–Tippett–Gnedenko theorem is a fundamental result in this respect [10].

Theorem 4.1 (Fisher–Tippett–Gnedenko)**.** *Let $X_1, \ldots, X_n$ be i.i.d. random variables with common distribution function F. Suppose that there exist a sequence of $a_n > 0$, and $b_n \in \mathbb{R}$ such that the following limit converges to a non-degenerate distribution function*

$$G(x) = \lim_{n\to\infty} \mathbb{P}\left(\frac{\max\{X_1,\ldots,X_n\} - b_n}{a_n} \leq x\right). \tag{4.5}$$

The class of functions G is characterized by a single parameter γ, called the extreme value index, and is given by

$$G_\gamma(x) = \exp\left(-(1+\gamma x)^{-1/\gamma}\right), 1+\gamma x > 0, \tag{4.6}$$

with $\gamma \in \mathbb{R}$ and $\exp(-e^{-x})$ for $\gamma = 0$.

The Theorem 4.1 allows for the convergence in distribution of the normalized sample maxima. But the conditions also imply convergence of other high-order statistics. In

fact, we can write for $X_1, X_2, ..., X_n$ i.i.d. with distribution function F, with F in the domain of attraction in the sense of Theorem 4.1, of a G_γ for $\gamma \in \mathbb{R}$ that the collection of points

$$\left\{\left(\frac{i}{n}, \frac{X_{(i)} - b_n}{a_n}\right)\right\}_{i=1}^{\infty} \tag{4.7}$$

define the following point process, which converges in distribution to a Poisson process [10]

$$N_n(B) = \sum_{i=1}^{\infty} \mathbb{1}_{\left\{\left(\frac{i}{n}, \frac{X_{(i)} - b_n}{a_n}\right)\right\}_{i=1}^{\infty}}. \tag{4.8}$$

For more on the study of extreme order statistics, we refer the reader to [49] and [10]. For the uses of order statistics in the study of robust statistical methods, we refer the reader to [18], [51], and [52].

4.2 The Definition of R-Statistic

As the study of outliers invariably focuses on deviations, it is intuitive and tempting to use order statistics to quantify them. Naturally, the greatest order statistic, the maximum, would be the first suspect. For an example, suppose we consider the case where we have only one potential outlier. More formally, let $X_1, X_2, \ldots, X_n$ be i.i.d. random variables. For our purposes, we can further assume that these are non-negative, since as far as the theory and the computations are concerned non-negativity does not lose generality. Now, let $X_{(1)}, X_{(2)}, \ldots, X_{(n-1)}, X_{(n)}$ be the corresponding order statistics. We are to investigate whether $X_{(n)}$ is an outlier. We begin with a small lemma about the maximum.

Lemma 4.1. *Let $X_1, \ldots, X_n$ be i.i.d. random variables with common cdf F. Then the maximum has the cumulative distribution function*

$$\mathbb{P}(\max\{X_1, \ldots, X_n\} \leq x) = F^n(x). \tag{4.9}$$

Proof.

$$\mathbb{P}(\max\{X_1, \ldots, X_n\} \leq x) = \mathbb{P}\left(\bigcap_{i=1}^{n} \{X_i \leq x\}\right) \tag{4.10}$$

$$= \prod_{i=1}^{n} \mathbb{P}(X_i \leq x) \tag{4.11}$$

$$= F^n(x), \tag{4.12}$$

where we use X_i's are independent in (4.11), and identically distributed in (4.12). ■

One idea is to use Lemma 4.1, see how likely it is to observe the current maximum. This method has several drawbacks, we would have to know the distribution function F beforehand to name just one. Another idea, which does not rely on knowing F, is to look at the ratio $\frac{X_{(n)}}{X_{(n-1)}}$ and compare to $\frac{X_{(i)}}{X_{(i-1)}}$ for $i < n$. This is similar to the definition of an outlier Klebanov et al. [53] gave, which we give below.

Definition 4.3. Let $X_1, \ldots, X_n$ be i.i.d. random variables, and $X_{(1)}, \ldots, X_{(n)}$ be the corresponding order statistics. We say $X_{(n)}$ is an outlier of order $1/\kappa$ if $X_{(n-1)} \leq \kappa X_{(n)}$, where $\kappa \in (0,1)$ is a fixed number.

In the general case, we are investigating the possibility of any number of outliers in the data. In such a case we can not simply extend Definition 4.3 iteratively, as the ratio $\frac{X_{(n)}}{X_{(n-1)}}$ is between two potential outliers and thus meaningless. Ideally, we would like to have a definition that allows us to take into account the entire sample and consider the number of potential anomalies.

Definition 4.4 (**Proposed Method** [16])**.** Let $X_1, X_2, \ldots, X_n$ be i.i.d. non-negative random variables with cumulative distribution function F. We may further assume that these are absolutely continuous and call the common pdf f. Let $X_{(1)}, \ldots, X_{(n)}$ be the corresponding order statistics. We are to investigate the problem of outliers in a way that the number of κ-outliers is defined via moving blocks as

$$\mathcal{O}_n := n - \min\left\{ i : \sum_{j=1}^{i} |X_{(j)}| < \kappa \sum_{j=i+1}^{n} |X_{(j)}| \right\} \tag{4.13}$$

We start by defining the following statistics

$$T_{k,n} \equiv T_k = \sum_{i=n-k+1}^{n} X_{(i)}, \quad 1 \leq k \leq n \tag{4.14}$$

denoting the sum of the top $k \in \{1, \ldots, n\}$ order statistics from a sample of size n and

$$S_{m,n} \equiv S_m = \sum_{i=1}^{m} X_{(i)}, \quad 1 \leq m \leq n \tag{4.15}$$

denoting the sum of the first $m \in \{1, \ldots, n\}$ order statistics from the same sample. Subsequently, putting (4.14) and (4.15) together, we investigate probabilities of the form $P(S_m < \kappa T_{n-m})$ where κ is fixed. We call the random variable

$$R_{m,n} := \mathbb{P}\left(\frac{S_m}{T_{n-m}} \leq \kappa \right) \tag{4.16}$$

as our outlier statistic.

4.3 Application: Simulation Studies

Our proposed statistic (4.16) is an important step away from iteratively considering each point for anomaly. Moreover, it satisfies the informal criterion of taking into account the entire sample. Yet, it is not clear which points are potentially considered as anomalous. There is also no upper bound for the κ parameter as in Definition 4.3.

A natural way to choose the κ parameter is essential for our statistic, as each κ value may correspond to a new cut-off index for potential outliers. A set of simulation studies is needed in order to show how our statistic looks in some of the more well-known distributions, and to get a hint at how to choose the κ value empirically. We will look at normal and χ^2 distributions as they are used in various statistical models. We also choose Cauchy distribution as an example heavy-tailed distribution.

Since our statistic is defined for non-negative distributions, we will look at half-normal, half-Cauchy, and χ^2 distributions. We will take a sample of 1000000 from each distribution and plot the results for each order statistic. The results are plotted in Figure 4.1.

It is clear from the plots that our statistic sharply increases at the tail of each distribution creating an elbow-like region. Except for the Cauchy distribution the increase is more sharp compared to the sample order statistics. This is to be expected as Cauchy is a heavy-tailed distribution and it is comparably more difficult to give a point estimate of a cut-off point for its tail.

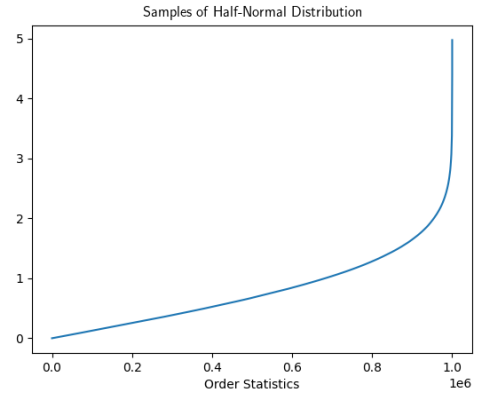
How do we choose the κ-threshold value? One option would be to decide on a case-by-case basis what it should be. Alternatively, we can try to look for a well-defined notion of the "elbow-like region" which our statistic produces.

4.4 Elbow Detection

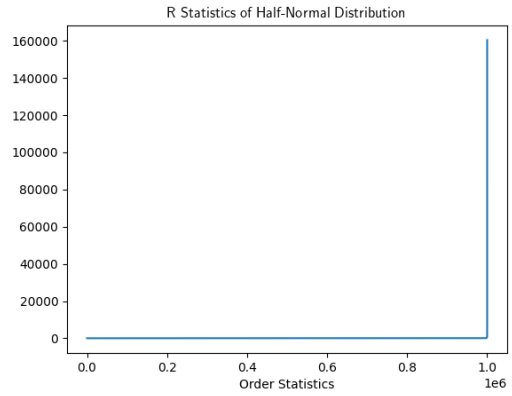
Ideally, we would like to define the cut-off point with respect to the derivatives of the distribution function of our statistic, which would then allow us to define a theoretical cut-off point for any sample. However, there is no established well-defined notion for an elbow point. The literature of elbow detection, also called knee detection, is therefore algorithmically focused [54].

Much of elbow detection literature is based on the following pointwise definition of the curvature of a function. Most works then define the elbow as the point of maximum curvature, and use algorithmic means to calculate it.

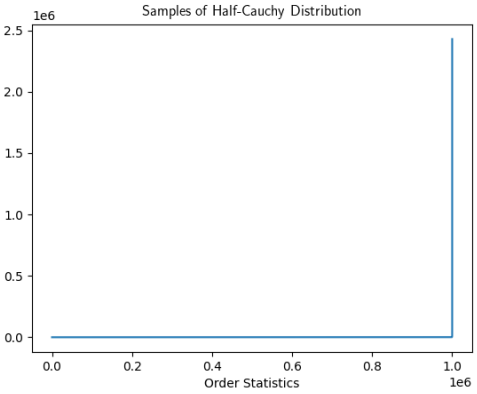
Definition 4.5 (Curvature of a function [55])**.** For any continuous function f, there



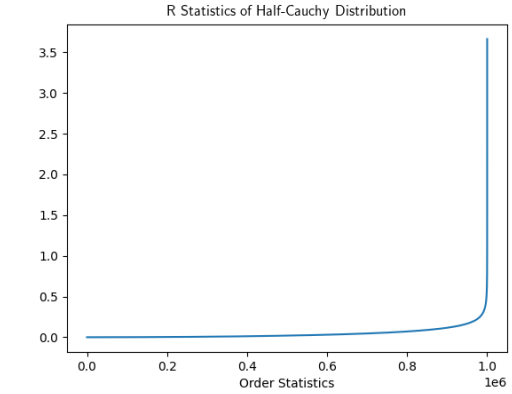
Half-Normal Distribution



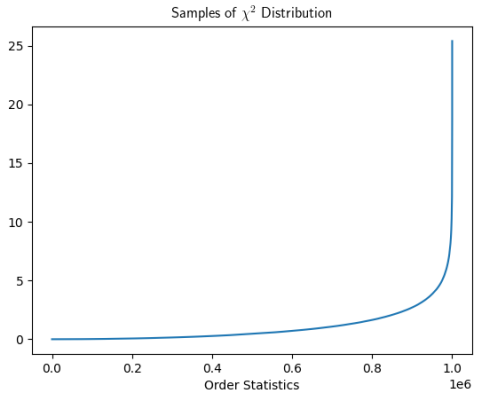
Statistic for Half-Normal Distribution



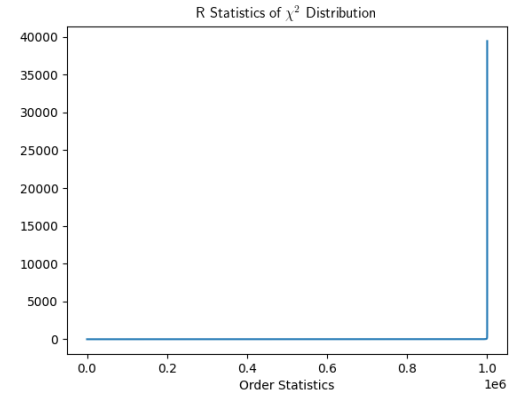
Half-Cauchy Distribution



Statistic for Half-Cauchy Distribution



χ^2 Distribution



Statistic for χ^2 Distribution

Figure 4.1 Sorted samples of 1 000 000 taken from half-normal, half-Cauchy and χ^2 distributions. We show the order statistic from the samples on the left, and the corresponding $R = \frac{\sum_{i=1}^{n-k} X_{(i)}}{\sum_{i=n-k+1}^{n} X_{(i)}}$ for each point on the right.

exists a standard closed-form $K_f(x)$ that defines the curvature of f at any point as a function of its first and second derivative:

$$K_f(x) = \frac{f''(x)}{(1 + f'(x)^2)^{\frac{3}{2}}} \tag{4.17}$$

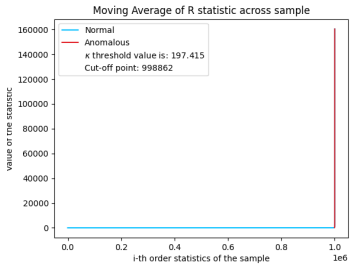
Definition 4.5 of curvature does extend easily to discrete data, instead [55], [56], and [54] use Menger curvature, which is defined for three points as the curvature of the circle circumscribed by those points. There are also angle-based [55] and exponentially weighted moving average (EWMA) based methods [57] which use the differences between successive points and EWMA smoothing to check deviations from arrival times respectfully.

4.4.1 Kneedle algorithm

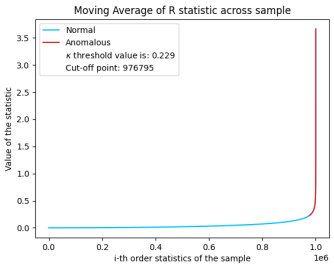
We will use the kneedle detection algorithm for estimating the "elbow point" of our statistic [55]. The kneedle algorithm uses dynamic first derivative threshold, in combination with IsoData [58] to find the elbow points for discrete data. It can work on discrete datasets and has a sensitivity parameter which can be fine-tuned for how sensitively a knee is to be detected. While the smaller values of the sensitivity parameter respond to quick change, the larger values are more robust. Briefly the algorithm tries to find the points of maximum curvature, as the points in which the curve deviates most from a straight line. The knees are defined as the points of the curve which would correspond to a local maxima if the curve is rotated clockwise.

4.5 Simulation Studies Revisited

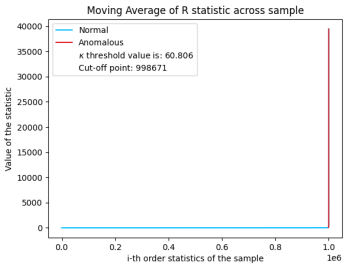
Now that we have a method to select the κ value, we can revisit the simulation studies and see where we would choose our cut-off points. We choose the sensitivity parameter as 5.0. You can see the chosen κ-thresholds for half-normal, half-Cauchy, and χ^2 distributions respectively.



Half-Normal Distribution



Half-Cauchy Distribution



χ^2 distribution

Figure 4.2 Outlier κ-thresholds selected by the kneedle algorithm for the samples in Figure 4.1

5
Towards Robust Autoencoders with R-Statistic

5.1 Order Statistic Augmented Loss

We will discuss the ways we have used to incorporate the *R*-statistic into autoencoder training. It would be ideal to make use of the fact that early in the optimization process the gap between outliers and normal observations is the most noticeable [39]. We have already mentioned that one particular weakness of autoencoders is that it can generalize to outliers as well as normal data due to the high complexity of the model. Practical remedies such as regularization, early stopping, and dropout layers can be used to ameliorate this weakness.

We can use our statistic throughout the optimization procedure to create weights for each point during the calculation of the loss function. In this way we can regularize the contribution of each observation to the loss by its outlier score, and retain the initial distinction between outliers and the rest of the data while optimizing for a model with greater generalizability.

In order to make use of our statistic, we sample a proportion of the data to use as a scoring system for the entire set. As our statistic is relatively inexpensive to calculate, we may also use the entire dataset. After every epoch, we calculate our statistic on this sample to use as outlier scores and also use the kneedle algorithm to estimate the κ-threshold. In the next iteration, after the calculation of loss for each batch, we calculate the weight of each observation by the outlier score of the nearest loss value in the initial sample. We discuss a few methods about what the best weight function would be.

5.1.1 Investigating weight functions

First it is possible to use the inverse *R*-statistic outlier scores in order to increase the weights of normal observations and decrease the weights of the outlier. However, as the weights would not be normalized this method would temper with the loss

function too much and would result in a frail optimization procedure dependent upon the balance in each batch.

Next, the simplest method is to remove the weights of any observation above the κ-threshold to restrain the influence of outliers in the training. This has a few important advantages. It tempers very little with the loss function and incurs very little extra cost. More importantly since it bars data above the κ-threshold from making a contribution, any decrease in the loss function is attributed to the normal data. Therefore, after optimization this method would results in a model in which outliers share very little with the rest of the data.

Finally, as a compromise between the two methods, we can apply inverse weights to data points above the threshold and leave the weights of data below threshold untouched as in the first case. The motivation for this method is to be able to make use of the entire dataset while limiting the contribution from any potential outliers. We give a summary of how the order statistic weighted loss would be implemented in Algorithm 2.

Algorithm 2: Order Statistic Adjusted Loss function

Data: `outlier_sample` := Sample a portion of the data
Initialize uniform loss weights
for *epoch in Total Epochs* **do**
    for *batch in Data* **do**
        Calculate loss
        Estimate loss weights using the `outlier_sample`
    end
    Calculate *R*-statistic for the `outlier_sample`
    Estimate the new κ-threshold for the `outlier_sample`
end

5.2 Kappa Threshold Early Stopping

Deep neural networks are not convex statistical models, as a result their training take longer and do not necessarily converge to a global minimum. This leads to a search for methods which can pre-emptively stop training when the model is successful enough. Early stopping is one such method. It works by monitoring a metric pertinent to the model's success, and stops the training when there is no meaningful improvement on the metric's value.

As autoencoders have high complexity, they can effectively memorize the entire data,

if trained until convergence. Early stopping methods help to stop the autoencoder training before the model starts learning about outliers. Leading to a model that can better distinguish between outliers and the normal observations.

Autoencoders are unsupervised methods, which limits the options for choosing a reliable metric for early stopping. One likely candidate is the loss function itself [39], but the loss function is also used for other algorithms that dynamically change optimizer parameters such as the learning rate. The κ-threshold value, which is calculated in each epoch for the order statistic augmented loss, can be a potential metric. Since stability of the κ value shows that the model has stopped distinguishing between outliers and normal data. During training we would expect the κ-threshold to decrease as the loss is decreasing and stop when the changes in the loss no longer contribute to discriminating outliers.

In practice, because it is a point estimate rather than a summary statistic, κ-threshold is not very consistent. Between consecutive epochs κ-threshold values can increase, despite an overall decreasing trend. It also has a tendency to increase early on in the optimization. As a result it is difficult find a general set of criteria for the κ-threshold to be used in early stopping.

As a solution, we can set a number of preliminary iterations in which the early stopping would not be active. This early iteration limit would fix any confusion caused by the increases in the κ values early on. Since it is early on in the training, we do not expect to be at any disadvantage.

We can look for smoothing methods in order to make the κ estimates more reliable for early stopping. One simple idea would be to use moving block averages. This can lead to a much smoother curve with a more prominent decreasing trend. The trade-off is that as the length of blocks increases, the changes in the κ values aggregated within the blocks become much less prominent. Therefore, it may become more difficult to find a good stopping point. In this context block size is an important hyperparameter balancing between the smoothness of the curve and the ability to pinpoint a good stopping epoch. In Figure 5.1 we plot example κ values from a training session with a block size of 3 and no block aggregation.

In principle, we would like to stop training when the κ values stop decreasing. However, as block-wise smoothing is a key part of our algorithm, we may need to define our stopping condition in terms of the block size. This would help reduce the dependency of our model on block size and give us more repeatable results. Another problem is that if we just focus on the decrease within the κ values, we risk stopping prematurely. Therefore, we will try to stop at a stable region in which κ values do

not change greatly. Due to block aggregation we need select at least 2 times the consecutive κ's as block size. In practice, our algorithm complements our scoring method by stopping at the first point outliers outlier scores are stabilised which also helps with consistency. Please see Algorithm 3 for a summary of our early stopping algorithm.

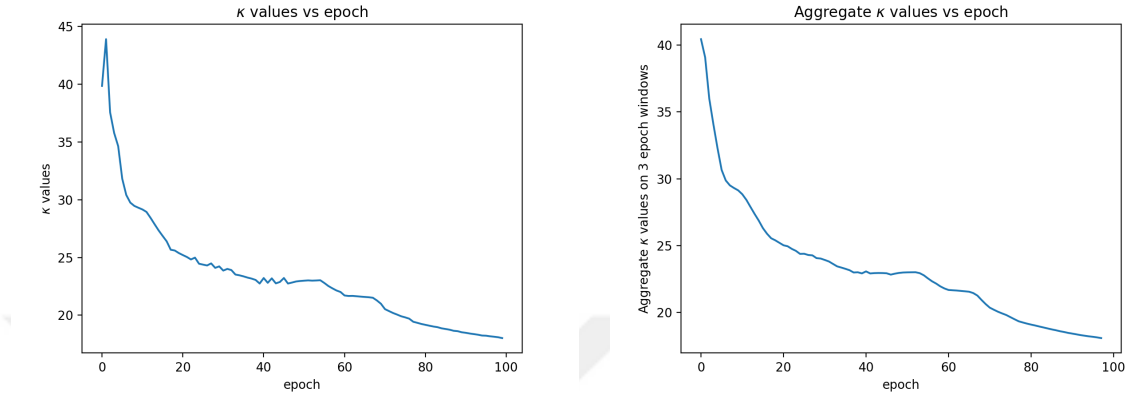


Figure 5.1 Unaltered and 3 epoch aggregated κ-threshold values during the first 100 epochs of a training session.

Algorithm 3: Early Stopping algorithm using κ-threshold

Data: Data := Training data

Data: `outlier_sample` := Sample a portion of the Data

`int early_iteration_limit`

`int block_size`

`κ_epochs = []`

Function `check_improvement(`*κ_epochs, block_size, early_iteration_limit*`):`

- **if** *size(κ_epochs)* $\leq$ `early_iteration_limit` **then**
  - **return** *continue*
- **end**
- `block_κ` = `block_sum(κ_epochs, block_size)`
- **if** `block_sum` *not increasing* **then**
  - **return** *stop*
- **else**
  - **return** *continue*
- **end**

for *epoch in Total Epochs* **do**

- **for** *batch in Data* **do**
  - Calculate loss
- **end**
- Calculate *R*-statistic for the `outlier_sample`
- Calculate κ-threshold for the epoch and append to `κ_epochs`
- `check_improvement(κ_epochs, block_size, early_iteration_limit)`

end

6
Case Studies

Our proposal to improve autoencoders is threefold. An order statistic guided scoring method which can produce outlier scores without any a priori assumptions about the nature of the data, and does not necessarily depend on the choice of loss function. An order statistic augmented loss function which reduces the bias introduced by outliers during training, thus leading to a more robust model. Finally, an early stopping algorithm using the κ-threshold values of each training iteration.

As our algorithms are built on top of already established autoencoders, we conduct ablation studies starting from an autoencoder, adding order-statistics-based scoring, and finally ending with the fully fledged algorithm using κ-threshold early stopping. In this manner we hope to show the efficacy our algorithm in a comparable and repeatable manner. Moreover, we conduct our experiments on public benchmark datasets which have been already studied in the context of outlier detection in order to provide further repeatability. In order to test the reliability of our method, we will pursue experiments in different kinds of datasets. MNIST [31] digits will be used as an image dataset. We will also use the Musk dataset [59] for an experiment on tabular data. All code used in the trainings are provided in our python library [15].

Unless prearranged train and test sets are provided, datasets will be split into 80% - 20% train and test pairs. As outlier detection is a classification problem we will report contingency tables, $F1$ scores, and AUC metrics.

6.1 Experiments on MNIST digit

MNIST is an image data set consisting of handwritten digits [31]. Each image is black and white as seen in Figure 6.1. We conduct ten experiments in which we take each digit to represent normal instances and then undersample from the rest of the data to represent anomalies.



Figure 6.1 Examples from MNIST data set [60].

For all trials with MNIST we follow the architecture used by Cavallari et al. [61]. An autoencoder network with linear layers, the weights are respectively of shapes $(784, 256)$, $(256, 32)$, $(32, 256)$, $(256, 784)$. Prior to training we rescale each image to $[0, 1]$. For each dataset our final sample has an outlier rate of 1%. We use Adam [62] optimizer with initial learning rate set at 1×10^{-3}, for the loss function we use mean squared error (MSE). Batch size in each training session is set to 2991 and the total number of epochs is set to 100 unless it is stopped pre-emptively with early stopping.

In addition to the metrics we report, please see Appendix A for κ-threshold values of each iteration of early stopping, ROC curves, and the outlier plots for the test sets.

6.2 Experiments on Musk

Musk [59] is a chemical dataset comprised of molecules classified to be musk and non-musk. It has 102 molecules in total from which 39 are classified as musks by human experts. The dataset consists of 3062 points 97 (3%) of which are outliers. There are 166 features in total that describe these molecules based on their exact shape, or conformation [32]. The goal is to predict whether an unknown molecule is a musk. A plot of outliers and normal values from the musk dataset is given in Figure 6.2.

Before training we sample from the dataset in order to reduce the percentage of outliers to 1%, we also rescale each column to $[0, 1]$. We conducted a few prior experiments in order to determine the exact specifications of the autoencoder and finally decided on an architecture with consecutive linear and dropout layers for regularization. The linear layers have the following weights $(166, 166)$, $(166, 166)$, $(166, 128)$, $(128, 64)$, $(64, 32)$, $(32, 64)$, $(64, 128)$, $(128, 166)$ each linear had 0.1 l_2 regularization and was followed by a non-linear rectified linear unit (ReLU) function,

with the exception of the final layer. For the dropout layers 0.2 dropout rate was chosen. This autoencoder architecture is inspired by the autoencoder model provided by the pyod library [63]. We use Adam optimizer with initial learning rate set at 1×10^{-3}, for the loss function we use MSE. Batch size in each training session is set to 32 and the total number of epochs is set to 100 unless it is stopped pre-emptively with early stopping.

Similar to trials on MNIST, we report the κ-threshold values of each iteration of early stopping, ROC curves, and the outlier plots for the test sets. Please see Appendix A.

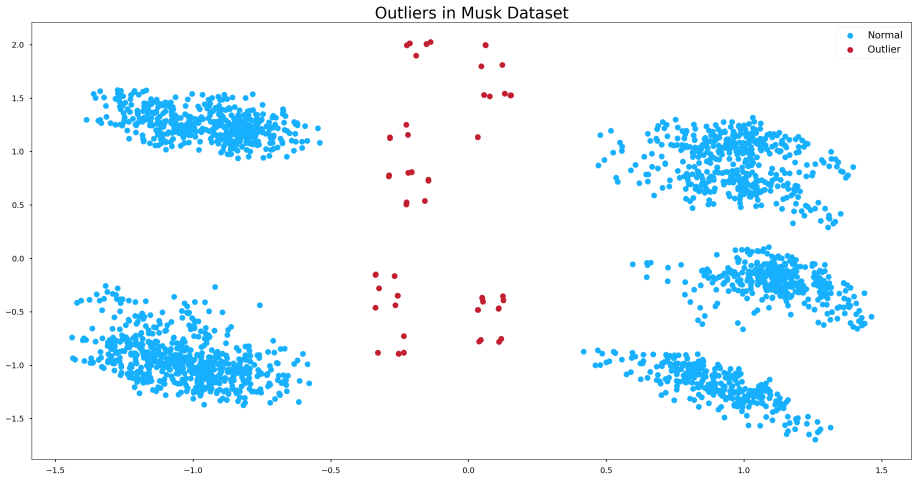


Figure 6.2 Outliers in the musk dataset.

6.3 Results

- **MNIST Trials for 0**

Table 6.1 Training results for the digit 0.

Algorithms	AUC	$F1$	Contingency table
Simple Autoencoder	0.97	0.31	5692 231 5 54
Order Statistics Scoring	1.0	0.82	5897 26 0 59
Order Statistics with Early Stopping	1.0	0.83	5898 25 0 59

Table 6.2 Test results for the digit 0.

Algorithms	AUC	$F1$	Precision	Recall	Contingency table
Simple Autoencoder	0.97	0.35	0.35	0.35	947 33 0 9
Order Statistics Scoring	0.99	0.48	0.48	0.48	973 7 4 5
Order Statistics with Early Stopping	0.99	0.42	0.42	0.42	970 10 4 5

- **MNIST Trials for 1**

Table 6.3 Training results for the digit 1.

Algorithms	AUC	$F1$	Contingency table
Simple Autoencoder	0.99	0.67	6688 54 6 2
Order Statistics Scoring	1.0	0.88	6723 19 0 68
Order Statistics with Early Stopping	1.0	0.87	6721 21 0 68

Table 6.4 Test results for the digit 1.

Algorithms	AUC	$F1$	Precision	Recall	Contingency table
Simple Autoencoder	0.99	0.81	0.81	0.81	1130 5 0 11
Order Statistics Scoring	0.99	0.82	0.82	0.82	1133 2 2 9
Order Statistics with Early Stopping	0.99	0.83	0.83	0.83	1132 3 1 10

- **MNIST Trials for 2**

Table 6.5 Training results for the digit 2.

Algorithms	AUC	$F1$	Contingency table
Simple Autoencoder	0.93	0.20	5605 353 14 46
Order Statistics Scoring	1.0	0.81	5929 29 0 60
Order Statistics with Early Stopping	1.0	0.8	5929 29 0 60

Table 6.6 Test results for the digit 2.

Algorithms	AUC	$F1$	Precision	Recall	Contingency table
Simple Autoencoder	0.93	0.21	0.21	0.21	972 60 2 8
Order Statistics Scoring	0.99	0.52	0.52	0.52	1025 7 4 6
Order Statistics with Early Stopping	0.96	0.48	0.48	0.48	1023 9 4 6

- **MNIST Trials for 3**

Table 6.7 Training results for the digit 3.

Algorithms	AUC	$F1$	Contingency table
Simple Autoencoder	0.95	0.22	5886 245 23 38
Order Statistics Scoring	1.0	0.75	6091 40 0 61
Order Statistics with Early Stopping	1.0	0.75	6090 41 0 61

Table 6.8 Test results for the digit 3.

Algorithms	AUC	$F1$	Precision	Recall	Contingency table
Simple Autoencoder	0.96	0.22	0.22	0.22	965 45 3 7
Order Statistics Scoring	0.95	0.25	0.25	0.25	999 11 7 3
Order Statistics with Early Stopping	0.95	0.23	0.23	0.23	997 13 7 3

- **MNIST Trials for 4**

Table 6.9 Training results for the digit 4.

Algorithms	AUC	$F1$	Contingency table
Simple Autoencoder	0.94	0.4	5731 11 17 42
Order Statistics Scoring	1.0	0.79	5810 32 0 59
Order Statistics with Early Stopping	1.0	0.8	5813 29 0 59

Table 6.10 Test results for the digit 4.

Algorithms	AUC	$F1$	Precision	Recall	Contingency table
Simple Autoencoder	0.89	0.4	0.4	0.4	967 15 3 6
Order Statistics Scoring	0.95	0.57	0.57	0.57	976 6 3 6
Order Statistics with Early Stopping	0.95	0.5	0.5	0.5	973 9 3 6

- **MNIST Trials for 5**

Table 6.11 Training results for the digit 5.

Algorithms	AUC	$F1$	Contingency table
Simple Autoencoder	0.96	0.15	4873 548 6 48
Order Statistics Scoring	1.0	0.8	5394 27 0 54
Order Statistics with Early Stopping	1.0	0.81	5395 26 0 54

Table 6.12 Test results for the digit 5.

Algorithms	AUC	$F1$	Precision	Recall	Contingency table
Simple Autoencoder	0.96	0.16	0.16	0.16	809 83 1 8
Order Statistics Scoring	0.97	0.19	0.19	0.19	882 10 7 2
Order Statistics with Early Stopping	0.97	0.25	0.25	0.25	880 12 6 3

- **MNIST Trials for 6**

Table 6.13 Training results for the digit 6.

Algorithms	AUC	$F1$	Contingency table
Simple Autoencoder	0.99	0.24	5569 349 4 55
Order Statistics Scoring	1.0	0.83	5893 25 0 59
Order Statistics with Early Stopping	1.0	0.84	5895 23 0 59

Table 6.14 Test results for the digit 6.

Algorithms	AUC	$F1$	Precision	Recall	Contingency table
Simple Autoencoder	0.99	0.26	0.26	0.26	908 50 0 9
Order Statistics Scoring	0.98	0.57	0.57	0.57	952 6 3 6
Order Statistics with Early Stopping	0.98	0.52	0.52	0.52	950 8 3 6

- **MNIST Trials for 7**

Table 6.15 Training results for the digit 7.

Algorithms	AUC	$F1$	Contingency table
Simple Autoencoder	0.97	0.53	6216 49 23 40
Order Statistics Scoring	1.0	0.79	6231 34 0 63
Order Statistics with Early Stopping	1.0	0.8	6232 33 0 63

Table 6.16 Test results for the digit 7.

Algorithms	AUC	$F1$	Precision	Recall	Contingency table
Simple Autoencoder	0.95	0.42	0.42	0.42	1012 16 3 7
Order Statistics Scoring	0.96	0.64	0.64	0.64	1023 5 3 7
Order Statistics with Early Stopping	0.96	0.54	0.54	0.54	1019 9 3 7

- **MNIST Trials for 8**

Table 6.17 Training results for the digit 8.

Algorithms	AUC	$F1$	Contingency table
Simple Autoencoder	0.89	0.14	5393 453 20 39
Order Statistics Scoring	1.0	0.8	5822 29 0 59
Order Statistics with Early Stopping	1.0	0.79	5819 32 0 59

Table 6.18 Test results for the digit 8.

Algorithms	AUC	$F1$	Precision	Recall	Contingency table
Simple Autoencoder	0.9	0.19	0.19	0.19	921 53 3 6
Order Statistics Scoring	0.9	0.29	0.29	0.29	965 9 6 3
Order Statistics with Early Stopping	0.9	0.24	0.24	0.24	961 13 6 3

- **MNIST Trials for 9**

Table 6.19 Training results for the digit 9.

Algorithms	AUC	$F1$	Contingency table
Simple Autoencoder	0.96	0.25	5738 211 22 38
Order Statistics Scoring	1.0	0.77	5914 35 0 60
Order Statistics with Early Stopping	1.0	0.77	5914 35 0 60

Table 6.20 Test results for the digit 9.

Algorithms	AUC	$F1$	Precision	Recall	Contingency table
Simple Autoencoder	0.95	0.29	0.29	0.29	978 31 3 7
Order Statistics Scoring	0.96	0.36	0.36	0.36	1001 8 6 4
Order Statistics with Early Stopping	0.96	0.32	0.32	0.32	998 11 6 4

- **Trials with Musk**

Table 6.21 Training results for musk.

Algorithms	AUC	$F1$	Contingency table
Simple Autoencoder	1.0	1.0	2223 0 0 22
Order Statistics Scoring	1.0	0.55	2187 36 0 22
Order Statistics with Early Stopping	1.0	0.55	2187 36 0 22

Table 6.22 Test results for musk.

Algorithms	AUC	$F1$	Precision	Recall	Contingency table
Simple Autoencoder	0.5	0.02	0.02	0.02	0 742 0 7
Order Statistics Scoring	0.97	0.20	0.20	0.20	712 30 3 4
Order Statistics with Early Stopping	0.96	0.20	0.20	0.20	712 30 3 4

7
RESULTS AND DISCUSSION

In terms of testing our hypotheses about autoencoders, three points are significant. In the discussion about the shortcomings of autoencoders in section 3.3, we propounded that autoencoders are complex models which, if trained long enough, may generalize to outliers as well as normal data. Based on that conjecture, we would expect a simple autoencoder, trained without order statistics regularization, to overfit and thus have a greater disparity between the training and the test sets. Moreover, we postulated that due to the noticeable difference between outliers and normal data early on in the training, outliers would have a leverage effect. Therefore, we would expect greater false negative errors for the simple autoencoder.

The cut-off point is also of interest, order-statistics-based models work in a completely unsupervised manner while the simple autoencoder needs the percentage of outliers in the data as a priori. In light of this fact, we would expect to see better training results from the simple autoencoder. However, we see that across MNIST datasets the order-statistics-based algorithms have a higher success rate both in training and in test sets, which suggests the augmented loss function is helping to reduce the effect of outliers during training.

We see in Table 6.21 for the musk dataset that there is a perfect separation of outliers in the training set. However, ostensible success in the training set comes at a great cost in generalization. It is clear from Table 6.22 that the model has overfitted. In comparison, the order-statistics-based models have a more balanced result with similar rate of success between the training and the test data.

Despite the clear example of overfitting in the musk dataset, it is difficult to corroborate our supposition about the reduced disparity between train and test sets when looking at MNIST. For digits 9 (Table 6.19, Table 6.20) and 8 (Table 6.17, Table 6.18) in particular, the disparity is greater for the order-statistics-based methods. We note that order-statistics-based methods remain more successful on both train and test sets, which is a testament to our statistics adaptability. The cause of the disparity could

potentially be the relative differences in success. It is also possible that the threshold calculated by our statistic may not generalize well to the test sets.

Congruent with our hypothesis concerning the leverage effect of outliers, in Table 6.6, Table 6.12, and Table 6.14 we see greater number of false negatives in the simple autoencoder. The number of false negatives is a particularly important measure because it indicates the potential blind spots of an outlier detection system. Order-statistics-based autoencoders, on the other hand, have comparably less false negatives.

It is relatively more difficult to test the effects of the early stopping algorithm, as it is only used jointly with the order statistics scoring method. Nevertheless, we would anticipate a decrease in the number of epochs required in training, without any great forfeit of model success. Furthermore, we expect κ-threshold values to decrease more regularly in successful training sessions. When we compare the success of order statistics scoring autoencoder with the fully fledged algorithm using early stopping, we see that the order statistics scoring is generally more successful with small differences in both the musk dataset (see Table 6.22) and MNIST (see Table 6.2, Table 6.16, and Table 6.18). This suggests that the κ based early stopping algorithm may need to be more dynamically adjusted during training.

Plots of the original κ-threshold values for each model are given in Appendix A. The moving block average may smooth too much with the block length set to 3, as we see the training stopped in a very early and unstable epoch in Figure A.7, Figure A.31, and in Figure A.43. We may also need to investigate the stopping condition which is based on the stability of the past 10 epochs.

Another shortcoming we see from our experiments is that the difference between train and test sets is greater for $F1$ metrics compared to AUC metrics. This suggests that our κ-threshold may be suboptimal. We may need to select the κ-threshold value more carefully during training, perhaps using methods such as cross-validation.

Merrill and Iskandarian [39] conducted similar experiments on MNIST dataset using a similar autoencoder architecture. However, their experiments were not extensive. In their trials, they picked a single digit to represent the normal instances called background in Table 7.1, and sampled from one or the rest of the digits to represent the outliers. Similar to our experiments, they undersampled outliers by a percentage value of α. They only reported results on the training set as their method do not work on test sets. They did not report $F1$ metric but used AUC and average precision (AP) instead.

Table 7.1 AUC (left) and AP (right) results on MNIST provided by Merrill and Iskandarian [39].

background	anomaly	N	α	AUC | AP
Digit 0	Digit 2	3665	2%	**0.996** | **0.904**
	Mix	3665	2%	**0.992** | **0.790**
Digit 9	Digit 6	7089	2%	**0.993** | **0.698**
Digit 2	Digit 5	7117	2%	**0.990** | **0.777**

Another study conducted by An and Cho, studied the MNIST dataset using a variational autoencoder [29]. Instead of sampling, they created 10 datasets by labelling a digit as outlier in each one. As a result, each dataset had an outlier rate of approximately 10%. We share their results in Table 7.2. It is difficult to do a case-by-case comparison as the rate of outliers between our studies are different.

Table 7.2 AUC and $F1$ results on MNIST test sets provided by An and Cho [29].

Anomaly digit	AUC	$F1$
0	0.517	0.537
1	0.063	0.205
2	0.644	0.598
3	0.251	0.332
4	0.337	0.381
5	0.325	0.427
6	0.432	0.433
7	0.148	0.212
8	0.499	0.49
9	0.104	0.21

We contend that our order-statistics-based loss function and scoring method improves the success of autoencoders and makes the models more robust. The κ-threshold based early stopping algorithm offers a trade-off between model success and the number of epochs in training. However, as autoencoders are not particularly expensive to train, it does not provide an attractive alternative.

For future work, it may be possible to increase model success by improving on smoothing and stopping condition of early stopping. In particular, choosing the number of blocks for smoothing and stopping dynamically, rather than leaving them as hyperparameters, would be an important step. Another shortcoming of our method

concerns the cut-off point. As mentioned before, selecting the κ-threshold using a single sample may cause problems for generalizability. We recommend further experimentation with methods such as cross-validation.

As we have stated at the beginning of chapter 2, our statistic is not just compatible with autoencoders but with any outlier detection algorithm producing outlier scores. We are interested in seeing its use in other outlier detection methods such as GANs, SVMs, or in fields related to outlier detection such as change point detection.

REFERENCES

[1] D. M. Hawkins, Identification of Outliers. Springer, 1980, vol. 11.

[2] J. P. Edelen, N. M. Cook, "Anomaly detection in particle accelerators using autoencoders," arXiv preprint arXiv:2112.07793, 2021.

[3] H. Chan, S. Cheung, V. A. Villar, S. Ho, "A convolutional autoencoder-based pipeline for anomaly detection and classification of periodic variables," arXiv preprint arXiv:2111.13828, 2021.

[4] V. Snorovikhina, A. Zaytsev, "Unsupervised anomaly detection for discrete sequence healthcare data," in International Conference on Analysis of Images, Social Networks and Texts, Springer, 2020, pp. 391–403.

[5] D. Snyder, "On-line intrusion detection using sequences of system calls," Journal of Computer Security, vol. 6, pp. 151–180, 1998.

[6] V. Kumar, "Parallel and distributed computing for cybersecurity," IEEE Distributed Systems Online, vol. 6, no. 10, 2005.

[7] E. Aleskerov, B. Freisleben, B. Rao, "Cardwatch: A neural network based database mining system for credit card fraud detection," in Proceedings of the IEEE/IAFE 1997 computational intelligence for financial engineering (CIFEr), IEEE, 1997, pp. 220–226.

[8] S. Basu, M. Meckesheimer, "Automatic outlier detection for time series: An application to sensor data," Knowledge and Information Systems, vol. 11, no. 2, pp. 137–154, 2007.

[9] R. Fujimaki, T. Yairi, K. Machida, "An approach to spacecraft anomaly detection problem using kernel feature space," in Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, 2005, pp. 401–410.

[10] L. De Haan, A. Ferreira, A. Ferreira, Extreme Value Theory: An Introduction. Springer, 2006, vol. 21.

[11] C. A. Charu, Outlier Analysis. springer, 2019.

[12] G. Pang, C. Shen, L. Cao, A. V. D. Hengel, "Deep learning for anomaly detection," ACM Computing Surveys, vol. 54, no. 2, pp. 1–38, Apr. 2021, ISSN: 1557–7341. DOI: 10.1145/3439950. [Online]. Available: http://dx.doi.org/10.1145/3439950.

[13] C. Zhou, R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 665–674.

[14] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, A. Lerchner, "Understanding disentangling in beta-vae," arXiv preprint arXiv:1804.03599, 2018.

[15] A. Z. Balcıoğlu, *Meta-outliers*, version 1.0.0, Jun. 2022. [Online]. Available: https://github.com/Selozhd/meta-outliers.

[16] A. Z. Balcıoğlu, O. Gürerk, "On a notion of outliers based on ratios of order statistics," 2022. DOI: 10.48550/ARXIV.2207.13068. [Online]. Available: https://arxiv.org/abs/2207.13068.

[17] F. Y. Edgeworth, "On discordant observations," The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, vol. 23, no. 143, pp. 364–375, 1887.

[18] V. Barnett, T. Lewis, Outliers in Statistical Data. Wiley, 1978.

[19] Z. He, X. Xu, S. Deng, "Discovering cluster-based local outliers," Pattern Recognition Letters, vol. 24, no. 9-10, pp. 1641–1650, 2003.

[20] S. Kibish, *A note about finding anomalies*, [Online; accessed May 28, 2022], 2018. [Online]. Available: https://miro.medium.com/max/1400/1*cY9PvTtR_B2NdJEo0WOUvA.png.

[21] X. Song, M. Wu, C. Jermaine, S. Ranka, "Conditional anomaly detection," IEEE Transactions on knowledge and Data Engineering, vol. 19, no. 5, pp. 631–645, 2007.

[22] P. Sun, S. Chawla, B. Arunasalam, "Mining for outliers in sequential databases," in Proceedings of the 2006 SIAM international conference on data mining, SIAM, 2006, pp. 94–105.

[23] C. C. Noble, D. J. Cook, "Graph-based anomaly detection," in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, pp. 631–636.

[24] V. Chandola, A. Banerjee, V. Kumar, "Anomaly detection: A survey," ACM computing surveys (CSUR), vol. 41, no. 3, pp. 1–58, 2009.

[25] T. Kieu, B. Yang, C. Guo, C. S. Jensen, "Outlier detection for time series with recurrent autoencoder ensembles," in Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, "IJCAI-19", International Joint Conferences on Artificial Intelligence Organization, Jul. 2019, pp. 2725–2732. DOI: 10.24963/ijcai.2019/378. [Online]. Available: https://doi.org/10.24963/ijcai.2019/378.

[26] W. Lu, Y. Cheng, C. Xiao, S. Chang, S. Huang, B. Liang, T. Huang, "Unsupervised sequential outlier detection with deep architectures," English (US), IEEE Transactions on Image Processing, vol. 26, no. 9, pp. 4321–4330, Sep. 2017, ISSN: 1057–7149. DOI: 10.1109/TIP.2017.2713048.

[27] D. Xu, E. Ricci, Y. Yan, J. Song, N. Sebe, "Learning deep representations of appearance and motion for anomalous event detection," arXiv preprint arXiv:1510.01553, 2015.

[28] K. Balakrishnan, Exponential Distribution: Theory, Methods and Applications. Routledge, 2019.

[29] J. An, S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," Special Lecture on IE, vol. 2, no. 1, pp. 1–18, 2015.

[30] Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang, X. He, "Generative adversarial active learning for unsupervised outlier detection," 2018. DOI: 10.48550/ARXIV.1809.10816. [Online]. Available: https://arxiv.org/abs/1809.10816.

[31] Y. LeCun, C. Cortes, C. Burges, "Mnist handwritten digit database," ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist, vol. 2, 2010.

[32] D. Dua, C. Graff, *Uci machine learning repository*, 2017. [Online]. Available: http://archive.ics.uci.edu/ml.

[33] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, G. H. Chen, "Ecod: Unsupervised outlier detection using empirical cumulative distribution functions," arXiv preprint arXiv:2201.00382, 2022.

[34] R. Samrin, D. Vasumathi, "Review on anomaly based network intrusion detection system," in 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), IEEE, 2017, pp. 141–147.

[35] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, G. Bontempi, "Learned lessons in credit card fraud detection from a practitioner perspective," Expert systems with applications, vol. 41, no. 10, pp. 4915–4928, 2014.

[36] J. Sato, Y. Suzuki, T. Wataya, D. Nishigaki, K. Kita, K. Yamagata, N. Tomiyama, S. Kido, "Anatomy-aware self-supervised learning for anomaly detection in chest radiographs," 2022. DOI: 10.48550/ARXIV.2205.04282. [Online]. Available: https://arxiv.org/abs/2205.04282.

[37] J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya, *et al.*, "Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison," in Proceedings of the AAAI conference on artificial intelligence, vol. 33, 2019, pp. 590–597.

[38] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[39] N. Merrill, A. Eskandarian, "Modified autoencoder training and scoring for robust unsupervised anomaly detection in deep learning," IEEE Access, vol. 8, pp. 101 824–101 833, 2020. DOI: 10.1109/ACCESS.2020.2997327.

[40] V. Škvára, J. Francå, M. Zorek, T. Pevný, V. Šmıédl, "Comparison of anomaly detectors: Context matters," IEEE Transactions on Neural Networks and Learning Systems, 2021.

[41] W. Sultani, C. Chen, 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.

[42] J. J. Davis, A. J. Clark, "Data preprocessing for anomaly based network intrusion detection: A review," computers & security, vol. 30, no. 6–7, pp. 353–375, 2011.

[43] Y. LeCun, Y. Bengio, G. Hinton, "Deep learning," nature, vol. 521, no. 7553, pp. 436–444, 2015.

[44] H. Abdi, L. J. Williams, "Principal component analysis," Wiley interdisciplinary reviews: computational statistics, vol. 2, no. 4, pp. 433–459, 2010.

[45] Y. Fujikoshi, V. Ulyanov, R. Shimizu, Multivariate Statistics: High-Dimensional and Large-Sample Approximations, ser. Wiley Series in Probability and Statistics. Wiley, 2011, ISBN: 9780470539866.

[46] J. Jeremy, *Introduction to autoencoders*, [Online; accessed April 3, 2022], 2018. [Online]. Available: https://www.jeremyjordan.me/content/images/2018/03/Screen-Shot-2018-03-07-at-8.24.37-AM.png.

[47] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, L. Chang, "A novel anomaly detection scheme based on principal component classifier," Miami Univ Coral Gables Fl Dept of Electrical and Computer Engineering, Tech. Rep., 2003.

[48] H. A. David, H. N. Nagaraja, Order Statistics. John Wiley & Sons, 2003.

[49] J. Pickands III, "Statistical inference using extreme order statistics," the Annals of Statistics, pp. 119–131, 1975.

[50] U. Gather, "Robust estimation of the mean of the exponential distribution in outlier situations," Communications in Statistics-Theory and Methods, vol. 15, no. 8, pp. 2323–2345, 1986.

[51] P. J. Rousseeuw, A. M. Leroy, Robust Regression and Outlier Detection. John wiley & sons, 2005.

[52] P. J. Huber, E. Ronchetti, Robust Statistics. Wiley, 2009.

[53] L. B. Klebanov, A. V. Kakosyan, A. Karlova, "Outliers, the law of large numbers, index of stability and heavy tails," arXiv preprint arXiv:1612.09265, 2016.

[54] M. Antunes, D. Gomes, R. L. Aguiar, "Knee/elbow estimation based on first derivative threshold," in 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService), 2018, pp. 237–240. DOI: 10.1109/BigDataService.2018.00042.

[55] V. Satopaa, J. Albrecht, D. Irwin, B. Raghavan, "Finding a "kneedle" in a haystack: Detecting knee points in system behavior," in 2011 31st International Conference on Distributed Computing Systems Workshops, 2011, pp. 166–171. DOI: 10.1109/ICDCSW.2011.20.

[56] X. Tolsa, "Principal values for the cauchy integral and rectifiability," Proceedings of the American Mathematical Society, vol. 128, no. 7, pp. 2111–2119, 2000.

[57] J. R. Albrecht, C. Tuttle, A. C. Snoeren, A. Vahdat, "Loose synchronization for large-scale networked systems.," in USENIX Annual Technical Conference, General Track, 2006, pp. 301–314.

[58] T. Ridler, S. Calvard, *et al.*, "Picture thresholding using an iterative selection method," IEEE Transactions on Systems, Man, and Cybernetics, vol. 8, no. 8, pp. 630–632, 1978.

[59] S. Rayana, *Odds library*, 2016. [Online]. Available: http://odds.cs.stonybrook.edu.

[60] J. Cazala, *Mnist*, [Online; accessed May 12, 2022], 2017. [Online]. Available: https://github.com/cazala/mnist.

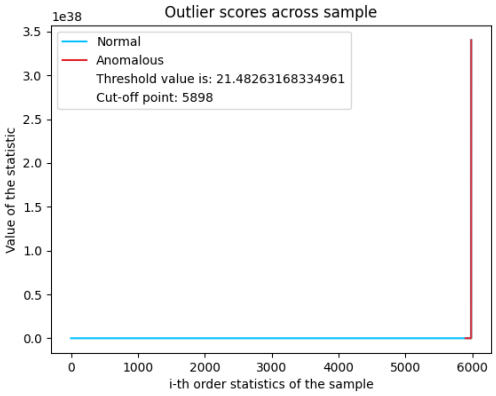
[61] G. B. Cavallari, L. S. Ribeiro, M. A. Ponti, “Unsupervised representation learning using convolutional and stacked auto-encoders: A domain and cross-domain feature space analysis,” in 2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), IEEE, 2018, pp. 440–446.

[62] D. P. Kingma, J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

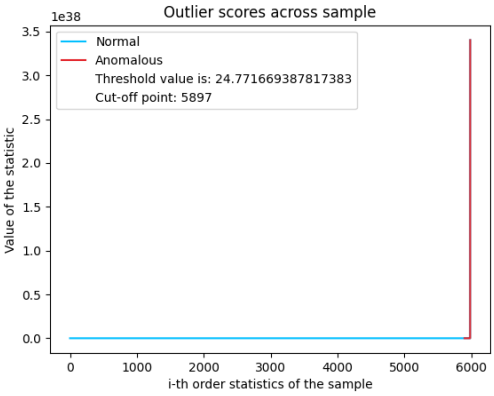
[63] Y. Zhao, Z. Nasrullah, Z. Li, “Pyod: A python toolbox for scalable outlier detection,” Journal of Machine Learning Research, vol. 20, no. 96, pp. 1–7, 2019. [Online]. Available: http://jmlr.org/papers/v20/19-011.html.

A MISCELLANEOUS PLOTS

A.1 Extra plots from MNIST digit 0

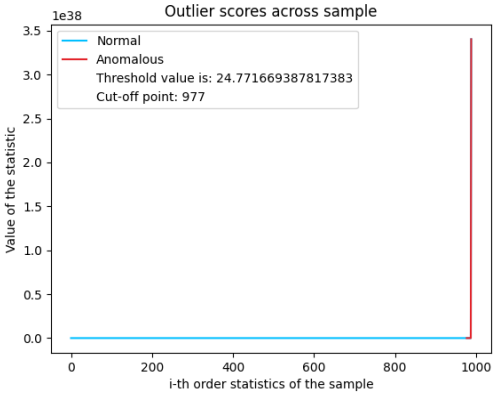


Order Statistics with Early Stopping

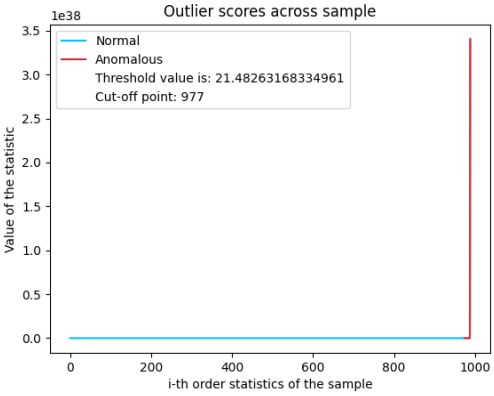


Order Statistics Scoring

Figure A.1 Outlier plots for the train set of MNIST digit 0.



Order Statistics with Early Stopping



Order Statistics Scoring

Figure A.2 Outlier plots for the test set of MNIST digit 0.

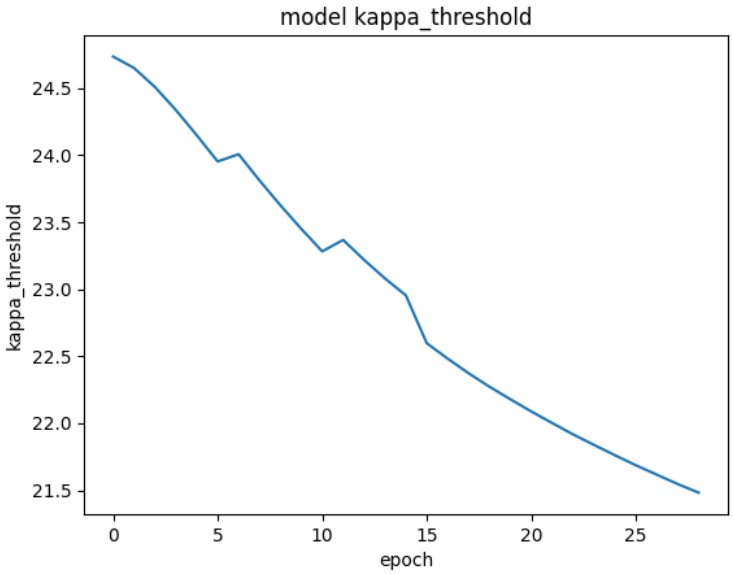
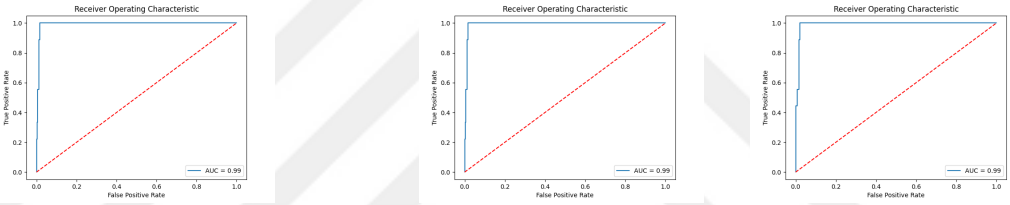


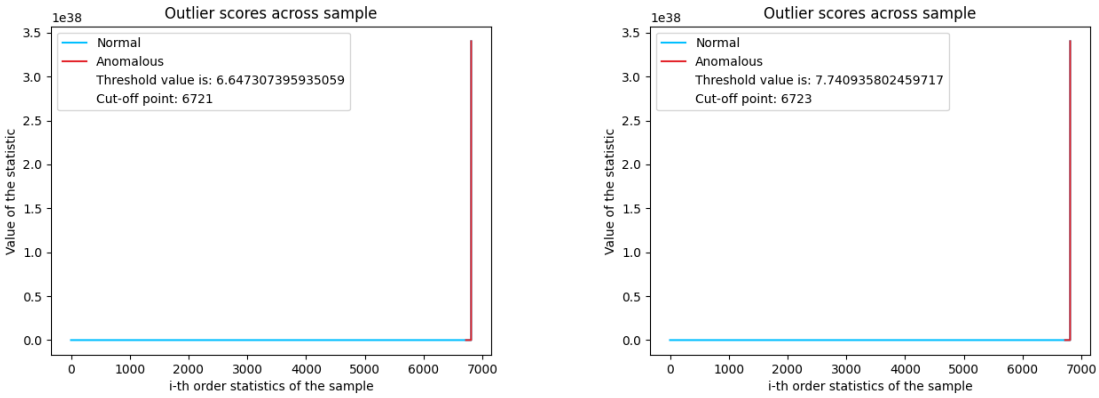
Figure A.3 κ-threshold values during training of the algorithm for MNIST digit 0.



Order Statistics with Early Stopping Order Statistics Scoring Simple Autoencoder

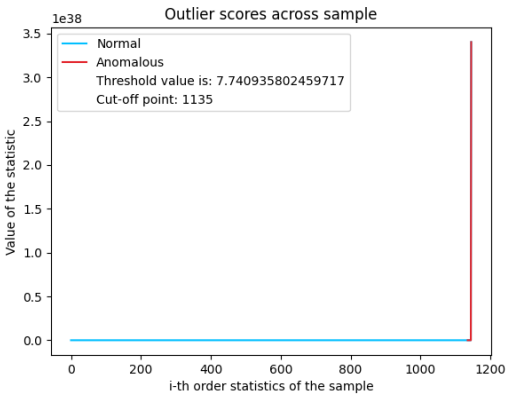
Figure A.4 ROC curves for the test set of MNIST digit 0.

A.2 Extra plots from MNIST digit 1

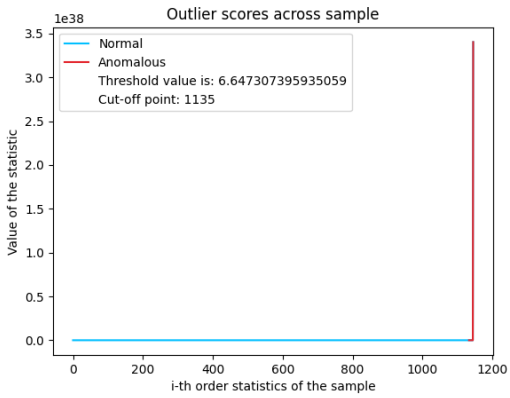


Order Statistics with Early Stopping Order Statistics Scoring

Figure A.5 Outlier plots for the train set of MNIST digit 1.



Order Statistics with Early Stopping



Order Statistics Scoring

Figure A.6 Outlier plots for the test set of MNIST digit 1.

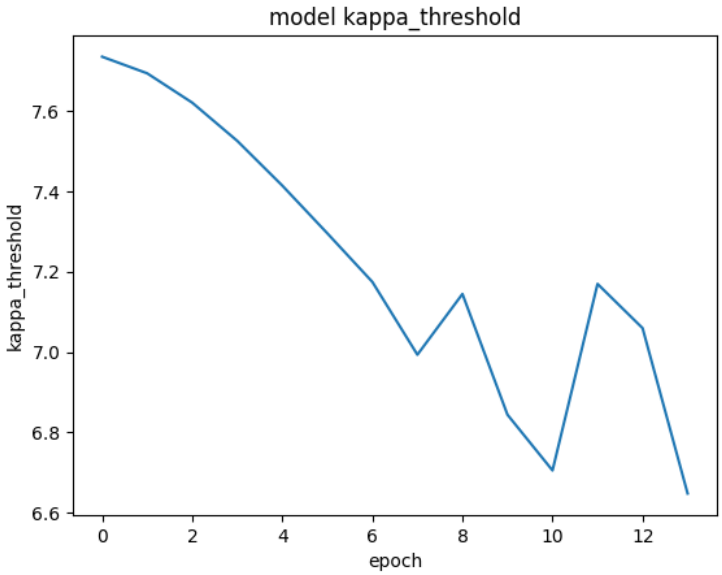
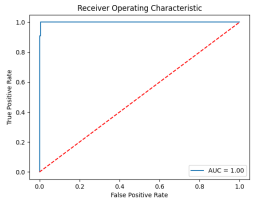
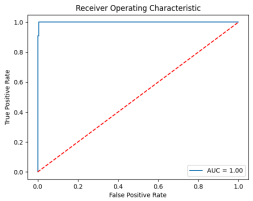


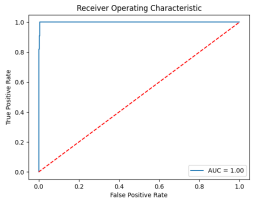
Figure A.7 κ-threshold values during training of the algorithm for MNIST digit 1.



Order Statistics with Early Stopping



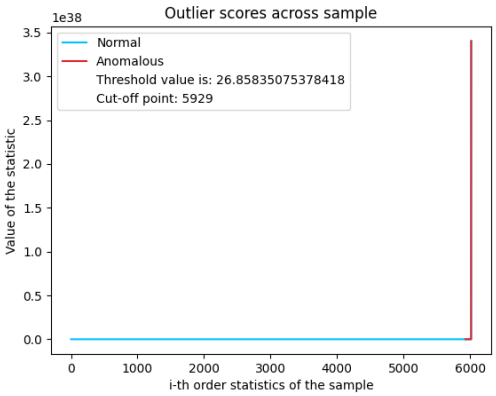
Order Statistics Scoring



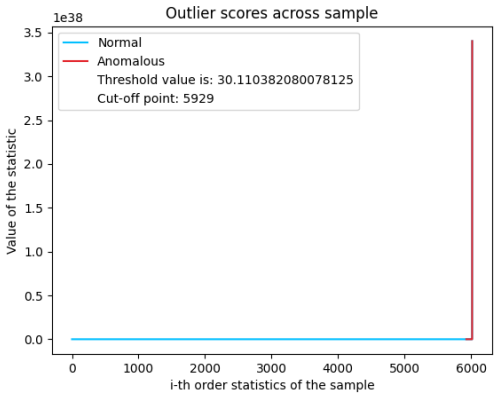
Simple Autoencoder

Figure A.8 ROC curves for the test set of MNIST digit 1.

A.3 Extra plots from MNIST digit 2

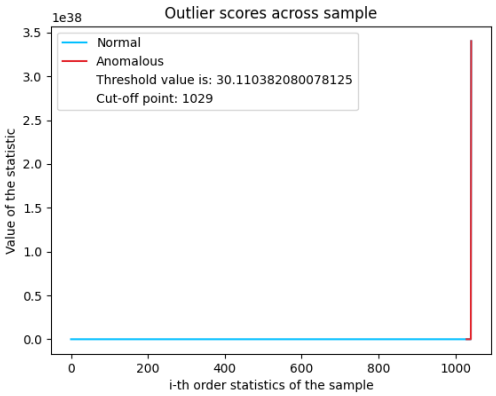


Order Statistics with Early Stopping

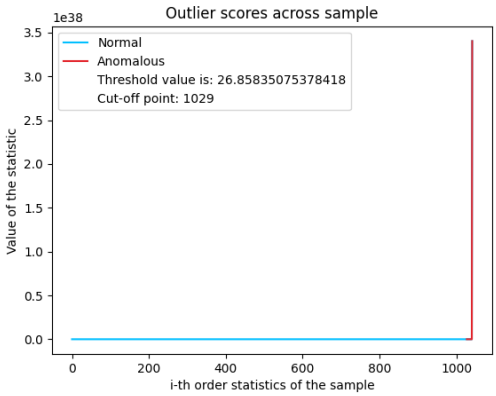


Order Statistics Scoring

Figure A.9 Outlier plots for the train set of MNIST digit 2.



Order Statistics with Early Stopping



Order Statistics Scoring

Figure A.10 Outlier plots for the test set of MNIST digit 2.

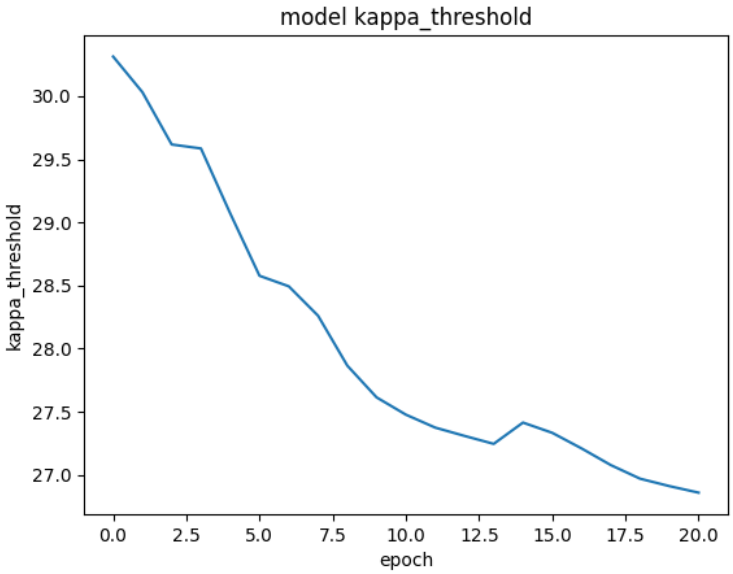
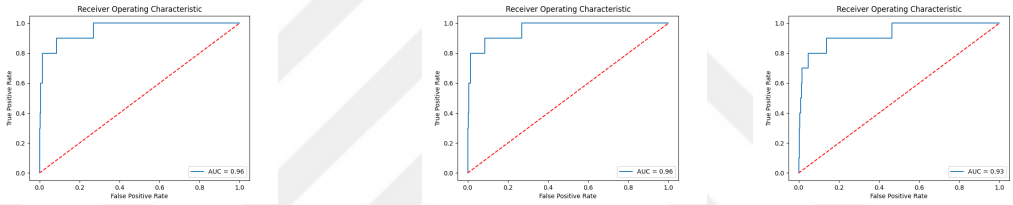


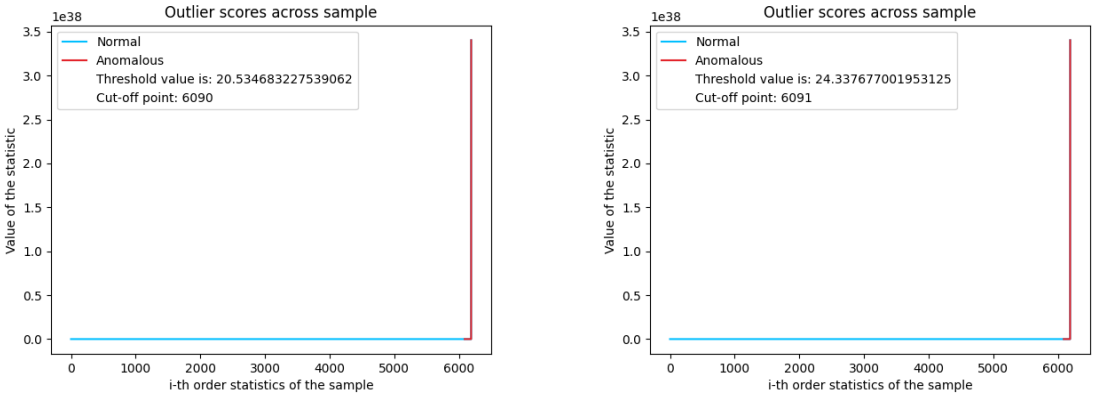
Figure A.11 κ-threshold values during training of the algorithm for MNIST digit 2.



Order Statistics with Early Stopping Order Statistics Scoring Simple Autoencoder

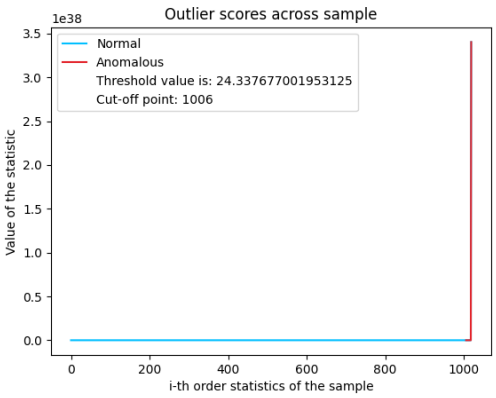
Figure A.12 ROC curves for the test set of MNIST digit 2.

A.4 Extra plots from MNIST digit 3

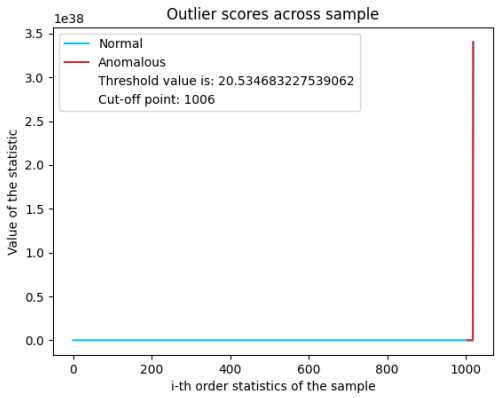


Order Statistics with Early Stopping Order Statistics Scoring

Figure A.13 Outlier plots for the train set of MNIST digit 3.



Order Statistics with Early Stopping



Order Statistics Scoring

Figure A.14 Outlier plots for the test set of MNIST digit 3.

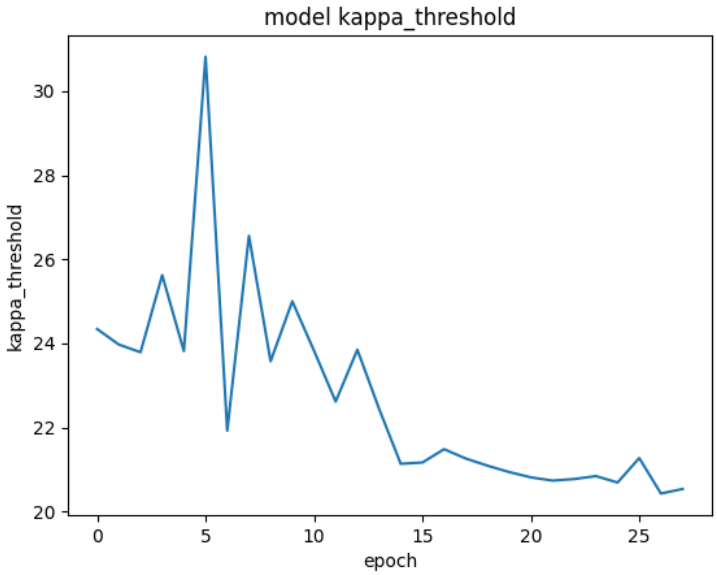
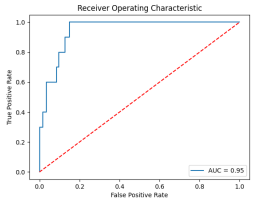
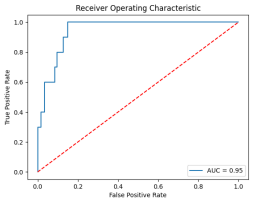


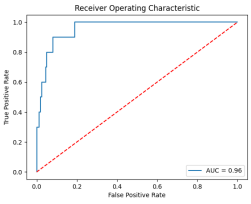
Figure A.15 κ-threshold values during training of the algorithm for MNIST digit 3.



Order Statistics with Early Stopping



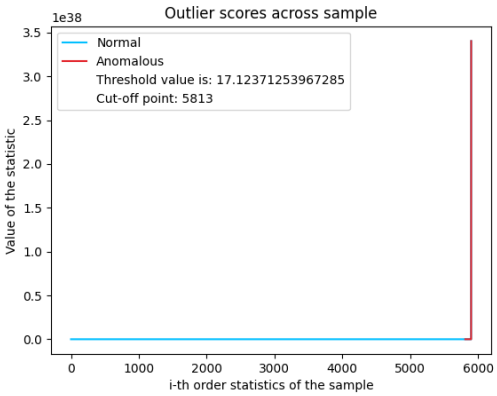
Order Statistics Scoring



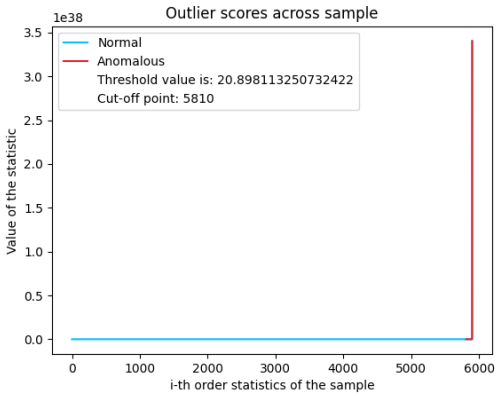
Simple Autoencoder

Figure A.16 ROC curves for the test set of MNIST digit 3.

A.5 Extra plots from MNIST digit 4

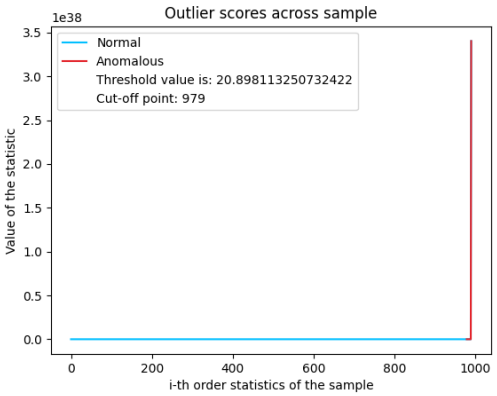


Order Statistics with Early Stopping

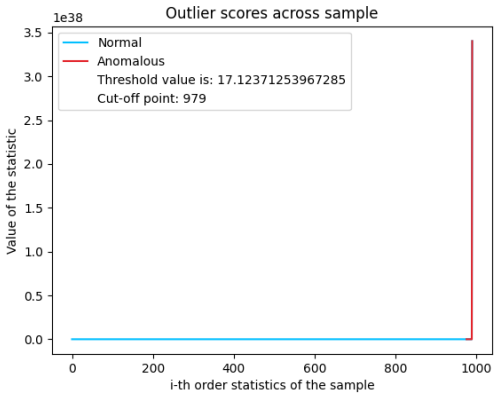


Order Statistics Scoring

Figure A.17 Outlier plots for the train set of MNIST digit 4.



Order Statistics with Early Stopping



Order Statistics Scoring

Figure A.18 Outlier plots for the test set of MNIST digit 4.

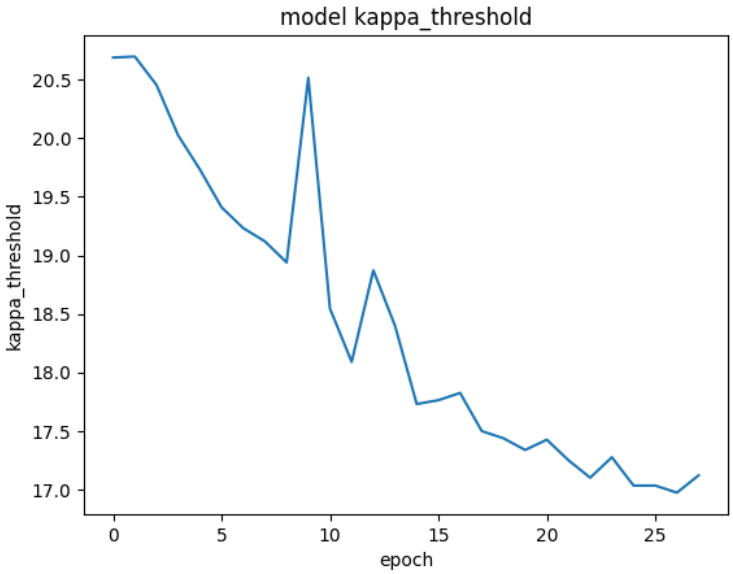
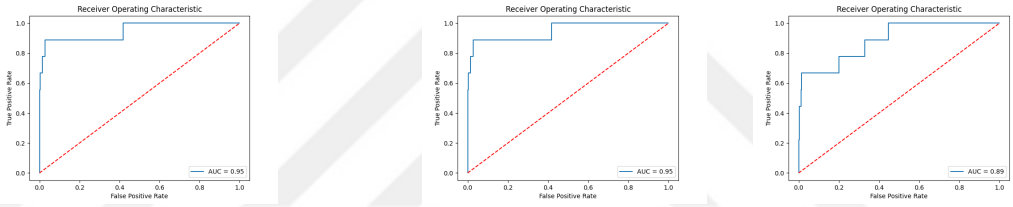


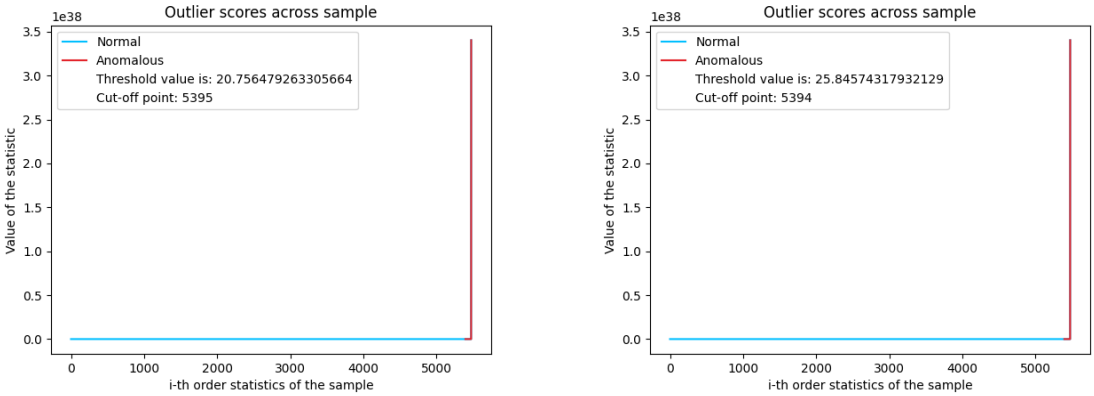
Figure A.19 κ-threshold values during training of the algorithm for MNIST digit 4.



Order Statistics with Early Stopping Order Statistics Scoring Simple Autoencoder

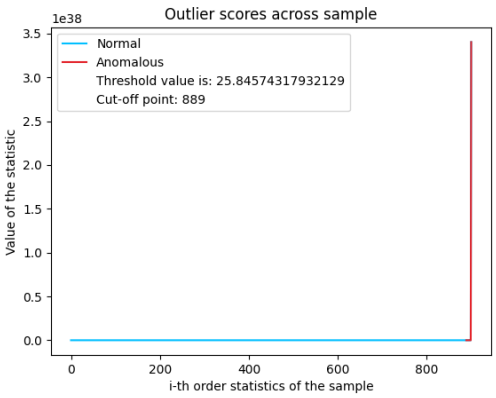
Figure A.20 ROC curves for the test set of MNIST digit 4.

A.6 Extra plots from MNIST digit 5

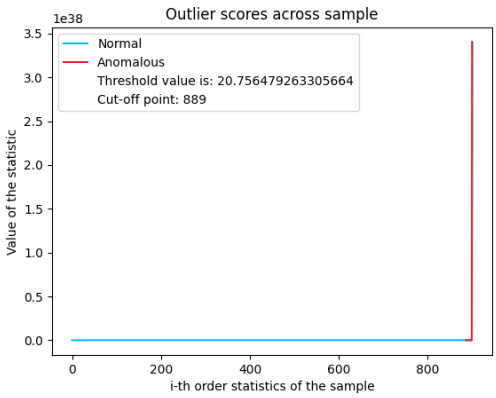


Order Statistics with Early Stopping Order Statistics Scoring

Figure A.21 Outlier plots for the train set of MNIST digit 5.



Order Statistics with Early Stopping



Order Statistics Scoring

Figure A.22 Outlier plots for the test set of MNIST digit 5.

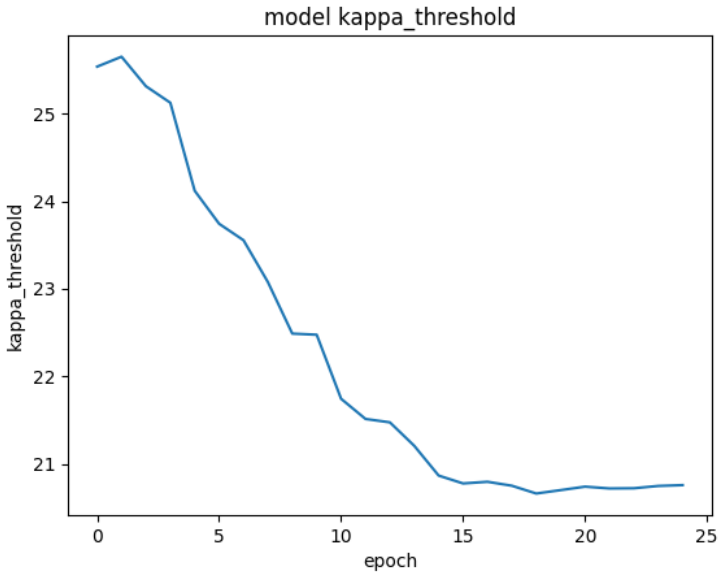
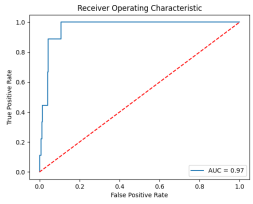
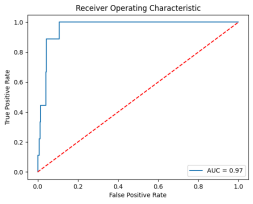


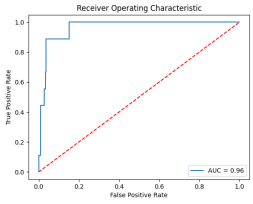
Figure A.23 κ-threshold values during training of the algorithm for MNIST digit 5.



Order Statistics with Early Stopping



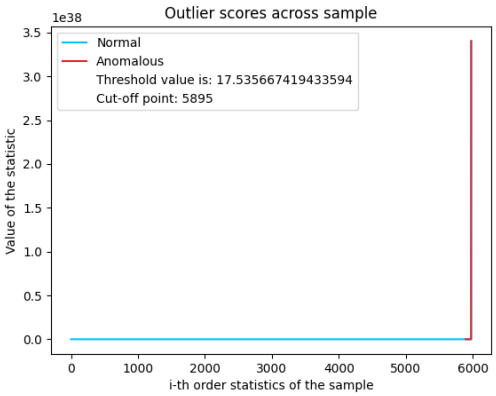
Order Statistics Scoring



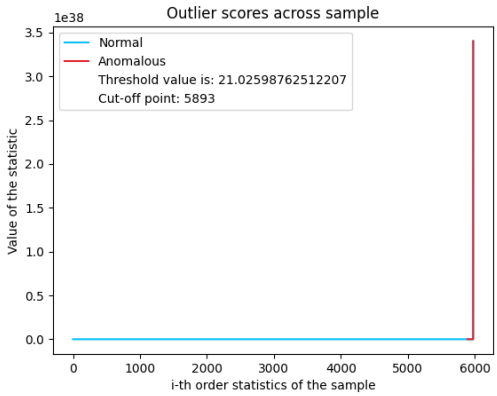
Simple Autoencoder

Figure A.24 ROC curves for the test set of MNIST digit 5.

A.7 Extra plots from MNIST digit 6

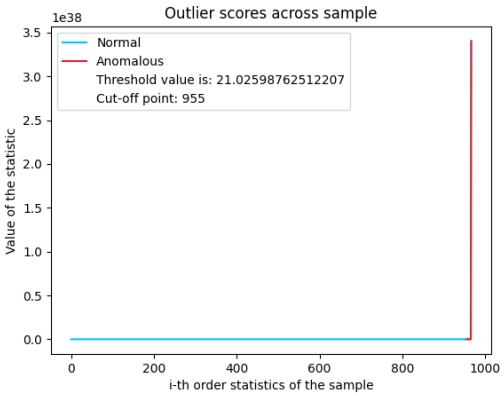


Order Statistics with Early Stopping

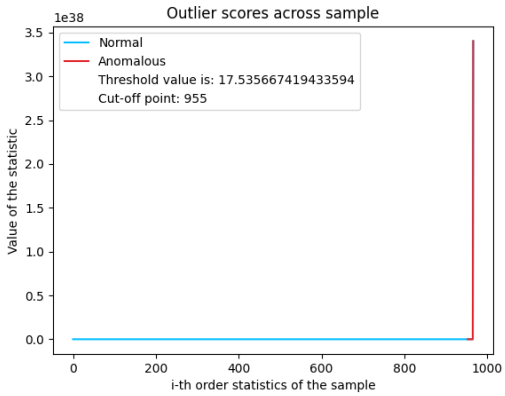


Order Statistics Scoring

Figure A.25 Outlier plots for the train set of MNIST digit 6.



Order Statistics with Early Stopping



Order Statistics Scoring

Figure A.26 Outlier plots for the test set of MNIST digit 6.

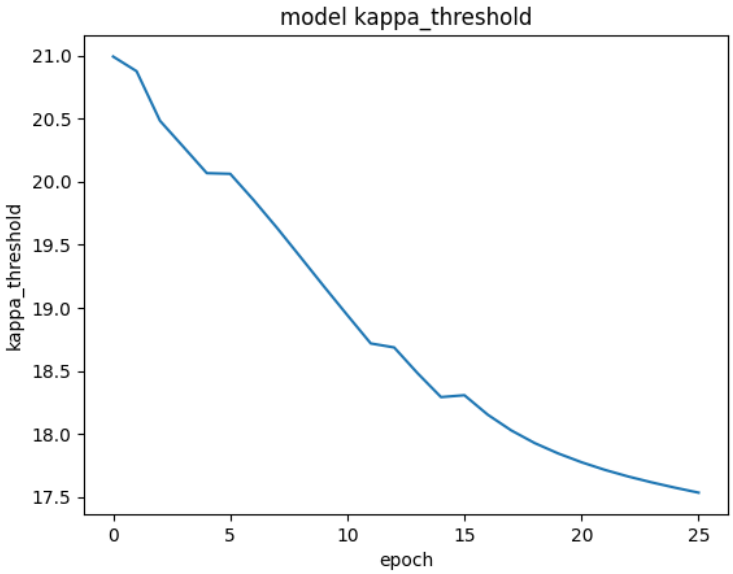
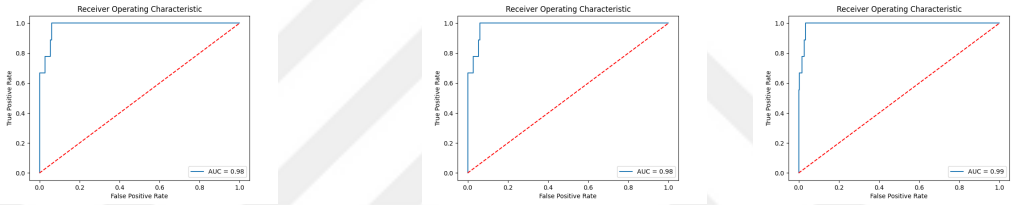


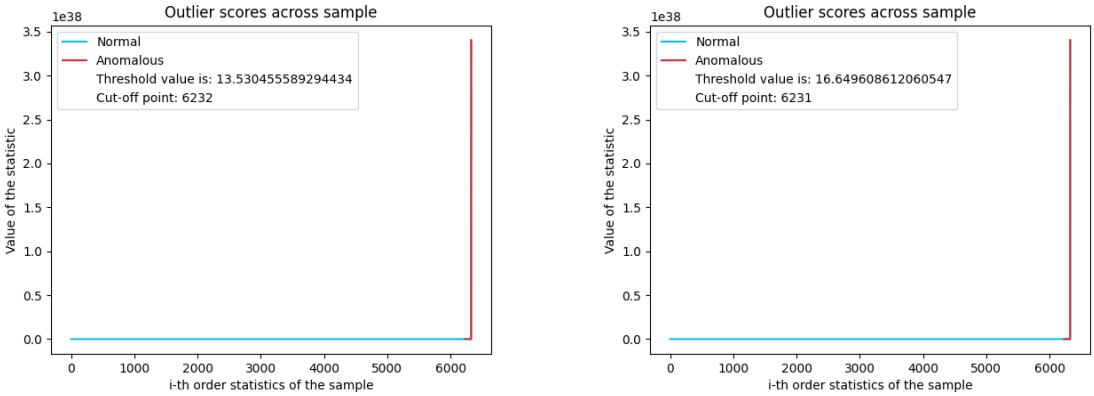
Figure A.27 κ-threshold values during training of the algorithm for MNIST digit 6.



Order Statistics with Early Stopping Order Statistics Scoring Simple Autoencoder

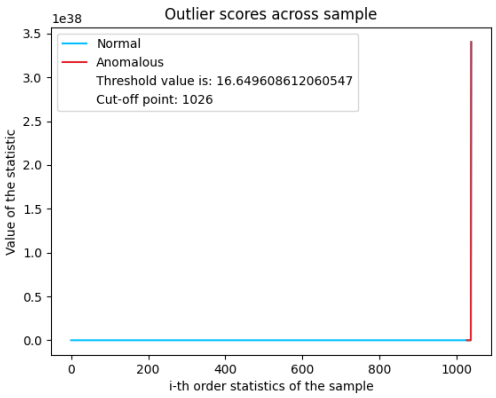
Figure A.28 ROC curves for the test set of MNIST digit 6.

A.8 Extra plots from MNIST digit 7

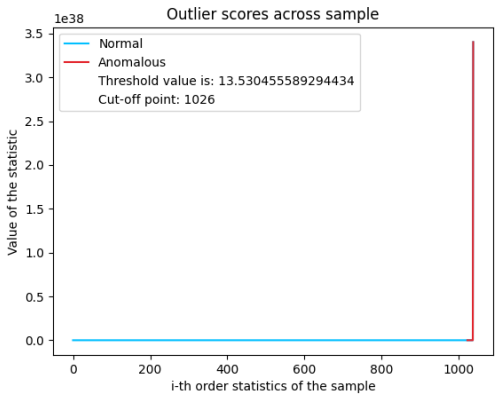


Order Statistics with Early Stopping Order Statistics Scoring

Figure A.29 Outlier plots for the train set of MNIST digit 7.



Order Statistics with Early Stopping



Order Statistics Scoring

Figure A.30 Outlier plots for the test set of MNIST digit 7.

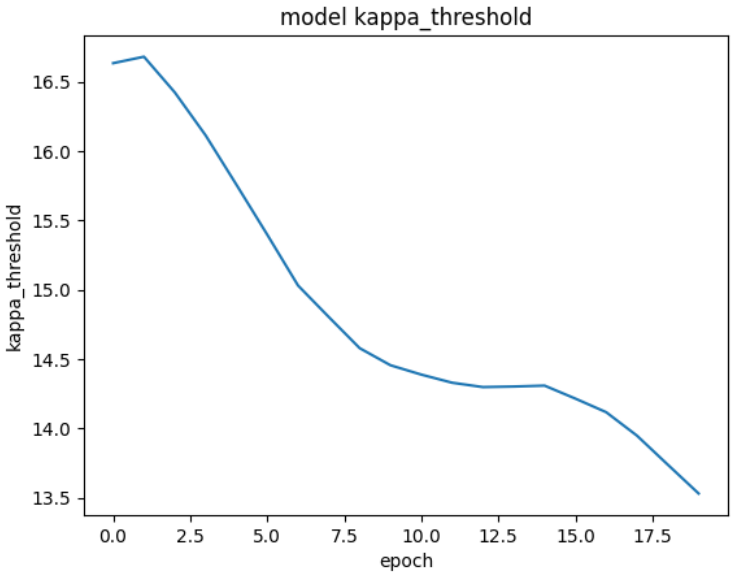
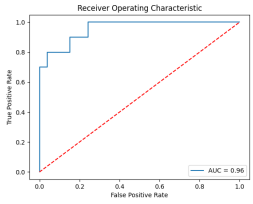
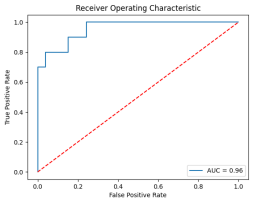


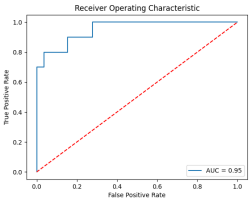
Figure A.31 κ-threshold values during training of the algorithm for MNIST digit 7.



Order Statistics with Early Stopping



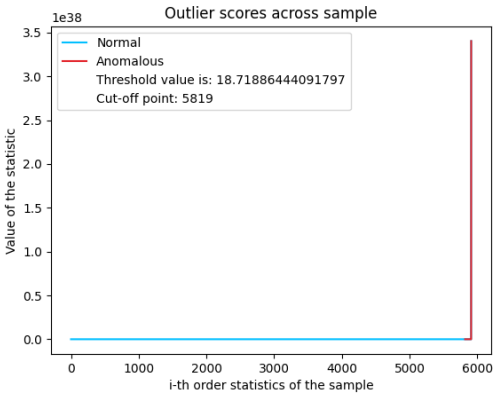
Order Statistics Scoring



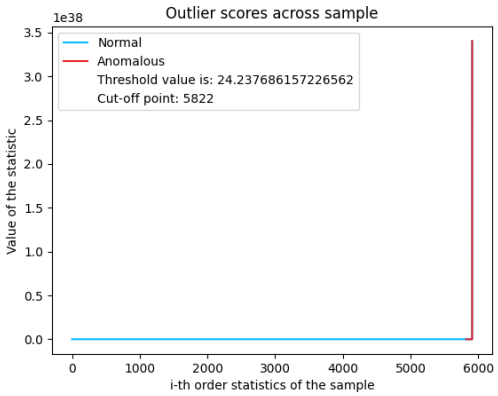
Simple Autoencoder

Figure A.32 ROC curves for the test set of MNIST digit 7.

A.9 Extra plots from MNIST digit 8

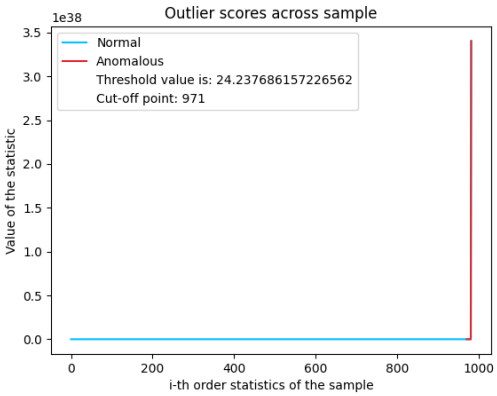


Order Statistics with Early Stopping

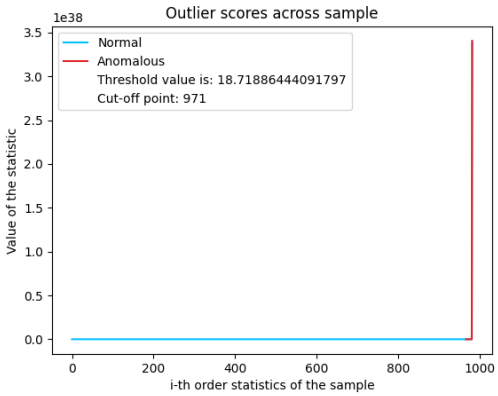


Order Statistics Scoring

Figure A.33 Outlier plots for the train set of MNIST digit 8.



Order Statistics with Early Stopping



Order Statistics Scoring

Figure A.34 Outlier plots for the test set of MNIST digit 8.

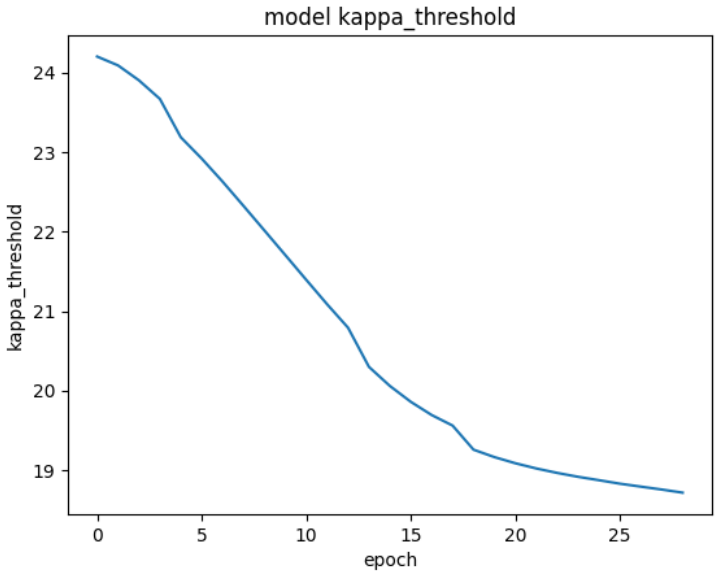
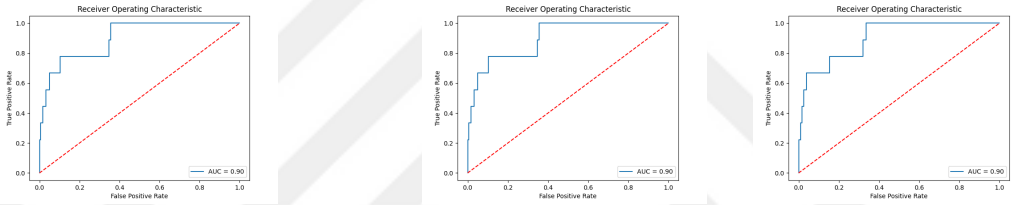


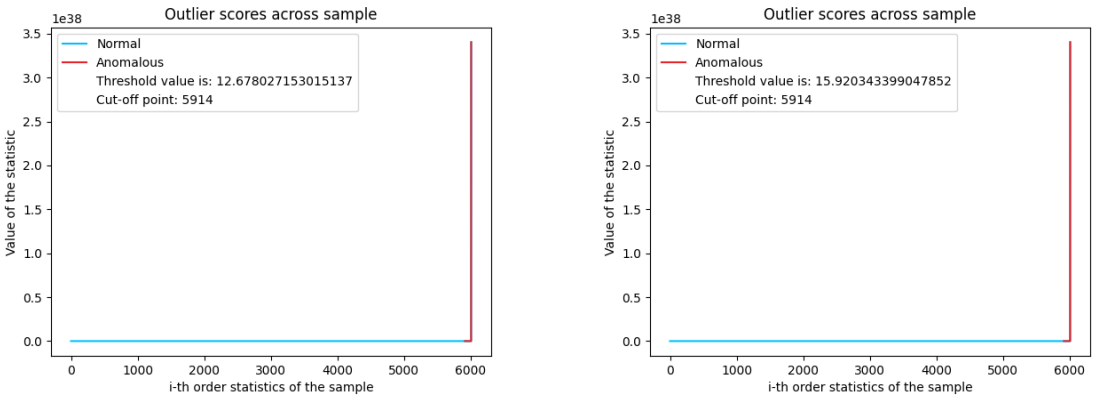
Figure A.35 κ-threshold values during training of the algorithm for MNIST digit 8.



Order Statistics with Early Stopping Order Statistics Scoring Simple Autoencoder

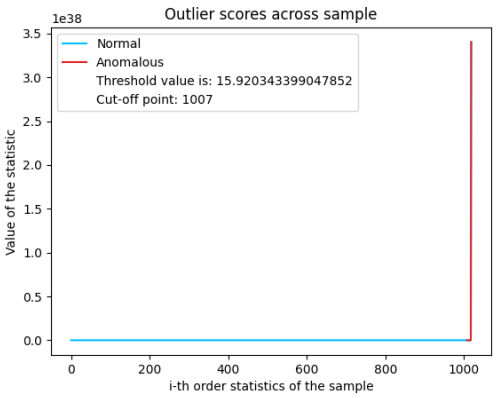
Figure A.36 ROC curves for the test set of MNIST digit 8.

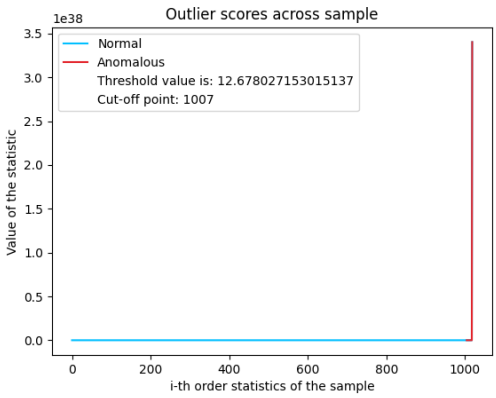
A.10 Extra plots from MNIST digit 9



Order Statistics with Early Stopping Order Statistics Scoring

Figure A.37 Outlier plots for the train set of MNIST digit 9.





Order Statistics with Early Stopping Order Statistics Scoring

Figure A.38 Outlier plots for the test set of MNIST digit 9.

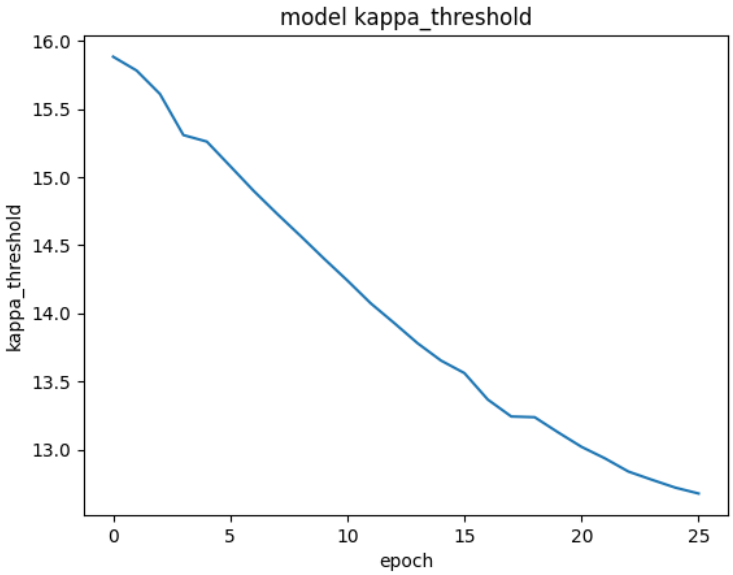
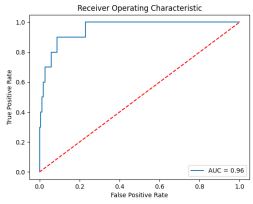
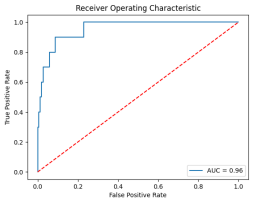
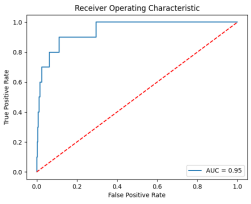


Figure A.39 κ-threshold values during training of the algorithm for MNIST digit 9.



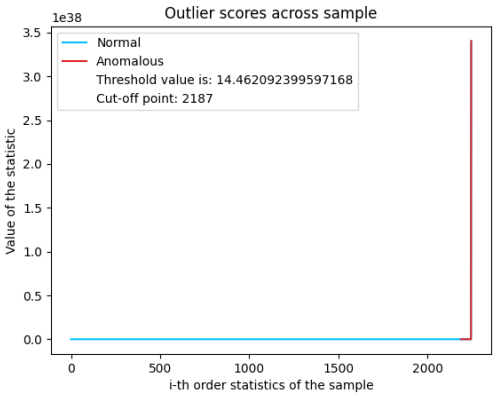




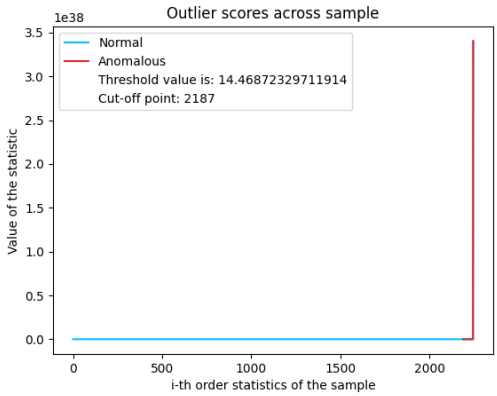
Order Statistics with Early Stopping Order Statistics Scoring Simple Autoencoder

Figure A.40 ROC curves for the test set of MNIST digit 9.

A.11 Extra plots from Musk

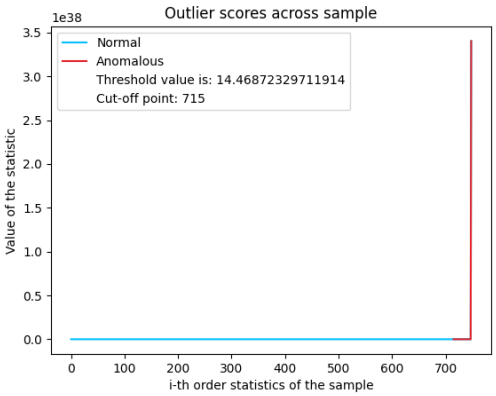


Order Statistics with Early Stopping

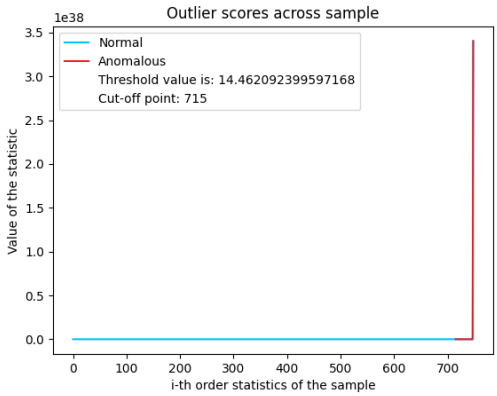


Order Statistics Scoring

Figure A.41 Outlier plots for the train set of Musk.



Order Statistics with Early Stopping



Order Statistics Scoring

Figure A.42 Outlier plots for the test set of Musk.

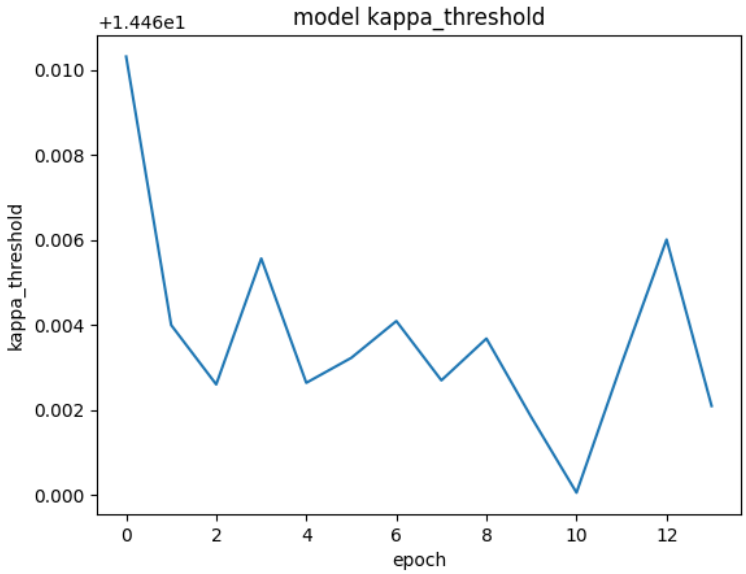
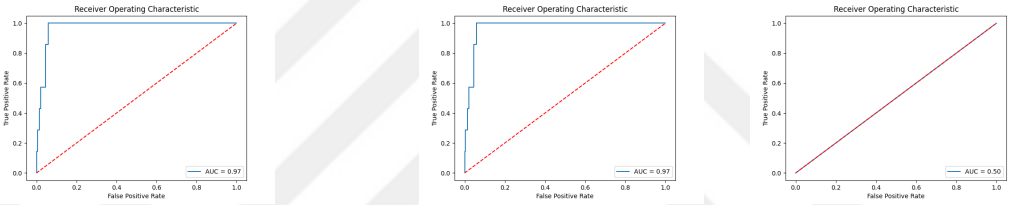


Figure A.43 κ-threshold values during training of the algorithm for Musk.



Order Statistics with Early Stopping Order Statistics Scoring Simple Autoencoder

Figure A.44 ROC curves for the test set of Musk.

PUBLICATIONS FROM THE THESIS

Conference Papers

1. A. Z. Balcıoğlu, E. Çene, “Order statistics based training and scoring algorithms for deep outlier detection,” in 8th International Conference on Advances in Statistics, 2022, p. 15