

Research Repository UCD

Title	Semi-Supervised Anomaly Detection Using One-Class RBF Networks
Authors(s)	Bazargani, Mehran Hossein Zadeh
Publication date	2021
Publication information	Bazargani, Mehran Hossein Zadeh. "Semi-Supervised Anomaly Detection Using One-Class RBF Networks." University College Dublin. School of Computer Science, 2021.
Publisher	University College Dublin. School of Computer Science
Item record/more information	http://hdl.handle.net/10197/12864

Downloaded 2026-11-16 00:02:14

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)



© Some rights reserved. For more information



Semi-Supervised Anomaly Detection Using One-Class RBF Networks

Mehran Hossein Zadeh Bazargani

This thesis is submitted to
University College Dublin
in fulfilment of the requirements for the degree of Doctor of Philosophy
in Computer Science.

School of Computer Science

Head of School: Prof. Chris Bleakley

Supervisor: Dr. Brian Mac Namee

January 2021

CONTENTS

Abstract viii

Acknowledgements xi

Parameters xii

Abbreviations xiv

1 Introduction 1

1.1 Research Context 1

1.2 Research Questions 4

1.3 Contributions 4

1.4 Thesis Overview 6

2 Related Work 7

2.1 Introduction 7

2.2 Anomaly Detection Techniques 10

2.2.1 SVM-based Anomaly Detection 10

2.2.2 Statistical Anomaly Detection 11

2.2.3 Distance-based Anomaly Detection 14

2.2.4 Reconstruction-based Anomaly Detection 14

2.2.5 Decision Tree based Anomaly Detection 16

2.2.6 Hybrid Anomaly Detection 17

2.2.7 Deep One-Class Anomaly Detection 18

2.3 RBF Networks 20

2.4 Using RBF Networks for Anomaly Detection 23

2.5 Summary . . . 24

3 Experiment Design **27**

3.1 Introduction . . . 27

3.2 Datasets Used . . . 27

3.2.1 Feature-based Datasets . . . 27

3.2.2 Raw Datasets . . . 29

3.3 Dataset Pre-processing . . . 30

3.3.1 Feature-based Datasets Pre-processing . . . 30

3.3.2 Image Dataset Pre-processing . . . 31

3.3.3 MIT-BIH Arrhythmia Dataset Pre-processing . . . 31

3.4 Experimental Method . . . 33

4 Radial Basis Function Data Descriptor Networks **35**

4.1 Introduction . . . 35

4.2 The Radial Basis Function Data Descriptor Network . . . 36

4.2.1 Interpreting RBFDD Networks . . . 39

4.3 Evaluation Experiment . . . 41

4.4 Results & Discussion . . . 43

4.5 Summary . . . 48

5 Elliptical Basis Function Data Descriptor Networks **50**

5.1 Introduction . . . 50

5.2 The Elliptical Basis Function Data Descriptor (EBFDD) Network . . . 51

5.3 Evaluation Experiment . . . 57

5.4 Results & Discussion . . . 57

5.4.1 The Number of Trainable Parameters . . . 64

5.5 Summary . . . 65

6 Deep Radial Basis Function Data Descriptors **67**

6.1 Introduction . . . 67

6.2 Deep Radial Basis Function Data Descriptor (D-RBFDD) Networks . . . 69

6.3 Evaluation Experiment . . . 72

6.4 Results and Discussions . . . 74

6.5 Summary . . . 82

7 Conclusion **83**

7.1 Summary of Contributions 83

7.1.1 The RBFDD Network 83

7.1.2 The EBFDD Network 84

7.1.3 The D-RBFDD Network 85

7.2 Future Work 85

LIST OF TABLES

1.1 Contribution Summary 6

3.1 The details of feature-based datasets 29
3.2 The experiment scenarios for the MNIST, Fashion-MNIST, and MIT-BIH Arrhythmia datasets 30
3.3 Classes used from the MIT-BIH Arrhythmia Dataset 31

4.1 Hyper-parameter ranges for RBFDD, OCSVM, AEN, GMM, and Isolation Forest 43
4.2 Experiment Scenarios and Results for Binary-Class Datasets 43
4.3 Experiment Results for Multi-Class Datasets: One-vs-All Scenarios . . 44
4.4 Experiment Results for Multi-Class Datasets: All-vs-One Scenarios . . 45
4.5 Experiment Results for Multi-Class Datasets: Difficult Scenarios . . . 46
4.6 Average Rank Across All Experiments 47

5.1 Hyper-parameter ranges for EBFDD, RBFDD, OCSVM, AEN, GMM, and Isolation Forest 57
5.2 Experiment Results and Scenarios for Binary-Class Datasets 59
5.3 Experiment Results for Multi-Class Datasets: Difficult Scenarios . . . 59
5.4 Experiment Results for Multi-Class Datasets: One-vs-All Scenarios . . 60
5.5 Experiment Results for Multi-Class Datasets: All-vs-One Scenarios . . 61
5.6 Average Rank Across All Experiments 62
5.7 All Scenarios: Win/Lose/Tie for Friedman Aligned Test with Significance Levels $\alpha : * = 0.1, ** = 0.05, * * * = 0.01$ 63

5.8 One-vs-All scenarios: Win/Lose/Tie for Friedman Aligned Test with Significance Levels $\alpha : * = 0.1, ** = 0.05, * * * = 0.01$ 63
5.9 All-vs-One scenarios: Win/Lose/Tie for Friedman Aligned Test with Significance Levels $\alpha : * = 0.1, ** = 0.05, * * * = 0.01$ 63

6.1 Hyper-parameter ranges for RBFDD, D-RBFDD, DeepSVDD, OCSVM, CAE-1/CAE-2, and Isolation Forest 73
6.2 Experiment results on the MIT-BIH Arrhythmia dataset 76
6.3 Experiment results on the MNIST and Fashion MNIST datasets 76
6.4 Results of Friedman test for the MNIST and Fashion MNIST datasets 79
6.5 Results of Friedman test for the MIT-BIH Arrhythmia dataset 79
6.6 Total training time of the models 81

LIST OF FIGURES

2.1 Standard Architecture of an RBF Network with H Gaussian Kernels (Alpaydin, 2004) 21

3.1 The basic pattern of electrical activity across the heart (Ashley and Niebauer, 2004) 32

3.2 Individual heart-beat visualisations from MIT-BIH Arrhythmia dataset 33

4.1 The RBFDD network. 37

4.2 Visualisation of the training process of the RBFDD network and the learned decision surface 40

4.3 Visualisation of Gaussian centres in a trained RBFDD network 41

4.4 The visualisation of the rank distribution of RBFDD, AEN, GMM, iForest, and OCSVM across the experiments 47

5.1 The Architecture of the EBFDD network. 51

5.2 Visualisation of the training process of the EBFDD network and the learned decision surface 56

5.3 The visualisation of the rank distribution of EBFDD, RBFDD, AEN, GMM, iForest, and OCSVM across the experiments 58

5.4 Critical difference plots from Friedman's test with Finner p-value correction with $\alpha = 0.05$ 62

6.1 Three approaches to deepening RBFDD networks. The red lines highlight the fixed portion of each model. 70

6.2 The Deep RBFDD Anomaly Detector 72

6.3 Critical difference plots from Friedman's aligned test results where the statistical significance is $\alpha = 0.05$ 78

ABSTRACT

Anomaly detection can be defined as *"the problem of finding patterns in data that do not conform to expected behavior"* (Chandola et al., 2009) and has been used in real world applications such as fraud detection (Ghosh and Reilly, 1994), monitoring healthcare (Šabić et al., 2020), intrusion detection (Ghosh et al., 1998), and crowd motion analysis (Zhou et al., 2016). The main challenge in anomaly detection is the imbalanced nature of the training data, since obtaining data belonging to the anomalous class is usually expensive, if not impossible, while the data belonging to the normal class is much more accessible. Hence, since the training data is almost always dominated by the data instances of the normal class, anomaly detection is sometimes framed as the *one-class classification* problem.

This thesis investigates the possibility of modifying the Radial Basis Function (RBF) networks into effective *one-class RBF networks* for the task of anomaly detection. RBF networks are selected as the foundation for this thesis due to their attractive properties, which include but are not limited to: being fast and efficient, being interpretable (Jin and Sendhoff, 2003) and being adaptable to dynamic changes (Liu et al., 2020; Han et al., 2011) in the distribution of the data with respect to time.

A shallow one-class RBF network is proposed, and then by eliminating certain restrictions, a stronger and more flexible version of these networks is also developed. Finally, a strategy for deepening one-class RBF networks is proposed so that they can work with raw high-dimensional datasets, while benefiting from the automatic feature extraction capability of deep learning. Through evaluation experiments it is shown that these newly proposed approaches lead to anomaly detection models that match or exceed the performance of current state of the art approaches while maintaining the advantages of RBF networks.

ACKNOWLEDGEMENTS

Thanks be to God for crossing my path with the right people, at the right time, and deflecting the pitfalls, unbeknown to me.

My gratitude goes to my supervisor, Dr. Brian Mac Namee. Working with him has taught me a lot as to what it means to be a researcher. He never deprived me of his ever-present support throughout my academic and personal life and I am indeed privileged to have known him.

I would like to thank my lovely parents for teaching me the right values, and for helping me build a solid foundation for my character. Even under the most difficult and excruciating circumstances, they sacrificed whatever they could to raise me up in the world. Indeed words fall short to show my enormous gratitude and love for them. Unlimited thanks to my lovely mother, Narges, whose refreshing smile and positive energy made the miles between us meaningless and the inundating difficulties of life weightless. Infinite thanks to my father, Kamel Pasha, whose mountain-like character was always there for me to boost my confidence and shed light on my path, whenever I felt lost.

My deepest gratitude goes to my Irish parents, Irenee and Brian, who were there literally every single step of the way. They were always there to celebrate my success, as well as, listen and support me whenever I needed to pour my heart out. They made their home, my home and haven where I knew I was safe and welcome. They were there, with an unfailing patience and support to help me figure out the right path whenever I was lost in the darkness. I feel absolutely honored and privileged to have known my lovely Irish parents. Their lovely daughter and granddaughter, Emma and Isabella, would pay us occasional visits and I have to say that having them around was always like a fresh breath of air to distract me from whatever algorithmic disaster that was awaiting me behind my screen. My lovely Irish grandma, Dolores, would

always teach me the right values in life and would always remind me of what truly matters in life so I would not lose my way. I will never forget her joyful smile, laughs, and cheerful character. I would also like to thank Lisa, who in my mind is The True Spirit of Ireland, with an ever-lasting sense of humor and joy! Whenever she was around it would have been impossible not to feel good and laugh your head off.

I would like to thank my external examiner Dr. Adriano Lorena Inacio de Oliveira, my viva chair Dr. Catherine Mooney and my internal examiner Dr. Aonghus Lawlor for agreeing to be there for my viva and for giving me the chance to defend my thesis on an important day of my life.

I would also like to thank my RSP panel members, Prof. Pádraig Cunningham, and Dr. Gianluca Pollastri. Their insights throughout these four years were always there to make sure I would not make the wrong choices and stay on the right path in my research.

I would like to thank my dear friend and colleague Ellen Rushe. When I first started my Ph.D. studies in UCD, everything was alien to me. Ellen always supported me throughout my problems and helped me fix them and settle in as smoothly as possible. I will never forget the day that she sat with me and spent four painstaking hours to fix a bug in my code. I can guarantee that if it was not for her, my progress would have been delayed by months with loads of headaches. I have learned a lot from her, and I would like to think of her as one of my mentors in computer science. I do not have the slightest doubt that she will be an influential figure in the realm of machine learning, in the future!

I would like to thank my dear friend, Hamed, who was the pushing force behind many good decisions in my life, including creating my Machine Learning educational platform, MLDawn. Of course, many thanks goes to Hamed's lovely wife, Nasim, who would graciously send me the loveliest Iranian dishes that were reminiscent of my home. Hamed, his loveliest little son and daughter, Radman and Mania, and I have had hours of fun walks on the UCD campus, which never failed to change my mood for the better (We used to laugh and call this a *concept drift* in my mind). Thank you all for being there for me and creating so many great memories in my life in Ireland! You are a gorgeous family.

Last but not least, I would like to thank my guardians who were involved in the pro-

cess of proof-reading my thesis and gave me the most helpful feedback to improve the quality of my thesis: Ellen Rushe, Dr. Arjun Pakrashi, Dr. Hamed Z. Jahromi, Dr. Gunjan Kumar, Dr. Khalil Muhammad, Eoin Delaney, and Farzad Abar.

PARAMETERS

μ	The mean of a Gaussian kernel/distribution
s	The spread of a Gaussian kernel/distribution
σ^2	The variance of a Gaussian kernel/distribution
Σ	The covariance matrix of a Gaussian kernel/distribution
π	The mixing probability (i.e., prior probability) for a given Gaussian in a Gaussian Mixture Model (GMM)
w	The weight associated with a Gaussian kernel in RBF, RBFDD, EBFDD, and DRBFDD networks
β	The spread/covariance regulariser coefficient for RBFDD, EBFDD, and DRBFDD networks
λ	The weight regulariser coefficient for RBFDD, EBFDD, DeepSVDD, and DRBFDD networks
ν	The upper bound on the fraction of training errors and a lower bound of the fraction of support vectors in OCSVM and the controller of the trade-off between violations of the boundary and the volume of the hyper-sphere in DeepSVDD
γ	The kernel coefficient in OCSVM

ABBREVIATIONS

AEN	Auto-Encoder
ANN	Artificial Neural Network
AMDN	Appearance and Motion Deep Network
AUC	Area Under the Curve
CAE	Convolutional Auto-Encoder
D-RBFDD	Deep Radial Basis Function Data Descriptor
DDA	Dynamic Decay Adjustment
DAGMM	Deep Auto-encoding Gaussian Mixture Model
DAN	Denoising Auto-Encoder
DBN	Deep Belief Network
DeepSVDD-OC	One-class Deep Support Vector Data Descriptor
DeepSVDD-SB	Soft-boundary Deep Support Vector Data Descriptor
EBFDD	Elliptical Basis Function Data Descriptor
EEG	Electroencephalogram
ECG	Electrocardiogram
EM	Expectation Maximisation
GMM	Gaussian Mixture Model
GAN	Generative Adversarial Network

iForest	Isolation Forest
iTree	Isolation Tree
KNN	K-Nearest Neighbor
LSTM	Long-Short Term Memory
MLP	Multi-Layer Perceptron
OCSVM	One-Class Support Vector Machine
OCNN	One-Class Neural Network
PSO	Particle Swarm Optimisation
PCA	Principal Component Analysis
RBF	Radial Basis Function
RBFDD	Radial Basis Function Data Descriptor
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic Curve
ResNet	Residual Neural Network
SVM	Support Vector Machine
SVDD	Support Vector Data Description
SOM	Self-Organising Map

CHAPTER

ONE

INTRODUCTION

This chapter first introduces the overall context in which the research described in this thesis was undertaken. Based on this context, a set of research questions that will be addressed are described. The contributions to the literature made within the thesis are then summarised, and finally an overview of the thesis document itself is provided.

1.1 Research Context

Chandola et al. (2009) define *anomaly detection* as *"the problem of finding patterns in data that do not conform to expected behavior"*. Anomaly detection is similar to classification, as input instances are labeled as either normal or anomalous, however, the training data in anomaly detection problems, tend to be highly imbalanced. The reason for this is that, sometimes obtaining data belonging to the anomalous class is rather expensive if not impossible, while the data belonging to the normal class is much more accessible. As a result, the training data in anomaly detection scenarios is almost exclusively dominated by instances of the normal class. This is why anomaly detection is sometimes framed as the *one-class classification* problem. Anomaly detection has many applications in the real world such as fraud detection (Ghosh and Reilly, 1994), healthcare monitoring (Šabić et al., 2020), intrusion detection (Ghosh et al., 1998), and crowd motion analysis (Zhou et al., 2016).

The specific requirements of any anomaly detector have a connection with the application at hand, however, in general the following features are expected from an anomaly detector:

- It is certainly desirable for an anomaly detector to function properly with low computational complexity, so that it can be applied to applications where large and high-dimensional data is used.
- Due to the imbalanced nature of data in anomaly detection scenarios, it is desirable for an anomaly detector to require few, if any, anomalous instances during training.
- A strong anomaly detector should be accurate in flagging anomalies in order to reduce the number of both *false positives* and *false negatives*.
- Since it is likely that anomalies will arise from very different data distributions —possibly even those that were previously unknown—a strong anomaly detector should be capable of learning decision boundaries that distinguish a core distribution (i.e., the distribution of the normal data) from many anomalous distributions.
- In many applications, due to their sensitive nature (e.g., healthcare), it is important to understand the reasoning behind an anomaly detector's decisions. Thus, an anomaly detector should be interpretable in such cases.
- In certain applications such as learning from streams (e.g., streams of incoming frames from a CCTV camera) the definitions of normal and anomalous can change over time, which is also known as concept drift/shift. In such cases, it is required for an anomaly detector to be able to adapt to such dynamic changes.

Radial Basis Function (RBF) networks (Alpaydin, 2004) already possess most of these properties. Not only are RBF networks computationally inexpensive, but they have proven effective in both classification and regression tasks. In addition, due to their local learning approach, RBF networks easily lend themselves to interpretation (Jin and Sendhoff, 2003), and are adaptable to changing concepts (Liu et al., 2020; Han et al., 2011). An anomaly detector based on RBF networks will automatically inherit these attractive properties.

The research in this thesis investigates the possibility of modifying RBF networks into effective *one-class RBF networks* for the task of anomaly detection. In addition, methods of improving the performance of these networks have also been investigated. First by making these models more flexible by eliminating certain constraints that they have

inherited from RBF networks, and also by exploring strategies to turn them into deep models suitable for dealing with raw and high-dimensional input data. Being able to work with raw data using deep models is desirable since feature engineering tends to be expensive, domain-specific and sometimes sub-optimal. In fact, RBF networks have already been successfully deepened (Zadeh et al., 2018; Hryniowski and Wong, 2019; Vidnerová and Neruda, 2018) with LeNet-5 (LeCun et al., 1998) being the first one.

In particular, the motivation behind deepening one-class RBF networks for anomaly detection is due to the fact that common deep learning approaches can be fooled by small changes in certain features of the input data (Nguyen et al., 2015; Szegedy et al., 2014). This relates to the inability of deep neural networks to classify input data that is slightly different in certain features from the training data (e.g., subtle anomalies such as adversarial attacks). One particular reason for this is due to the fact that most deep neural networks consist of a deep feature extractor and a linear classifier on the top, which can force the model to assign high confidence score to regions without any data point (Zadeh et al., 2018), where anomalies can occur. Unlike such methods, a deep one-class RBF network has a collection of non-linear RBF units in its last layer, which allows the model to capture more complex non-linear patterns in the data. It is interesting to note that RBF networks are naturally immune to negative examples (i.e., examples that do not belong to either of the classes in the training data, such as anomalies) and they give low confidence score to such examples (Goodfellow et al., 2015), which makes them a good choice as a foundation for anomaly detectors such as one-class RBF networks. By deepening one-class RBF networks, the deep feature extractor is pushed to map the input data to a new dimension where they form clusters and are covered by RBF units. As a result, an anomalous input data will be transformed to regions far from the covered region by the RBF units, and hence, flagged as anomaly.

Three specific directions of research under these themes were identified, and these directions led to the definition of a set of research questions that are described in the next section.

1.2 Research Questions

The following research questions (RQ) are addressed in this thesis:

RQ 1 RBF networks have been shown to be effective learners for regression and classification problems, but have not been extensively used for anomaly detection. **Is it possible to turn the Radial Basis Function (RBF) network into an effective anomaly detector that can learn only from the normal class?**

RQ 2 Models that are based on RBF networks are restricted by the inflexible radial nature of the kernels in RBF networks. **Will replacing the radial kernels with elliptical kernels in a one-class RBF network make the model a stronger anomaly detector?**

RQ 3 If the input data is not composed of well-behaved features—especially the case with raw high-dimensional data—it is general practice to use deep learning to automatically extract strong features from the data. **Is it possible to deepen the one-class RBF network architecture to make an effective anomaly detector applicable to high-dimensional raw data inputs, and what is the best way to do this deepening?**

1.3 Contributions

The main contributions to the literature described in this thesis are as follows:

- Motivated by RBF networks, a one-class RBF network named the *Radial Basis Function Data Descriptor (RBFDD) network* is proposed. The RBFDD network turns the RBF network into an effective anomaly detector by modifying its structure as well as using a novel one-class cost function. A large-scale benchmark experiment suggests that RBFDD networks, on average, out-perform state of the art approaches. This contribution addresses *RQ 1*.
- RBFDD networks are based on RBF networks and inherit the restriction of having inflexible radial kernels in their architecture. By replacing the radial kernels in RBFDD networks with elliptical kernels, a more flexible anomaly detector is proposed—the *Elliptical Basis Function Data Descriptor (EBFDD) network*. A set

of benchmark experiments show that the added flexibility of EBFDD networks leads to a more effective anomaly detector that out-performs the RBFDD network approach and current state of the art methods. This contribution addresses *RQ 2*.

- Similar to any shallow neural network, RBFDD networks are designed to work with input data that is composed of a set of well-behaved features. An effective approach to deepening RBFDD networks called the *Deep RBFDD (D-RBFDD) network* is proposed, which allows them to work with high-dimensional, raw input data. In a set of benchmark experiments, it is shown that when dealing with raw high-dimensional data, on average, the D-RBFDD network out-performs its predecessor the RBFDD network, while either out-performing the state of the art or demonstrating competitive performance with it. This contribution addresses *RQ 3*.

These contributions are supported by the following publications:

- M. H. Bazargani and B. Mac Namee, **Radial Basis Function Data Descriptor (RBFDD) network: An anomaly detection approach**, In ODD v5.0 workshop: Outlier Detection De-constructed London,UK, 2018.
- M. H. Bazargani and B. Mac Namee, **The Elliptical Basis Function Data Descriptor (EBFDD) network: A one-class classification approach to anomaly detection**, In Proceedings of European Conference on Machine Learning (ECML), 2019.
- M. H. Bazargani, A. Pakrashi, and B. Mac Namee, **The Deep Radial Basis Function Data Descriptor (D-RBFDD) Network: A One-Class Neural Network for Anomaly Detection**, Under review at: IEEE Transactions on Neural Networks and Learning Systems: Special Issue on Deep Learning for Anomaly Detection, 2020.

The contributions—as well as the research questions they address, the chapter in the thesis in which they are described, and the publications that support them—are summarised in Table 1.1.

Table 1.1: Contribution Summary

Research Question	Contribution	Chapter	Publication
RQ 1	RBFDD networks	Chapter 4	(H. Bazargani and Mac Namee, 2018)
RQ 2	EBFDD networks	Chapter 5	(H. Bazargani and Mac Namee, 2019)
RQ 3	D-RBFDD networks	Chapter 6	(H. Bazargani et al., under review)

1.4 Thesis Overview

The remainder of this thesis is structured as follows:

- Chapter 2 reviews related work in anomaly detection, lays the foundation of this thesis by exploring RBF networks and how they have been previously used for anomaly detection, and concludes by identifying the research gap that this thesis addresses.
- Chapter 3 introduces the experimental approach that is consistent throughout this thesis. The datasets used, the pre-processing of these datasets, the experimental method, and the performance measures used are all introduced.
- Chapter 4 introduces the Radial Basis Function Data Descriptor (RBFDD) network approach, a shallow one-class classifier, and demonstrates the efficiency of the RBFDD network approach through a large-scale benchmark experiment.
- Chapter 5 introduces the Elliptical Basis Function Data Descriptor (EBFDD) network, which is a stronger anomaly detector than its predecessor (i.e., the RBFDD network). A benchmark experiment evaluating its performance is included.
- Chapter 6 explores deepening strategies for RBFDD networks and proposes the Deep RBFDD (D-RBFDD) network. A benchmark experiment comparing deepening strategies and comparing D-RBFDD against current state of the art methods is included.
- Chapter 7 reflects back on the work described in the thesis, describes promising directions for future work, and concludes the thesis.

CHAPTER

TWO

RELATED WORK

2.1 Introduction

Anomaly detection is an important field of research as anomalies in data correspond to significant, often critical, and actionable information. In the literature the terms *outlier* and *anomaly* have both been used, sometimes interchangeably, in the context of anomaly detection (Chandola et al., 2009). Additionally, the term *novelty detection* (Markou and Singh, 2003a,b) is also often used synonymously, however, there is a distinction: in novelty detection, the goal is to detect previously unobserved patterns in the data (Chandola et al., 2009), for instance, the emergence of a new hashtag on Twitter that was previously unknown, and then incorporate the novel patterns into the model.

Anomaly detection is a challenging and important problem and there are many approaches that have been proposed to tackle it in different application domains. Typically, all such methods have been applied to datasets where there is an abundance of data instances that correspond to the normal condition (i.e., positive class) of a system (e.g., intrusion detection system, ecosystem monitoring system, etc.), whereas, there are insufficient, if any, data instances that describe the anomalies (i.e., negative class). The reason for scarcity of negative samples is usually due to the highly infrequent nature of the occurrences of anomalies. For example, it could be the case that the only way to obtain data regarding certain anomalies in the system is for certain components of the system to be destroyed, and this happens rarely. Conversely, most of the time the system works fine and normally, hence the data regarding the normal condition of the system is abundant and ever-growing.

Anomaly detection has attracted research attention in application domains including fraud detection (Ghosh and Reilly, 1994), healthcare monitoring (Šabić et al., 2020), intrusion detection (Ghosh et al., 1998), and crowd motion analysis (Zhou et al., 2016). The types of critical systems in these domains can be quite complex, making it difficult to fully understand the relationships between all of their individual components. A direct consequence of this is that the number of possible anomalies that could happen can grow very large. To add more to the challenge, some of these anomalies may well be unknown even by domain experts due to their rare occurrences. Lack of sufficient data for anomalies and lack of knowledge about all possible anomalies that could happen make multi-class classification approaches, common in other machine learning applications, often unsuitable for the task of anomaly detection. Consequently, anomaly detection is often approached within the framework of *one-class classification* (Moya et al., 1993), in which the assumption is made that the normal class is very well sampled, while *all* other classes are severely under-sampled, if not completely absent, in the available training data.

Based on the extent to which labels (i.e., ground truth) are available, anomaly detection techniques can operate in either of these three modes (Chandola et al., 2009):

- *Supervised Anomaly Detection*: These techniques assume that labels for both the normal and anomalous classes are available and the typical approach is to train a binary classifier to discriminate between normal and anomalous instances. The main challenge with the supervised approach to anomaly detection is that, in general, the anomalous data is far fewer than the normal data as obtaining labeled instances of anomalies is often very challenging. Hence, training datasets are highly imbalanced and this can deteriorate the performance of a binary classifier.

- *Semi-supervised Anomaly Detection*: These techniques assume that there is access to labels *only* for the normal data during training. Thus, they do not need labels for the anomalous data in order to be trained for anomaly detection, which makes them more applicable to anomaly detection, compared to approaches based on supervised learning. In general, the approach is to build a model to represent the normal behavior during training, and utilise the model for detecting anomalies during the testing phase.

- *Unsupervised Anomaly Detection*: These techniques require no labels during training, and assume that the training data consists of both normal and anomalous data. They also assume that the normal data is far more prevalent than the anomalous data in the test data, which, if not true, will cause a high false alarm rate.

The main focus of this thesis is on the semi-supervised approaches to anomaly detection. For the purpose of review, in this thesis, semi-supervised anomaly detection techniques in the literature have been grouped into the following categories [1]:

1. **SVM-based techniques**: This category focuses on some of the most important modified versions of Support Vector Machine (SVM) that have converted SVMs into one-class classifiers. This category is mainly dominated by *One-Class SVM (OCSVM)* (Scholkopf et al., 2001) and *Support Vector Data Description (SVDD)* (Tax and Duin, 2004) models.

2. **Statistical techniques**: Statistical anomaly detection approaches assume a stochastic model in which normal data occurs in the high probability regions.

3. **Distance-based techniques**: With a well-defined distance measure, this category takes advantage of both the *closeness* and prevalence of normal data instances and is mainly dominated by *clustering* (e.g., k-means) and *nearest neighbor* (e.g., One-Class K-nearest Neighbor) algorithms.

4. **Reconstruction-based techniques**: These techniques learn to reconstruct normal data instances with high quality after dimensionality reduction, and detect anomalies when instances cannot be reconstructed. The category is dominated by *Auto-Encoders (AENs)* (Baldi, 2012) and their different flavors.

5. **Decision tree based techniques**: In this category decision trees have been modified for the task of anomaly detection. This category is dominated by a family of decision tree models named the *Isolation Forest (iForests)*, which take advantage of the fact that normal data is densely represented, while anomalies are not.

6. **Hybrid techniques**: Under this category, a mixture of multiple approaches is used for anomaly detection. The combination tends to consist of a

[1] Sometimes a clear distinction between semi-supervised and unsupervised modes is not trivial as some of these techniques can be applied in either of the two modes.

reconstruction-based technique (e.g., AEN) and a one-class classifier (e.g., OCSVM). The training of the two components typically happens separately under different objectives.

7. **Deep one-class techniques**: These techniques contrast with hybrid techniques in that by using one global one-class objective to be minimised, a deep network is trained as a whole for anomaly detection.

In this chapter, we will describe these categories, and cover some of the work that have been done under each one. Furthermore, the Radial Basis Function (RBF) network, which is the foundation of the approaches described in this thesis, is introduced and its application to anomaly detection is explored. This leads us to an explanation of the research gap that this thesis addresses as well as the motivation behind the approach followed in order to address this gap.

2.2 Anomaly Detection Techniques

In this section, seven different categories of techniques for tackling anomaly detection are discussed, and for each category, some example techniques are explored.

2.2.1 SVM-based Anomaly Detection

Anomaly detection approaches using Support Vector Machines (SVMs) (Hearst et al., 1998) focus on finding classification boundaries based on the structure of the training data. When faced with a new data instance, the classification takes place based on the location of the test data point with respect to the boundary. The boundary is determined using the data that lie closest to it (i.e., support vectors) and all other non-support vector training data are ignored when determining boundaries. The dominant flavors of SVM models for the task of anomaly detection are: *One-Class SVM (OCSVM)* (Scholkopf et al., 2001) and *Support Vector Data Description (SVDD)* (Tax and Duin, 2004). The idea underpinning OCSVM is quite similar to the standard SVM method, and the *kernel trick* (Schölkopf, 2000) is still a key part of the OCSVM for the transformation of the input space into a feature space of higher dimensionality. The OCSVM approach finds a hyper-plane that separates all of the normal data points in

a training set from the origin, while maximising the distance between the origin and this hyper-plane. This results in a binary function, whose output is $+1$ for the regions of the input space belonging to the normal data, and -1 anywhere else. In OCSVM the origin plays an important role and this is a drawback since the origin here acts as a prior as to where the anomalous class instances are believed to lie, which is not always the case (Campbell and Bennett, 2000). The main issue with OCSVM models, however, is that they do not scale well. For large datasets the computational and storage requirements of OCSVMs grow polynomially with dataset size (Awad and Khanna, 2015). In particular, the time complexity of OCSVMs depends on the number of samples in the dataset, $n_{samples}$, and number of features, $n_{features}$, and scales between $O(n_{features} \times n_{samples}^2)$ and $O(n_{features} \times n_{samples}^3)$. Variations of the OCSVM approach include the Support Vector Machine Data Description (SVDD) (Tax and Duin, 2004), which uses hyper-spheres rather than hyper-planes to achieve separation.

In (Gardner et al., 2006), an OCSVM is used to detect seizures in electroencephalogram (EEG) time series data. EEG signals are first windowed, and then feature extraction is performed in interictal epochs (i.e., the time between successive seizures) in order to generate the training data for an OCSVM. The features are selected in such a way that they could strongly characterize the non-seizure EEG windows of data. An anomaly score is computed using the output of the trained OCSVM, which is then thresholded during testing, to classify a window of EEG signal as either normal or anomalous (i.e., seizure activity). In (Clifton et al., 2006), OCSVMs are used in the prediction of combustion instability in multi-channel combustion data, consisting of gas pressure and two combustion chamber luminosity measurements. A Wavelet transform (Zhang and Benveniste, 1992) is used for feature extraction, and these features are then fed into OCSVM for training it. One independent OCSVM is trained per time series channel only using the features extracted from the normal data for that channel. Multiple anomaly scores are computed using the outputs of each OCSVM, and the average of these anomaly scores are computed as the final anomaly score for a given test data instance.

2.2.2 Statistical Anomaly Detection

Statistical anomaly detection approaches assume a stochastic model in which normal data occurs in the high probability regions. Any data that occurs in the low probability

regions of the model are flagged as anomalous. In general, a statistical model is fit to the data (i.e., usually for the normal behavior) during training. In the test phase, if a test data instance has a low probability to have been generated by the learned model, it is flagged as anomalous. In general, there are two main approaches to statistical anomaly detection, namely, *parametric* and *non-parametric* approaches.

- *Parametric approaches*: In the parametric approach, it is assumed that the normal data come from a parametric distribution with parameters θ, where the parameters will be estimated using the training data. The most common distribution that is used is the Gaussian distribution whose parameters are (μ, σ^2), namely, the mean and the variance, respectively. Of course, with high-dimensional data, the Gaussians become multi-variate and that is where their shape is dictated by their covariance matrix, Σ, rather than a simple variance. These parameters can be estimated using *Expectation Maximisation (EM)* (Bishop, 2007). For a given test instance, an anomaly score is computed based on its distance from the mean of the distribution. As the next step, a threshold is applied to the score to determine whether the test data is an anomaly or not.

  There is another category of parametric approaches, that model the normal data with not a single parametric distribution, but as a mixture of parametric distributions. If a test instance does not belong to any of the learned models in the mixture, then it is flagged as anomalous. For example, Gaussian Mixture Models (GMMs) (Reynolds, 2009) assume that every single data point came from one of H different sources, each of which is a Gaussian probability distribution. For a given D-dimensional data point, each Gaussian component, h, in a GMM can be described by three main parameters: its D-dimensional mean vector, μ_h, a $(D \times D)$ covariance matrix, Σ_h, and a mixing probability, π_h, that measures the overall probability of observing a data point that comes from Gaussian h (i.e., the prior probability). The optimisation of the model parameters is done using the EM algorithm and the goal is to learn a model that maximises the likelihood of observing the training data. A GMM uses all of its Gaussian components during training, and one issue is to define the optimal number of Gaussian components, H.

  One of the earliest statistical anomaly detection approaches is proposed by Crathorne (1933), where the aim is to declare all data instances that are more

than 3σ distance away from the distribution mean, as anomalous. Their technique takes advantage of the fact that the $\mu \pm \sigma$ region contains 99.7% of the data instances for a Gaussian distribution, which is where the normal data is believed to reside. Several other statistical anomaly detection techniques that assume that the data follows a Gaussian distribution have been proposed (Rosner, 1983; Gilroy, 1995). In (Hickinbotham and Austin, 2000), GMMs are used to detect strains in airframe data. In (Spence et al., 2001), GMMs have been used to detect anomalies in mammographic image analysis, where for a given test data, an anomaly score is generated by computing the absolute log-likelihood of the Gaussian mixture given the test data, and if this anomaly score is below a threshold the test data is flagged as anomalous, and normal otherwise.

- *Non-parametric approaches*: In the non-parametric approach, however, we do not assume any distribution in the data in order to fit the data and capture its complexity. These techniques are also known as frequency-based techniques. Histograms are one of the most common approaches under this category and they can be used for anomaly detection. For instance, in the case of univariate training data, the histogram is built using the feature values in the training data. In the next step, for any given test data, this method checks if it falls into any of the bins in the histogram. If it does, it is labeled as normal, and anomalous otherwise. As a variant of the histogram-based techniques, an anomaly score can be assigned to each test instance based on the frequency of the bin in which it falls, where the lower the frequency of the bin is, the higher the anomaly score of the test instance will be. It is important to note that the size of the bins used during training plays an important role in the performance of this approach (Aggarwal, 2013; Chandola et al., 2009). The key point is that the methods under this category do not assume a particular distribution on the normal data (i.e., the data corresponding to the normal behavior). Histogram based techniques have been used for anomaly detection in a variety of applications such as intrusion detection (Eskin et al., 2001) and fraud detection (Fawcett and Provost, 1999).

2.2.3 Distance-based Anomaly Detection

Distance-based approaches to anomaly detection use a well-defined distance measure to compute the similarity between pairs of data points. Some of these methods use clustering to achieve anomaly detection. They assume that, after applying clustering to the normal data and computing the cluster centroids, anomalous data will lie far from their closest centroid, while normal data will lie close. The distance of each data sample from its cluster centroid is computed as the anomaly score for each data instance. When applied to test data, the distance between the test data sample and its closest centroid is computed, and if it is more than a pre-defined threshold, the test data is considered as an anomaly (Chandola et al., 2009; Alpaydin, 2004). Nearest neighbor algorithms also fall under distance-based methods, and they are quite popular in anomaly detection.

The classic K-nearest neighbor (KNN) algorithm has been modified for the task of one-class classification to address anomaly detection resulting in One-Class K-nearest neighbor (OCKNN) algorithm (Tax and Duin, 2000). OCKNN algorithm assumes that normal training data are surrounded by other normal samples in their neighborhood, whereas, an anomaly is assumed to be far away from those points. There are a variety of distance measures that can be suitable for different cases. The data type, numerical or categorical, plays an important role in the selection of the distance measure. Some of the most popular distance measures are the Euclidean distance (for numerical data), and Hamming distance (for categorical data). In (Hautamäki et al., 2004), the data instances are sorted by their average KNN distances. During testing, if a test data has an average KNN distance more than a threshold, it is flagged as anomalous. In (Angiulli and Pizzuti, 2002), a weighted sum of the distances of K-nearest neighbors is mapped to each data point in the input space, and then the data points with the largest weighted sum are labeled as anomalous.

2.2.4 Reconstruction-based Anomaly Detection

These methods model the underlying data and basically strive to reconstruct the input data as their output, typically through dimensionality reduction. Then an anomaly score is computed based on the reconstruction error, which is the distance between the input test data and its corresponding system output (i.e., the reconstructed input

test data). One of the most well-known reconstruction-based methods is the Auto-Encoder (AEN) (Baldi, 2012). An AEN is an Artificial Neural Network (ANN) used for unsupervised learning, which is used to learn a representation of the data, typically in a lower dimension. An AEN is not a discriminative but a generative model of the data and after transforming the input data to a representation with reduced dimensions (i.e., the features in the hidden layer), it tries to reconstruct the original input data from the dimensionally-reduced representation features. As a result, the output size of an AEN (regardless of the number of hidden layers) is always the same as the input. It is common practice to initiate the weights of an AEN at random and continue with the standard back-propagation of error algorithm (Hecht-Nielsen, 1988) to train the model, in an end to end fashion. In the task of anomaly detection, the AEN learns how to reconstruct the data belonging to the normal data with a low reconstruction error, however, during testing, if an anomalous data is inputted into the AEN, the model fails to reconstruct it accurately. As a result, the reconstruction error for an anomalous data is high, which is the reason that the magnitude of the reconstruction error is typically thresholded to make the final decision about the input. If this magnitude is higher than a certain threshold, then the input is classified as anomalous, and normal, otherwise.

AENs and all their variations have been increasingly used for anomaly detection (Xiong and Zuo, 2016; Yan and Yu, 2019; Wang et al., 2016; Li and Chang, 2019; Ribeiro et al., 2018). In (Xiong and Zuo, 2016), AENs are used to detect anomalies in geochemical population data, which tends to have unknown and complex multivariate probability distribution. During training, an AEN is trained to reconstruct the geochemical population data, which is mostly dominated by normal data instances, and has a small percentage of anomalies. Due to their rarity, anomalous data instances contribute far less to training the AEN compared to prevalent normal data instances. As a result, anomalous data instances are poorly encoded by the AEN and end up with high reconstruction error while the normal data is reconstructed well with a low reconstruction error. Finally, by thresholding the reconstruction error during testing, the AEN is used to recognise multivariate geochemical anomalies. As a common flavor of AENs, we have the Denoising Auto-Encoders (DANs), which use corrupted data as inputs (unlike the standard AEN where the inputs are left intact), and the reconstructed output is compared to the clean data input, thus resulting in more robust

learned features. For instance, in (Zhou and Paffenroth, 2017), an extension to DANs has been proposed for the task of anomaly detection.

Variational Auto-Encoders (VAEs) (Doersch, 2016) are deep Bayesian neural networks, and generative models unlike simple AENs. In (An and Cho, 2015), the reconstruction probability from the VAE is used for the task of anomaly detection. This reconstruction probability is computed by the stochastic hidden variables, and what is being reconstructed is not the input variable itself but the parameters of the input variable distribution. A given query is flagged as anomalous if its corresponding reconstruction probability is high, and normal otherwise. In (Xu et al., 2018) Donut, a method based on VAEs, is proposed to detect anomalies in time series of seasonal Key Performance Indicators (KPIs) (e.g., page views, number of orders, etc) in web applications. In their work, they have modified the Evidence Lower Bound (ELBO) that was introduced in traditional VAEs. Using these modifications, the contributions of the anomalies and missing values are excluded during training, and it has been shown that Donut out-performs VAE-based anomaly detection by a large margin.

2.2.5 Decision Tree based Anomaly Detection

Approaches based on decision trees have been developed for anomaly detection, such as the Isolation Forest (iForest) (Ting et al., 2008). The iForest tries to isolate individual data points in the training set by splitting the space randomly and repeatedly. The intuition behind this approach is that less splits should be required to isolate anomalous instances. The random partitioning produces noticeably shorter paths for anomalies mostly because an anomalous data instance has more distinguishable attribute values (compared to normal data instances) and is more likely to be isolated from the other data instances in early stages of random partitioning. More specifically, iForest builds an ensemble of Isolation Trees (iTrees) (Ting et al., 2008) for a given dataset. Every iTree is constructed by randomly partitioning a random subset of the entire dataset and the job is to isolate each individual data point in that subset. This random partitioning keeps happening until all samples are isolated, and then the next iTree is constructed on yet another random sub-sample of the dataset. A data instance is flagged as an anomaly if it has a short average path length (i.e., the expected path length for isolating the data instance) from the root of the iTrees across the entire iForest. Taking an average makes sense, since when an ensemble of random iTrees produce short path

lengths for isolating a particular data instance, then that data instance is most likely an anomaly. Finally, an anomaly score can be calculated based on the average number of splits required to isolate a data point across all iTrees.

iForests and their extensions have been frequently used for the task of anomaly detection. In (Susto et al., 2017), iForest is used to detect anomalies in a high-dimensional data monitoring problem, namely, a real-life industrial dataset related to Etching (i.e., one of the main steps in the semi-conductor manufacturing fabrication). In (Gopalan et al., 2019), a framework called *Partial Identification* is used for anomaly detection, which follows the intuition that anomalies can be easily discerned from the normal data by a few out-standing attributes. Similar to iForest, an ensemble of decision trees are constructed that partition the data into sub-cubes (i.e., data partitions), and the choice of split (unlike iForest) at each split node tends to favour partitions with a high variance in sparsity. Sparsity of a dataset in a sub-cube of the attribute space has been defined as the volume of the sub-cube over the total number of data points from within that sub-cube. Finally, the anomaly score for each leaf node is defined as the maximum sparsity over all the leaves reached across the entire forest. In (Karczmarek et al., 2020), iForest is augmented by k-means clustering so as to allow building search trees based on many branches as opposed to the original iForest model that is confined to two branches at every split node. In particular, k-means clustering is utilized to predict the number of such divisions on each split node for each tree in the forest. For a random sub-sample of the dataset, a random attribute is chosen. The interval of values between the minimum and maximum values of the sub-sample is divided into C clusters using k-means, and these clusters and their limits become the leaves for the node, and the same process will carry on for each of these leaf nodes. A final anomaly score for a given test data is computed based on the memberships of the test data across all the trees, where anomalous data instances will have a final anomaly score close to zero.

2.2.6 Hybrid Anomaly Detection

There are methods that combine different approaches together to accomplish anomaly detection. In most cases, hybrid methods consist of a reconstruction-based method, which handles the dimensionality reduction and feature extraction part for a subsequent classifier that would make the final classification task. As a result, the training

process consists of: 1) training the feature extractor, so that it would extract a rich set of features from the data and 2) feeding the extracted features into the classifier that is trained independently. The main issue with hybrid approaches is that the deep feature extractor is not trained on an anomaly detection objective, but on a different objective such as minimising the reconstruction error. As a result, the learned features may not be fully appropriate for the task of anomaly detection.

In (Marchi et al., 2015), after feeding a Denoising Auto-encoder (DAEN) with auditory spectral features, the output of the DAEN is used by a bidirectional RNN (Alpaydin, 2004) with LSTM (Hochreiter and Schmidhuber, 1997) units to detect acoustic anomalies. The training is conducted in a semi-supervised fashion, during which, the reconstruction error at the output layer is compared to a threshold for every time step to decide on whether the input signal is anomalous or not. In (Erfani et al., 2016), a Deep Belief Network (DBN) (Hinton et al., 2006) has been used to extract robust features from high-dimensional input data, while reducing the dimensionality so that a OCSVM (Scholkopf et al., 2001) can have a nice set of well-behaved feature vectors to work with. Similarly, in (Marir et al., 2018), a DBN model is trained in an unsupervised dimensionality reduction step, whose output features are then fed into a multi-layered ensemble SVM, for the task of detecting anomalous behavior in large-scale network traffic data. In (Xu et al., 2017), a fully unsupervised model called *Appearance and Motion DeepNet (AMDN)* is proposed for video anomaly detection at the frame level. For every frame of the video, appearance and motion features are extracted and fed into two separate *Stacked Denoising Auto-Encoders (SDAEs)* and the fusion of these two types of features are fed into a third SDAE. The features obtained in the bottle-neck layers of the three SDAEs are then fed into three OCSVMs each of which produces an anomaly score. Finally, for the task of anomaly detection, the anomaly score predictions of all three OCSVMs are linearly fused together to make a final decision.

2.2.7 Deep One-Class Anomaly Detection

Unlike hybrid approaches, these approaches aim to make the features extracted in latent representations more directly appropriate to the anomaly detection task by defining a one-class cost function that is used to train an entire network in an end-to-end fashion, guiding the network to produce features that are appropriate for anomaly detection.

In (Zong et al., 2018) the Deep Auto-encoding Gaussian Mixture Model (DAGMM) is proposed as an end-to-end approach to anomaly detection. An AEN is used to reduce the dimensionality of the data, while the reconstruction error and the low-dimensional representation from the bottle-neck of the AEN are fed into a Gaussian Mixture Model (GMM) (Reynolds, 2009), where both AEN and GMM are trained together as one model. Similarly, in (Chalapathy et al., 2018) a One-Class Neural Network (OCNN) for anomaly detection is proposed. OCNN combines the rich feature extraction property of deep neural networks with a proposed OCSVM-like cost function. At first, a deep AEN is trained to produce good representative features of the input data. Next, the encoder portion of this pre-trained AEN is fed into a simple one-layer neural network, the ultimate output of which will be used to compute the cost. The weights of both the encoder and the one-layer neural network will be trained simultaneously, while minimising the cost function. By combining the capability of deep neural networks to extract progressively rich features from the data with the proposed cost function, the aim is to obtain the hyper-plane that separates the normal data from the origin. In (Schlegl et al., 2017), AnoGAN, which is a deep model for anomaly detection based on Generative Adversarial Networks (GANs) (Goodfellow et al., 2016) is proposed. The generator network is trained to learn the distribution of the training data. Given a test instance it searches for a point in the latent space of the generator that would generate a sample that is closest to the test point. If an accurate distribution of the normal data has been learned, for a normal query, X_i, there should be a representation, u, in the latent space that the generator could use to generate a new data point, $G(u)$, that is similar to the normal query X_i. For an anomalous query a good representation, u, should not be found and, as a result, the generated data, $G(u)$, will not be similar to the query.

Finally, inspired by the Support Vector Data Description (SVDD) model (Tax and Duin, 2004), Deep-SVDD (Ruff et al., 2018) is another deep one-class neural network designed for anomaly detection. While the neural network is trained, the volume of a hyper-sphere that envelopes the data in the latent space is minimised. Thus, the neural network is trained to map the input data into a hyper-sphere with minimum volume. There are two versions of Deep SVDD: 1) Soft-boundary Deep SVDD which makes a compromise between the volume of the hyper-sphere and violations of the boundary; and 2) One-Class Deep SVDD which is a simpler version that assumes that

most of the training data is normal (i.e., low proportion of outliers).

2.3 RBF Networks

The new approaches described in this thesis are based on Radial Basis Function (RBF) networks. An RBF network (Broomhead and Lowe, 1988; Moody and Darken, 1989) is a local-representation learning technique, which divides the input space among local kernels. For every input data point, depending on where in the input space it appears, a fraction of these locally-tuned kernel units get activated. It is as if these local units have divided the input space among themselves and each one takes responsibility for a subspace. The very idea of locality, implies the need for a distance function that measures the similarity between a given input data, X_i (i.e., the i^{th} input data), with dimensionality D, and the center, μ_h, of every kernel unit, h. The common choice for this measure is the Euclidean distance, $||X_i - \mu_h||$. The response function for these local units, needs to have a maximum where $X_i = \mu_h$, and decrease as X_i and μ_h get less similar. The most commonly used response function for the RBF units is the Gaussian function:

$$P_h(X_i) = exp\left[-\frac{||X_i - \mu_h||^2}{2s_h^2}\right] \tag{2.1}$$

where μ_h and s_h denote the center and the standard deviation (i.e., spread) of the local unit h, defining a radially symmetric basis function.

The idea behind using such local models is that if we assume that, in the training data, there are clusters of data points, then for each one of these clusters we define a basis function, $P_h(X_i)$. This basis function gets activated if X_i belongs to cluster h. The closer X_i is to μ_h, the higher the value of the activation, $P_h(X_i)$, will be.

As a pre-training stage, it is common practice to apply an unsupervised clustering method (e.g., k-means clustering), to come up with a nice set of initial positions (i.e., centers) for the Gaussians, as opposed to random initialisation. A simple heuristic for initialising the spreads, once the centers are found by the pre-training stage, is to find the farthest data point covered by a cluster, and then consider half of that distance as the spread for that cluster (Alpaydin, 2004). One-third of that distance is also possible, but it is common practice to be more conservative. After these cluster centers and spreads are found, the outputs of the response functions, $P_h(X_i)$, for input data, X_i,

act as the inputs to a perceptron:

$$y_i = f\left(\sum_{h=1}^{H} w_h \times P_h(X_i) + w_0\right) \tag{2.2}$$

where f is the non-linear function applied in the output unit. Also, H, is the number of basis functions (i.e., kernels), w_h is the weight associated with unit h, and w_0 is the bias term. This whole structure is called a Radial Basis Function (RBF) network. The structure of a standard RBF network is shown in Fig. 2.1.

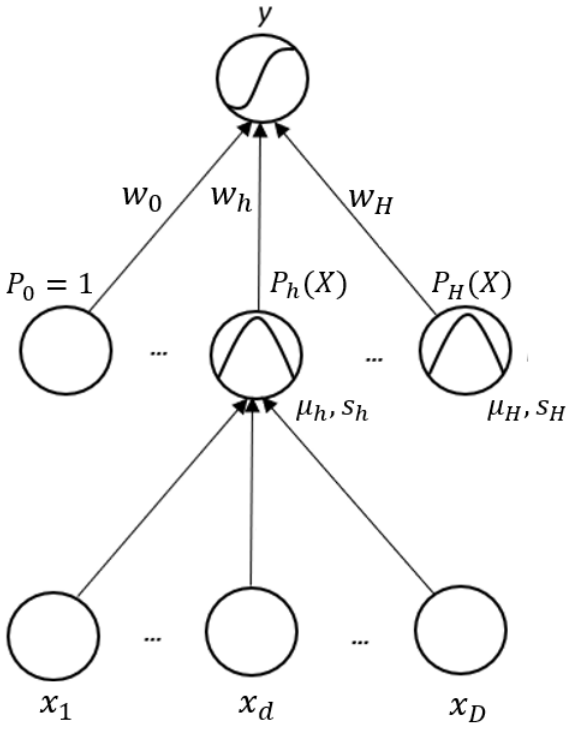


Figure 2.1: Standard Architecture of an RBF Network with H Gaussian Kernels (Alpaydin, 2004)

In Fig. 2.1, x_d denotes the d^{th} feature in the input data X_i, which is a D-dimensional vector. Also, μ_h, s_h, and $P_h(X_i)$ denote the mean, standard deviation, and the response (i.e., output) of the Gaussian kernel, h, for input data X_i with P_0 being the bias unit. Finally, w_h denotes the weight of the connection between the Gaussian kernel, h, and the output neuron. The standard RBF network in Fig. 2.1, is a supervised neural network. Provided a suitable non-linear function (i.e., f in Eq.(2.2)), the RBF network should be able to classify the input X_i into its corresponding class. The hyperbolic tangent, $tanh$, function is usually used for f, and the cost function to be minimised for mini-batches of size N during training is usually the squared cost function:

$$E = \frac{1}{2}\sum_{i=1}^{N}(t_i - y_i)^2 \tag{2.3}$$

where for a given input X_i, t_i is the ground-truth label, and the output of the model, y_i, is defined as:

$$y_i = tanh\left(z\left(X_i\right)\right) \tag{2.4}$$

where

$$z(X_i) = \sum_{h=1}^{H}\left(w_h \times P_h\left(X_i\right)\right) + w_0 \tag{2.5}$$

Based on the application of the back-propagation (Rojas, 1996) of error algorithm using gradient descent the following update rules to learn the parameters of the RBF network (i.e., means and spreads of the Gaussians, and the weights) can be derived[2]:

$$w_h \leftarrow w_h - \eta(y_i - t_i)(1 - y_i^2)P_h(X_i) \tag{2.6}$$

$$w_0 \leftarrow w_0 - \eta(y_i - t_i)(1 - y_i^2) \tag{2.7}$$

$$\mu_h \leftarrow \mu_h - \eta(y_i - t_i)(1 - y_i^2) \times w_h \times P_h(X_i)\frac{(X_i - \mu_h)}{s_h^2} \tag{2.8}$$

$$s_h \leftarrow s_h - \eta(y_i - t_i)(1 - y_i^2) \times w_h \times P_h(X_i)\frac{(||X_i - \mu_h||)^2}{s_h^3} \tag{2.9}$$

where $||X_i - \mu_h||$ is the Euclidean distance between input data X_i and the center of unit h.

Depending on the problem at hand, there are other possible variations of the RBF network. It could have multiple output units, for the task of multivariate regression or a classification task. The output activation function could be $tanh$ or $sigmoid$ for binary classification, and $softmax$ for multi-class classification.

[2]For simplicity, the update rules have been shown for when an individual input data X_i is fed into the RBF network and the output y_i is generated by the network. Generalisation of these update rules to a mini-batch learning scenario is straight-forward

2.4 Using RBF Networks for Anomaly Detection

RBF networks have been applied to anomaly detection in two main ways. First, if examples of both normal and anomalous data are available during model training, the standard binary/multi-class classification RBF networks can be used, as a supervised approach to anomaly detection.

In (Ma et al., 2008), the main focus of the work is on the optimisation method for the task of anomaly detection. A hybrid optimisation algorithm based on RBF networks has been proposed, which combines the traditional gradient descent algorithm with quantum-behaved *Particle Swarm Optimisation (PSO)* (Kennedy and Eberhart, 1995) to train the RBF network for anomaly detection. They have performed their experiments on the KDD Cup 1999 intrusion detection dataset (Dua and Graff, 2017), and during training, both normal and intrusion data have been used. Similarly in (Rapaka et al., 2003), an RBF network is trained for the task of intrusion detection. The model keeps adding hidden units to the architecture until a certain performance goal is met. An important feature of an RBF network is that it is capable of constructing local approximations to nonlinear input-to-output mappings. This approach is an example where the architecture of the RBF networks can be incrementally grown and adapted until a certain desired performance is achieved, by adding such local approximators. In (Bezerra et al., 2007), the RBF network is combined with a softmax output layer so as to produce outputs that can be interpreted as probabilities. The job is to not only detect faulty software modules but also to output the probability of defect. Interestingly, even though the datasets are highly imbalanced, a multi-class classification approach is followed in a supervised fashion, without any upsampling/data augmentation strategy to tackle the imbalancedness of the data.

In the second approach to using RBF networks for anomaly detection only examples of the normal class are used at training time, so the standard binary/multi-class classification approaches are no longer suitable. This problem can be addressed through modifying the dataset or through modifying the algorithm.

As an example of the first type, in (Oliveria et al., 2004a), an augmented training set is composed using the instances belonging to the normal class and random proxy anomalous patterns. Their work focuses on comparing the use of different classifiers, namely, Multi-Layer Perceptrons (MLPs), RBF networks, and the mixture of the two

in terms of their performance. The task is to detect anomalies in time series data, by classifying every window of data as either normal or anomalous. In (Oliveira et al., 2004b), RBF networks have been used for the task of anomaly detection for time series data, and training happens using Dynamic Decay Adjustment (DDA) (Berthold and Diamond, 1994), which is a constructive algorithm that has been used to construct and train RBF networks for classification tasks. The time series data is windowed and for each window, the RBF network will decide whether it is normal or anomalous. Even though their method does not require generating proxy anomalous patterns, for every given window from the time series, two margins are defined around it. More specifically, for each original time series window pattern, two additional patterns are generated, where the first corresponds to the upper margin and the second to the lower one. The generated margins are added to the training set in order to assist the DDA algorithm to adjust the spread of RBF units associated with the normal patterns. The RBF network has three outputs, one for original normal patterns, and the remaining two are associated with the upper and lower margin patterns. During testing, an input pattern is classified as anomalous if all three outputs have values below a certain threshold. Otherwise, the winner-takes-all rule is used to classify patterns as normal or anomalous.

As an example of modifying the algorithm for the task of anomaly detection, in (Li et al., 2008), the RBF network algorithm is modified by combining it with the Support Vector Data Description (SVDD) (Tax and Duin, 2004) algorithm. First, the parameters of Gaussian kernels in the hidden layer of an RBF network are initialised using k-means clustering (Alpaydin, 2004). Next, the hidden layer is used as a fixed feature extractor, where the parameters of its Gaussian kernels are frozen. It transforms the data into a feature space, where the transformed data are then used to train an SVDD model with a linear kernel.

2.5 Summary

Due to the prevalence of raw and high-dimensional data, there is a need for anomaly detection methods that can not only use the benefits of deep learning for extracting high-quality features, but also would have the desirable features of an anomaly detector described in Section 1.1. However, there is a dearth of deep approaches for

anomaly detection that use a one-class objective for training the entire model. The novel approaches described in this thesis are based on the RBF networks, which have certain characteristics that make them an attractive foundation for designing an anomaly detector:

- One of the advantages of local learning is the ability to divide the input space into sub-spaces and learn the distribution of the data in each sub-space. This results in a collection of local learners, each one an expert in a sub-space, which leads to good anomaly detection as apposed to just having one global understanding of the distribution of the normal data. An RBF network has the local learning ability as an inherent part of it and any anomaly detector designed based on RBF networks can benefit from this property in learning the distribution of the normal data.

- It is important for any machine learning algorithm to be computationally efficient and fast both during training and testing, and RBF networks are both fast and efficient models.

- When dealing with raw high-dimensional data, it is common to use deep learning to automate the process of extracting high quality features. Moreover, if a model is designed to be trained using gradient descent, this enables deepening the model, which can facilitate the process of end-to-end training while taking advantage of deep learning. RBF networks are trained using gradient descent and do have the ability to become deep.

- It is important for a model to provide a degree of interpretability, so that the reasoning behind its decisions can be better understood with confidence. Due to their local learning approach, RBF networks easily lend themselves to interpretation (Jin and Sendhoff, 2003).

- There are applications, such as learning from streams, where the distribution of the incoming data can change as a function of time. As a result, the model that is being trained in such a dynamic scenario, needs to be adaptable to new changes. RBF networks are easily adaptable in their architecture to changing concepts (Liu et al., 2020; Han et al., 2011; Rapaka et al., 2003).

However, RBF networks in their original form are not capable of performing one-class classification. To the best of our knowledge, there is no work where the problem of anomaly detection is treated in a semi-supervised manner (i.e., only examples of normal data available for training), where RBF networks are modified to become one-class classifiers that can be trained without needing any other one-class classifier (e.g., OCSVM) on the side, or any type of dataset modification step (e.g., generating anomalous patterns, dataset augmentation).

In this thesis, we modify RBF networks into strong one-class classifiers for the task of semi-supervised anomaly detection by proposing a cost function specifically designed for one-class classification. We will make the resultant one-class classifier stronger by omitting a certain restriction that it inherits from RBF networks, and finally propose a deepening strategy to combine the model with deep learning in order to allow it to work with high-dimensional raw input data.

The following chapters will show how this is done, and demonstrate through evaluation experiments that these newly proposed approaches lead to anomaly detection models that match or exceed the performance of current state of the art approaches while maintaining the advantages of RBF networks outlined above.

CHAPTER

THREE

EXPERIMENT DESIGN

3.1 Introduction

This chapter describes the experimental approach used throughout the remainder of the thesis. The datasets that have been used throughout the thesis will be described in detail, with particular focus on any pre-processing that is performed on them. Finally, the experimental approach and evaluation measures used, which are identical and consistent throughout all of the experiments in this thesis, will be introduced.

3.2 Datasets Used

The datasets used in this thesis fall under two categories: *1) Feature-based datasets* and *2) Raw datasets*.

3.2.1 Feature-based Datasets

Following Emmott et al. (2015), a set of fully labelled classification datasets have been used to simulate anomaly detection scenarios so that performance metrics can be calculated. For each one of these datasets, where possible, different anomaly detection scenarios have been considered, where different normal and anomalous classes are considered in order to add more variety into the experiments.

A subset of the datasets from Emmott et al. (2015) have been used for the experiments in this thesis[1]. All of these datasets come from the UCI machine learning repository

[1]Since every dataset leads to multiple experiments (one-vs-all/ all-vs-one/ difficult scenarios in (Em-

(Dua and Graff, 2017). From these datasets, depending on the nature of the dataset, a number of different anomaly detection scenarios can be generated:

- Binary Classification Datasets: In this case, one class is chosen as normal and the other as anomalous. The normal class is selected based on the recommendations in Emmott et al. (2015), and only examples from this class are used for training.
- Multi-Class Classification Datasets: In this case, three approaches have been followed:
  - *One-vs-all approach*: Instances in a particular class are selected as normal and all other classes are treated as anomalous. Only normal instances are used during training. This approach is interesting, because it explores scenarios where anomalous instances might come from a variety of different distributions, but a compact distribution defines the normal class. If a dataset has K classes, we define K one-vs-all experiments.
  - *All-vs-one approach*: Instances of one class are chosen as anomalous and the instances of the other classes as normal. Only normal instances are used during training. This method explores scenarios where there might be a variety of definitions of the normal data coming from different distributions but a well-defined compact distribution generates the anomalous instances. Again, if a dataset has K classes, K all-vs-one experiments are defined.
  - *Difficult scenarios*: Recommended by Emmott et al. (2015) where an analysis is performed to find the most challenging partition of classes, in terms of separability, into normal and anomalous groups for the datasets they use in their experiments. In this thesis, these partitions are used to define the third set of scenarios in our experiments.

Table 3.1 summarises the feature-based datasets. The number of rows, classes, and features in each one, as well as the number of anomaly detection scenarios extracted are shown. Even through the total number of scenarios will sum up to 64, since three of the difficult scenarios happen to be of one-vs-all or all-vs-one types, the *unique* number of experiment scenarios will be 61.

mott et al., 2015)) a subset of the datasets available was chosen in order to reduce the computation required to run the complete set of experiments.

Table 3.1: The details of feature-based datasets. One-vs-all, all-vs-one, and difficult scenarios can only be generated for multi-class classification datasets

Dataset Name	Rows	Classes	Features	One-vs-All Scenarios	All-vs-One Scenarios	Difficult Scenarios	Total Unique Scenarios
Magic Gamma Telescope	19,021	2	10	-	-	-	1
Spambase	4,602	2	57	-	-	-	1
Skin Segmentation	245,058	2	3	-	-	-	1
Steel Plates Faults	1,941	7	27	7	7	1	14
Image Segmentation	2,311	7	18	7	7	1	15
Page Blocks Classification	5,473	5	10	5	5	1	10
Statlog (Landsat Satellite)	6,436	6	36	6	6	1	13
Waveform Database Generator (Version 1)	5,000	3	21	3	3	1	6

3.2.2 Raw Datasets

Raw datasets do not have a set of well-behaved features and can come in a variety of types. The two flavours selected for the experiments in this thesis are image and time series datasets. The well-known MNIST and Fashion-MNIST image datasets are used:

- MNIST[2] (LeCun and Cortes, 2010) contains a training set of 60,000 gray-scale handwritten digit images of size 28x28, and a test set of 10,000 similar images.
- Fashion-MNIST[3] (Xiao et al., 2017) contains a training set of 60,000 greyscale images of 28x28 pixel images of different clothing items (e.g. t-shirt/top, trouser, pullover, dress), and a test set of 10,000 similar images.

Multiple anomaly detection scenarios are generated using these datasets. In each scenario one class is considered as normal and another class as anomalous. These pairs of classes (shown in Table 3.2) were selected to cover simple and challenging scenarios. For example, for MNIST there is a simple scenario where digit 0 is considered normal and digit 1 is anomalous, and similarly for Fashion MNIST there is a scenario where *T-shirts/tops* are normal and *boots* are anomalous. Images from these pairs of classes are easily discernible, and high performance across most of the algorithms is expected. More challenging scenarios have also been generated. For instance from MNIST there is a scenario where digit 4 is normal and digit 9 is anomalous, and for Fashion MNIST there is a scenario where *T-shirts/tops* are normal and *shirts* are anomalous. These pairs of image types are not easy to separate as they are so similar.

One real-world highly-imbalanced anomaly detection time series dataset is used: the

[2] `http://yann.lecun.com/exdb/mnist/`
[3] `https://www.kaggle.com/zalando-research/fashionmnist`

MIT-BIH Arrhythmia Dataset[4] (Moody and Mark, 2001; Goldberger et al., 2000). This dataset contains 48 half-hour excerpts of two-channel ambulatory ECG recordings, obtained from 48 subjects. Out of the 19 anomalous classes in the dataset, the 4 most common classes are used to make four binary anomaly detection scenarios. We have also added a one-vs-all scenario, where all four classes of anomalies are combined into one class. This generates a total of five anomaly detection scenarios for this dataset.

The experiment scenarios have been summarised for all three raw datasets in Table 3.2.

Table 3.2: For each dataset, the class label for the normal class and the anomalous class are shown. As a result, there are 6 experiment scenarios for MNIST, 4 for Fashion MNIST, and 4 for the MIT-BIH Arrhythmia database. The class labels are accompanied by their descriptions in parentheses, where needed.

Dataset	Normal	Anomalous
MNIST	0	1
	7	1
	4	9
	7	9
	9	4
	9	7
Fashion-MNIST	T (T-shirts/tops)	B (Ankle boots)
	T (T-shirts/tops)	S (Shirts)
	Sa (Sandals)	Sn (Sneakers)
	B (Ankle boots)	Sa (Sandals)
MIT-BIH Arrhythmia Database	N (Normal)	L (Left bundle branch block beat)
	N (Normal)	R (Right bundle branch block beat)
	N (Normal)	V (Premature ventricular contraction)
	N (Normal)	/ (Paced beat)

3.3 Dataset Pre-processing

This section describes the pre-processing applied to each of the datsets used in the thesis.

3.3.1 Feature-based Datasets Pre-processing

The only pre-processing step for all eight datasets under this category is normalisation of all feature values. This was achieved by independently and linearly re-scaling the

[4]https://physionet.org/content/mitdb/1.0.0/

values for each feature to the range $[0, 1]$, using min-max scaling.

3.3.2 Image Dataset Pre-processing

For the image datasets (i.e., MNIST and Fashion-MNIST), the only pre-processing stage is to normalise the pixel values to the range $[0, 1]$. Simple linear re-scaling from the range $[0, 255]$ to the range $[0, 1]$ is used.

3.3.3 MIT-BIH Arrhythmia Dataset Pre-processing

This is a bi-channel ECG dataset. The data is collected from 47 patients and for each patient we have 30 minutes (360 samples/sec) worth of sensory readings. Every heart-beat within a given 30 minute recording is labelled. The majority of the dataset is comprised of the normal class. For the anomalies, we have chosen the four most frequent anomaly types in the dataset. The details about these classes and the number of available heartbeat samples per class are available in Table 3.3. For a healthy human heart, the ECG waveform for each heart beat resembles the one depicted in Fig. 3.1: it is composed of the P, Q, R, S, T, and U waves and beat detection is usually performed by searching for each R-peak or QRS complex.

Table 3.3: Classes used from the MIT-BIH Arrhythmia Dataset

Class Label	Class Description	Number of Available Heartbeats in the Class
'N'	Normal heartbeat	72,813
'L'	Left bundle branch block beat	8,075
'R'	Right bundle branch block beat	7,259
'V'	Premature ventricular contraction	7,129
'/'	Paced beat	7,028

In the dataset the positions of the R-peaks are provided and since the ECG recordings are in the form of long recordings of consecutive heartbeats, these R-peak positions can be used to extract the individual heart-beats from the recordings, along with the provided class labels. Some visualisations of sample heart-beats from normal, left bundle branch block beat, right bundle branch block beat, premature ventricular contraction, and paced beat classes are shown in Fig. 3.2. Below, we will cover the pre-processing steps that we have taken for this dataset, including the heart-beat segmentation and padding method, which are also used to produce the heart-beat visualisations in Fig. 3.2.

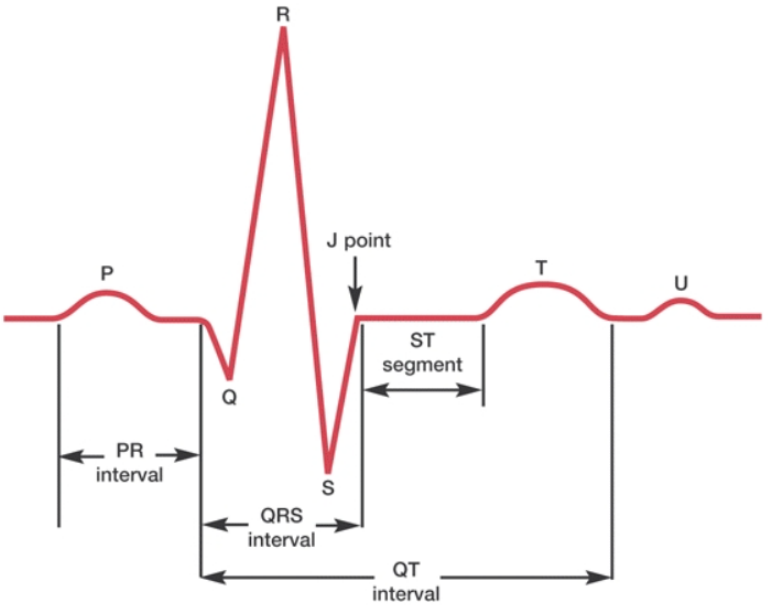


Figure 3.1: The basic pattern of electrical activity across the heart (Ashley and Niebauer, 2004)

As the original data is collected in the range $[0, 2047]$, it is normalised by linearly scaling the data from $[0, 2047]$ to $[0, 1]$. Next, in order to segment individual heart-beats in these ECG recordings, following the approach in (Xu et al., 2019), we use the mid-point of 2 consecutive R-peaks, as the boundary between each pair of heartbeats. This way, the segmented part will be guaranteed to include the QRS complex (i.e., the most important part of a heart-beat (Zidelmal et al., 2012)) for that heart-beat at the very least and there will be no overlaps between the segments. After isolating each heart-beat, the heart-beat segments will be of different lengths. Following Xu et al. (2019), we truncate or pad all heart-beats to a fixed length, L. The value of L is chosen as the 95^{th} percentile of the heart-beat lengths in the dataset, in this case 417. When heart-beats are padded we use a pad value of 0.5 (equivalent to a value of 1,024 in the original range). The direction for padding and truncating the heart-beats is in the right hand side. Finally, in order to lower the dimensionality of the data and accelerate the training time across all of our experiments, all heart-beats have been down-sampled to 187 Hz from the original sampling rate of 360 Hz.

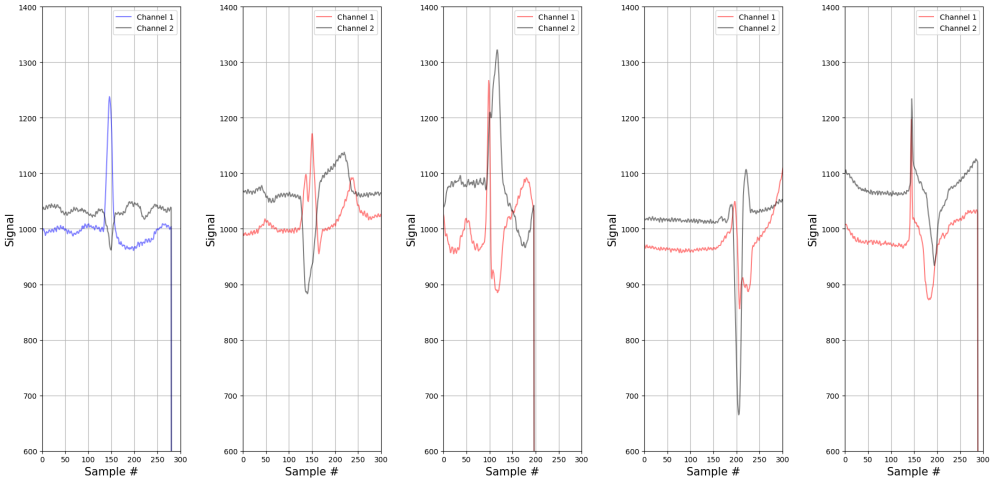


Figure 3.2: Individual heart-beat samples (from left to right) from normal, left bundle branch block beat, right bundle branch block beat, premature ventricular contraction, and paced beat classes

3.4 Experimental Method

To evaluate all models we perform an approach based on boot-strapping that makes maximum use of the small number of anomalous samples available in our datasets. At each iteration, for a given set of hyper-parameters in a given model, we randomly select 80% of all normal instances in the dataset (with no replacement) to train the model. The remaining 20% of normal instances is then mixed with all of the anomalous instances to form the test set. For a given set of hyper-parameters in a given model, we perform 10 such iterations of training and testing and measure the area under the ROC curve (AUC) on the test set at each iteration. These are averaged to give the overall model performance for that set of hyper-parameters.

We perform hyper-parameter tuning using a grid search that repeats the above process for each hyper-parameter combination. We report the best averaged AUC from the grid search for the corresponding experiment. We are aware that reporting the performance of the model with the best set of hyper-parameters over-estimates the generalisation performance of the models (known as the problem of *many comparisons in induction algorithms* (Tsamardinos et al., 2014)). However, as our goal is a relative comparison of algorithms, rather than an absolute estimate of generalisation error, and all algorithms are treated the same, this is an appropriate approach that makes better use of limited anomalous samples than measuring performance on a separate hold-out test set. These hyper-parameters and the values chosen for the grid search,

will be discussed in detail in later chapters.

In the next chapter, the first type of one-class RBF networks proposed in this thesis is introduced and its performance is compared with state of the art anomaly detectors in a large set of benchmark experiments.

CHAPTER

FOUR

RADIAL BASIS FUNCTION DATA DESCRIPTOR NETWORKS

4.1 Introduction

In this chapter, we propose a novel approach to semi-supervised anomaly detection based on the Radial Basis Function (RBF) network (Alpaydin, 2004; Bishop, 2007). This approach relies on a newly defined cost function that allows it to learn only from instances of the normal class. RBF networks have certain interesting features that make them an attractive foundation for any anomaly detector. For instance, the local learning approach makes RBF networks an attractive choice for anomaly detection where anomalies come from multiple distributions. In addition, the use of gradient descent allows these networks to be deepened, and this adds to the effectiveness of RBF networks when dealing with high-dimensional raw input data. Finally, because of their local learning approach, RBF networks easily lend themselves to interpretation (Jin and Sendhoff, 2003), and are adaptable to changing concepts (Liu et al., 2020; Han et al., 2011), while their training is fast and efficient. All of this makes any anomaly detector that is based on RBF networks, an attractive proposition.

The following research question is investigated in this chapter:

- *Is it possible to turn the Radial Basis Function (RBF) network into an effective anomaly detector that can learn only from the normal class?*

Motivated by RBF networks, a one-class RBF network named the Radial Basis Function Data Descriptor (RBFDD) network is proposed in this chapter. The RBFDD network turns RBF networks into effective anomaly detectors by modifying their structure as well as using a novel one-class cost function. A large-scale benchmark experiment suggests that RBFDD networks, on average, out-perform state of the art approaches to anomaly detection. The contributions of this chapter have been published in (H. Bazargani and Mac Namee, 2018).

This chapter proceeds as follows. Section 4.2 describes the proposed RBFDD anomaly detector. Section 4.3 describes an evaluation experiment that is designed to assess the performance of the RBFDD approach and compares it to state of the art, and Section 4.4 discusses the results of this experiment. Finally, Section 4.5 concludes the chapter.

4.2 The Radial Basis Function Data Descriptor Network

We have chosen RBF networks (described in detail in Section 2.3) as the basis for building an anomaly detector due to the inherent capability of the RBF network to learn the underlying distribution of training data. An RBF network, however, is not capable of performing a one-class classification task, so we need to introduce some modifications to it to adapt it to this task. The intuition behind our approach is that we would like our Gaussian kernels to gather around the subspace in the input space where the normal data reside, and then we would like for our model to output a high value for any data appearing in that region. However, for any data point appearing in other regions of the input space, we would like for our model to output a low value, thus distinguishing between the normal and anomalous data. We would like to make sure our kernels concentrate around the normal region of the input space, covering all of it, while excluding other regions. By regularising the spreads, s_h, of the Gaussians (i.e., kernels), we strive to converge to the *most compact* set of Gaussians that cover the normal region of the input space, while avoiding other unwanted regions where we believe the anomalous data reside. During testing, for a given input instance, X_i, the output of the model would be high if X_i belongs to the normal region, and low otherwise. The structure of the proposed RBFDD network, is shown in Fig. 4.1.

The main contribution of this chapter is the modifications that we have made to the cost function in Eq.(2.3), to transform the standard RBF network into the Radial Basis

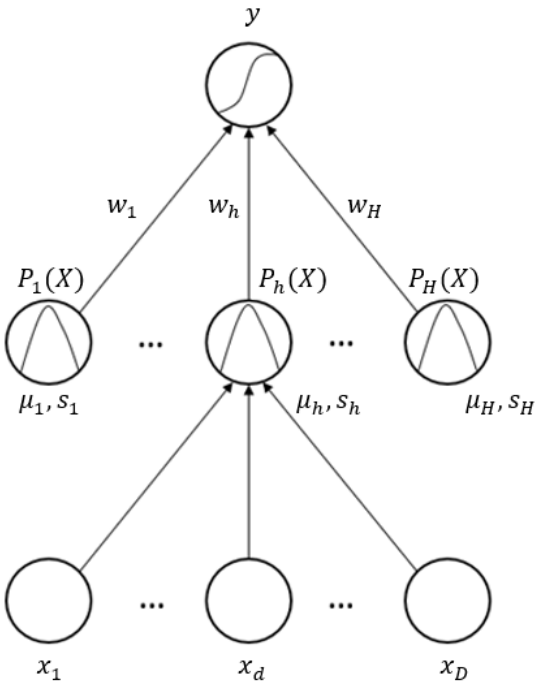


Figure 4.1: The RBFDD network.

Function Data Descriptor (RBFDD) network anomaly detector. We propose the following cost function for the RBFDD network, which is minimised for mini-batches of size N:

$$E = \sum_{i=1}^{N} \left(\frac{1}{2} \left[(1 - y_i)^2 + \beta \sum_{h=1}^{H} s_h^2 + \lambda \sum_{h=1}^{H} w_h^2 \right] \right) \tag{4.1}$$

The cost function is a weighted summation of three main terms. Variable y_i in the first term, $(1 - y_i)^2$, is the output of the RBFDD network for a given input data X_i. This first term in the cost function encourages the network training process to learn a model that outputs a value as close as possible to 1 for instances belonging to the normal class. The second term is the squared L2-norm of the spreads of the Gaussians, or $||s||_2^2$. This term introduces our optimisation goal of having the most compact set of Gaussians possible to represent the normal data.

The third term is the squared L2-norm of the weight vector connecting the RBF hidden layer units (i.e., Gaussians) to the output unit, or $||w||_2^2$. This stops the weights from becoming so large that they would actually ignore the outputs from the hidden units. This also makes the RBFDD network robust to outliers in the training set (Goodfellow et al., 2016). In the output node of the network we use the hyperbolic tangent, $tanh$, non-linear activation function proposed in (Orr and Müller, 1998) (i.e.,

$1.7159 \times tanh(\frac{2x}{3})$), as it avoids saturation. The architecture of the RBFDD network is shown in Fig. 4.1.

We argue that a gradient descent training process based on the minimisation of Eq.(4.1) can result in the most compact set of Gaussians, whose collective output is still high for the normal region of the input space and low anywhere else (i.e., where we believe the anomalies would appear). Here, we will explain how the RBFDD network can learn the normal class, using standard back-propagation (Hecht-Nielsen, 1988) and mini-batch gradient descent (Alpaydin, 2004).

For a given input data, X_i, let us consider Eq.(4.1), and the following:

$$y_i = 1.7159 \times tanh\left(\frac{2 \times z(X_i)}{3}\right) \tag{4.2}$$

$$z(X_i) = \sum_{h=1}^{H} w_h \times P_h(X_i) \tag{4.3}$$

where w_h is the weight connecting hidden unit h to the output unit and $P_h(X_i)$ for a given input X_i data is defined as:

$$P_h(X_i) = \exp\left(-\frac{||X_i - \mu_h||^2}{2s_h^2}\right) \tag{4.4}$$

where μ_h and s_h denote the center and the standard deviation (i.e., spread) of the local unit h, defining a radially symmetric basis function; and X_i is the i^{th} input data.

Based on the application of the back-propagation of error algorithm using gradient descent, the following update rules to learn the parameters of the RBFDD network (i.e., positions, spreads of the Gaussians, and the weights) have been derived[1]:

$$w_h \leftarrow w_h - \eta\left[(1.1439)\,(y_i - 1)\left(1 - tanh\left(\frac{2z(X_i)}{3}\right)^2\right)p_h + \lambda w_h\right] \tag{4.5}$$

$$\mu_h \leftarrow \mu_h - \eta\left[(1.1439)(y_i - 1)\left(1 - tanh\left(\frac{2z\,(X_i)}{3}\right)^2\right)\left(w_h \times p_h \frac{(X_i - \mu_h)}{s_h^2}\right)\right] \tag{4.6}$$

[1]For simplicity, the update rules have been shown for when an individual input data X_i is fed into the RBFDD network and the output y_i is generated by the network, and generalisation of these update rules to a mini-batch learning scenario is straight-forward.

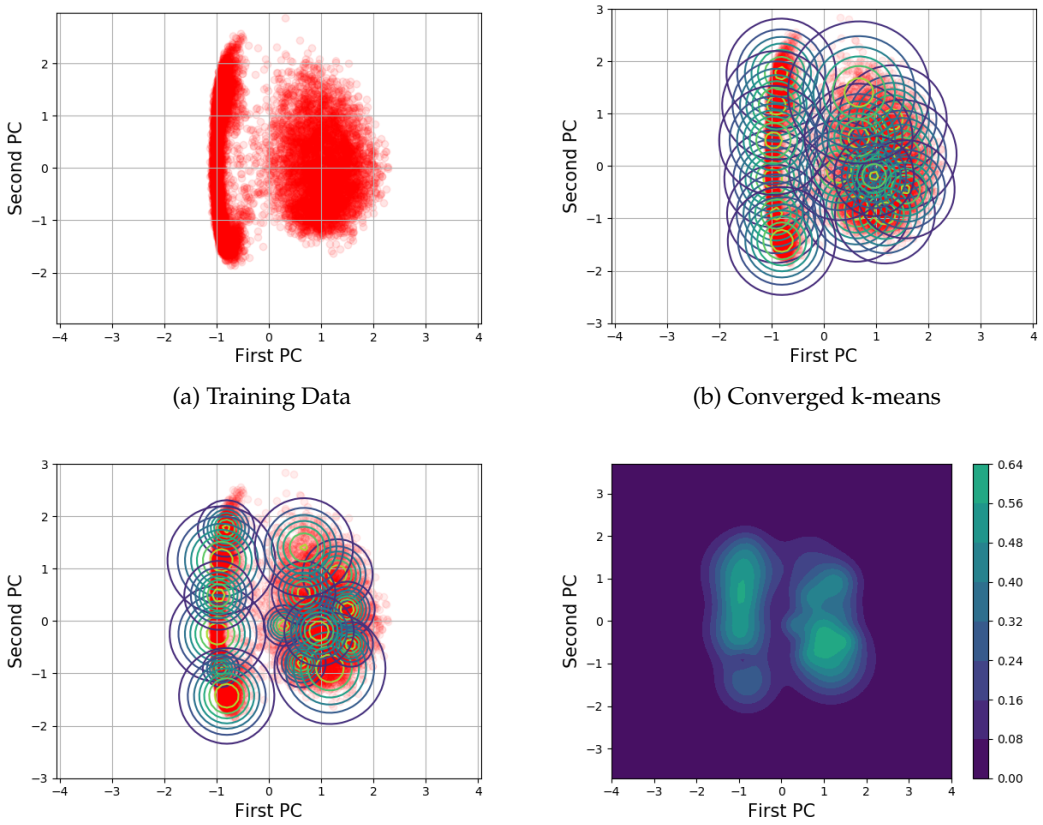
$$s_h \leftarrow s_h - \eta \left[(1.1439)(y_i - 1) \left(1 - tanh \left(\frac{2z\left(X_i\right)}{3} \right)^2 \right) \left(w_h \times p_h \frac{||X_i - \mu_h||^2}{s_h^3} \right) + \beta s_h \right] \tag{4.7}$$

In our proposal, the unsupervised pre-training phase has remained intact and identical to that of the standard RBF network (i.e., k-means clustering).

4.2.1 Interpreting RBFDD Networks

To illustrate the behaviour of an RBFDD network we present a simple example based on the MNIST dataset of handwritten digits (LeCun and Cortes, 2010). In this example 15 Gaussians are used in the RBFDD network and images of both digits 0 and 1 are considered as the normal class. So that the behaviour of the network can be visualised, the dimensionality of the input data is reduced to two, using Principal Component Analysis (PCA) (Alpaydin, 2004). Fig. 4.2a shows the training dataset of normal instances (images of digits 0 and 1). Fig. 4.2b shows the output of the k-means algorithm after the pre-training phase, which provides the RBFDD network with an initial set of well positioned Gaussian kernels. Fig. 4.2c shows the compact set of kernels covering the normal region after the RBFDD network is trained. Finally, Fig. 4.2d shows the decision surface that has been learned, where brighter colors correspond to higher outputs by the RBFDD network.

The learned features in an RBFDD network (i.e., the positions of the Gaussians, their spreads, and the learned weights) can give the model a degree of interpretability. The Gaussian positions and spreads tell us something about the exemplars that the model has found to represent the normal class within the data. Moreover, the learned weight vector, can tell us about the importance of each Gaussian in explaining the underlying distribution of the normal data. As a result, we argue that the learned features in the RBFDD network can give us a good level of tangible interpretability. As an example, an RBFDD network is trained on images of digit 7 from the MNIST dataset (in the original input space and without any dimensionality reduction) as the normal class. The hyper-parameters chosen for training this RBFDD network are: (number of kernels (H), η, β, λ) = (7, 0.01, 0.0001, 0.0001), and the number of epochs is set to 100. In



(a) Training Data

(b) Converged k-means

(c) Converged RBFDD

(d) Decision Boundary

Figure 4.2: Considering images of digits 0 and 1 from the MNIST dataset as class *normal* instances for training and using the first two principal components, we have the visualisations of the transformed training data after applying PCA in red circles, the Gaussians after k-means pre-training, trained RBFDD network, and the decision surface of the trained RBFDD network.

Fig. 4.3, the centres of the 7 Gaussians are visualised, and the final learned associated weights for each Gaussian are shown at the top of each sub-figure. It is interesting that the Gaussians with highest weight values tend to have a centre that does not follow a crisp and clear definition of the normal class (i.e., images of digit 7), but rather a shaded pattern which can help them generalise better to a higher variety of unseen patterns of normal data (i.e., images of digit 7). Interestingly, the centre of the Gaussian in the middle seems somewhat distorted, and the minimisation of the proposed cost function in Eq.(4.1), has forced the corresponding weight of this particular Gaussian to shrink to a very small value (w = -0.0910). It is as if the RBFDD network is pruning this Gaussian from its architecture, as a somewhat redundant component for describing the distribution of the normal class.

In the next section, we will describe a set of experiments conducted to compare the performance of the proposed RBFDD network approach with state of the art algo-

rithms for anomaly detection.

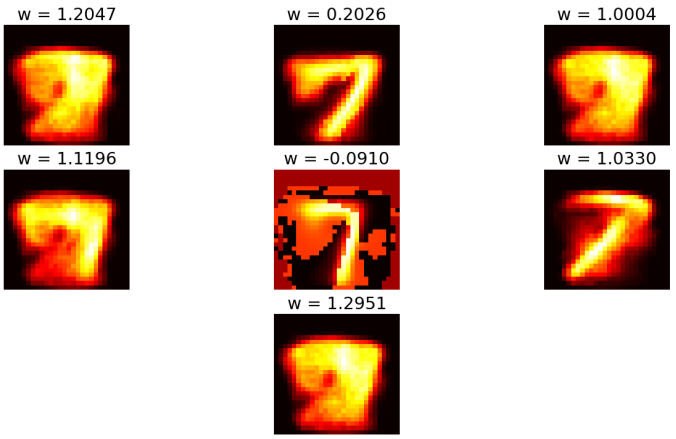


Figure 4.3: The centres of 7 Gaussians of a trained RBFDD network with their corresponding learned weights, where the normal class consists of images of digit 7 from the MNIST dataset.

4.3 Evaluation Experiment

Following Emmott et al. (2015) we use fully labelled classification datasets to simulate anomaly detection scenarios so that performance metrics can be calculated. The datasets used are the set of feature-based datasets introduced in Section 3.2.1. For each dataset used, where possible, we have considered different anomaly detection scenarios, where we consider different normal and anomalous classes to add more variety into our experiments. We use the scenarios and datasets described in Section 3.2.1.

The state of the art algorithms for anomaly detection that are used are: One-Class Support Vector Machines (OCSVMs), Auto-Encoders (AENs), Gaussian Mixture Models (GMMs), and Isolation Forests (iForests).

Among these state of the art algorithms, GMMs (described in Section 2.2.2) in particular are similar to RBFDD networks. There are some key differences, however, that are worth noting:

- A GMM is a probabilistic approach and its goal is to learn a model such that the likelihood of observing the training data is maximised. In an RBFDD network, there is no probabilistic interpretation.
- The mixing probabilities (i.e., weights associated with the Gaussians) in a GMM

are computed using class posterior probabilities and are described by a probability distribution, and as a result these weights always sum to one. The weights in an RBFDD network do not have this constraint, however, they are regularised.

- There is no constraint on the shape of the Gaussians in a GMM as they can be radial or elliptical. In an RBFDD network, the Gaussians are constrained to be radial, however, the size of these Gaussians (i.e., their variances) are regularised.
- The optimisation algorithm for training a GMM is Expectation Maximisation (EM) (Bishop, 2007), whereas, for an RBFDD network the model is trained through the application of the back-propagation of error algorithm using gradient descent.

We evaluate each algorithm, dataset and scenario combination using the experimental design described in Section 3.4.

The tunable hyper-parameters used to generate results, for each algorithm across all the datasets and scenarios, are presented in Table 4.1. It is important to mention that the values for H in the hyper-parameter grid search are determined by the dimensionality of each dataset, and in our experiments they do not exceed the dimensionality of the input. The range tested for H starts with 1 and then increases in steps of 5 all the way up to the dimensionality of the data, D. In the case of the Page Blocks Classification and Magic Gamma Telescope datasets we reduce the step size to 2, and for the Skin Segmentation dataset to 1, as the dimensionality of these datasets is rather low. As a result, the RBFDD and AEN always tend to compress the input data in their hidden representation. Similarly, the number of Gaussians in the GMM is also bounded by the dimensionality of the input, D. However, in the case of the Isolation Forest, we have selected the number of estimators, independent of the input dimensionality, across all datasets. In the case of OCSVM, ν is the upper bound on the fraction of training errors and a lower bound of the fraction of support vectors, and γ is the kernel coefficient.

The output of the RBFDD network, y, needs to be thresholded for anomaly detection to take place, as is the case for most anomaly detection approaches. Any of the common thresholding schemes used with other anomaly detection algorithms (Spence et al., 2001; Pickands III et al., 1975) can be used with RBFDD networks. As the thresholding method is not the focus of this thesis, we have used the Area Under the Curve (AUC)

Table 4.1: Hyper-parameter ranges for each algorithm

Algorithms	Hyper-parameters	Investigated Hyper-Parameters
RBFDD	number of kernels (H)	[1, 5, 10, ..., D]
	η	[0.01, 0.001, 0.0001]
	β	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
	λ	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
OCSVM	ν	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
	γ	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
AEN	number of hidden units (H)	[1, 5, 10, ..., D]
	η	[0.01, 0.001, 0.0001]
	hidden layer activation functions	[sigmoid, relu]
	output layer activation functions	[sigmoid, relu, linear]
	cost functions	[mean squared error, cross entropy]
GMM	number of kernels (H)	[1, 5, 10, ..., D]
Isolation Forest	number of estimators	[100, 200, 500, 800, 1000]

of the ROC curve (Japkowicz and Shah, 2011) as the main evaluation metric. This allows us to avoid defining a specific thresholding function to use with each model, but still measures the ability of a model to distinguish between normal and anomalous classes.

4.4 Results & Discussion

The performances of each algorithm on each anomaly detection scenario for each dataset were compared and ranked. Tables 4.2 to 4.5 show results from the binary, one-vs-all, all-vs-one, and difficult scenarios. The distribution of the ranks for each algorithm for all anomaly detection scenarios are shown in Fig. 4.4, and the average rank scores are summarised in Table 4.6. The average ranks in Table 4.6 show that the RBFDD network has the best average rank over all experiments [2].

Table 4.2: Experiment Scenarios and Results for Binary-Class Datasets

	(Normal-Anomalous) Class Labels	RBFDD	OCSVM	AEN	GMM	iForest
Gamma	('g'-'h')	0.77450 (4)	0.74096 (5)	0.81670 (2)	0.86147 (1)	0.77516 (3)
Spambase	(0-1)	0.63396 (5)	0.70562 (4)	0.82311 (3)	0.85171 (1)	0.84728 (2)
Skin	(2-1)	0.84751 (4)	0.70549 (5)	0.92357 (1)	0.88891 (3)	0.89098 (2)
Average Rank		4.3 (4)	4.7 (5)	2.0 (2)	**1.7 (1)**	2.3 (3)

[2]Since the experiments in this chapter are a subset of a larger set in Chapter 5, we will describe and conduct the statistical significance tests in Chapter 5

Table 4.3: Experiment Results for Multi-Class Datasets: One-vs-All Scenarios

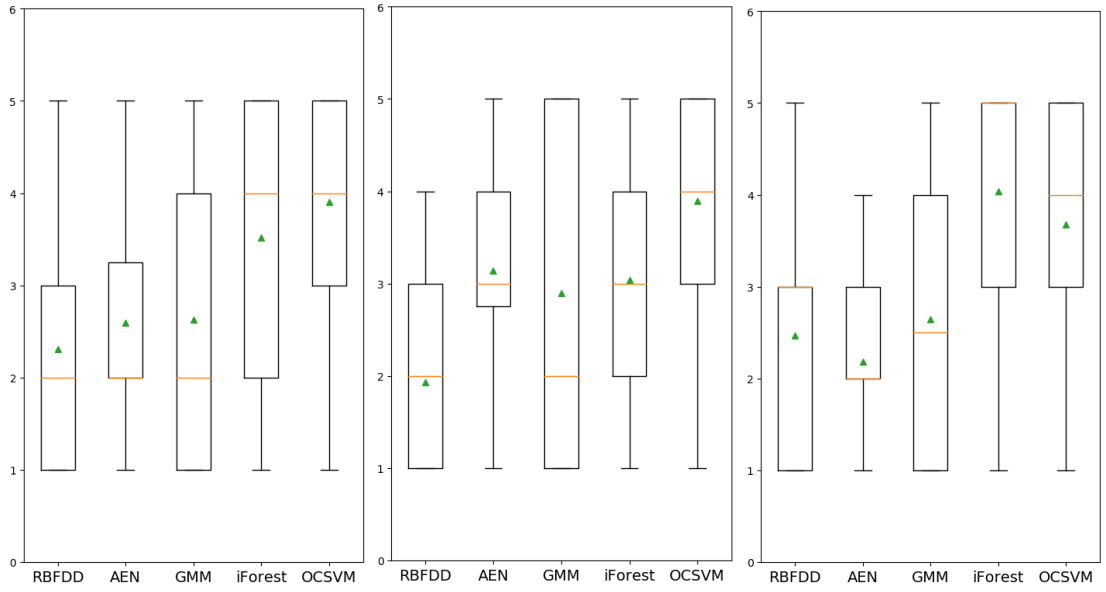
	Normal Class Label	RBFDD	OCSVM	AEN	GMM	iForest
Fault	Pastry	0.87388 (2)	0.85343 (5)	0.86261 (4)	0.88637 (1)	0.87033 (3)
	Z_Scratch	0.97667 (1)	0.97224 (3)	0.96593 (4)	0.97570 (2)	0.92684 (5)
	K_Scatch	0.98721 (2)	0.96846 (5)	0.98017 (3)	0.98888 (1)	0.98001 (4)
	Stains	0.99480 (2)	0.99184 (4)	0.99325 (3)	0.98924 (5)	0.99766 (1)
	Dirtiness	0.98214 (1)	0.93395 (4)	0.93515 (3)	0.97002 (2)	0.91264 (5)
	Bumps	0.82843 (3)	0.82729 (4)	0.82484 (5)	0.83205 (1)	0.83173 (2)
	Other_Faults	0.62015 (2)	0.59953 (4)	0.59474 (5)	0.65096 (1)	0.61955 (3)
Image Segmentation	BRICK	0.99394 (3)	0.97861 (5)	0.99328 (4)	0.99920 (1)	0.99815 (2)
	SKY	1.00000 (1)	1.00000 (1)	1.00000 (1)	0.99881 (4)	0.99810 (5)
	FOLIAGE	0.93342 (2)	0.87113 (5)	0.91278 (3)	0.95734 (1)	0.90678 (4)
	CEMENT	0.94126 (1)	0.90427 (4)	0.92766 (3)	0.94099 (2)	0.90318 (5)
	WINDOW	0.91362 (3)	0.86126 (5)	0.90077 (4)	0.95195 (1)	0.93139 (2)
	PATH	0.99823 (2)	0.98943 (4)	0.99440 (3)	0.99974 (1)	0.98015 (5)
	GRASS	1.00000 (1)	0.99887 (4)	0.99993 (2)	0.99848 (5)	0.99986 (3)
LandSat	red soil	0.98994 (3)	0.98820 (4)	0.99423 (1)	0.98729 (5)	0.99136 (2)
	cotton crop	0.99469 (1)	0.98829 (3)	0.98255 (4)	0.96337 (5)	0.98876 (2)
	grey soil	0.97932 (1)	0.97379 (3)	0.97434 (2)	0.95708 (5)	0.97054 (4)
	damp grey soil	0.92693 (1)	0.92038 (4)	0.92378 (3)	0.87460 (5)	0.92636 (2)
	soil with vegetation stubble	0.91687 (1)	0.87032 (5)	0.88969 (3)	0.87151 (4)	0.90125 (2)
	very damp grey soil	0.96146 (2)	0.95407 (5)	0.95853 (3)	0.95448 (4)	0.96252 (1)
Page	text	0.93654 (3)	0.90640 (5)	0.94363 (2)	0.95736 (1)	0.92717 (4)
	horiz. line	0.84573 (3)	0.61670 (5)	0.74304 (4)	0.93233 (1)	0.87221 (2)
	graphic	0.81629 (4)	0.92336 (3)	0.93125 (2)	0.94769 (1)	0.57743 (5)
	vert. line	0.97299 (1)	0.91514 (5)	0.94404 (4)	0.96387 (2)	0.95872 (3)
	picture	0.83315 (2)	0.81884 (3)	0.87817 (1)	0.79179 (5)	0.81032 (4)
Wave	0	0.86507 (1)	0.86207 (2)	0.84739 (4)	0.84634 (5)	0.86151 (3)
	1	0.91489 (2)	0.91239 (3)	0.90627 (4)	0.89499 (5)	0.92074 (1)
	2	0.91360 (3)	0.91934 (2)	0.90830 (4)	0.90010 (5)	0.92011 (1)
Average Rank		**1.9 (1)**	3.9 (5)	3.1 (4)	2.9 (2)	3.0 (3)

Table 4.4: Experiment Results for Multi-Class Datasets: All-vs-One Scenarios

	Anomalous Class Label	RBFDD	OCSVM	AEN	GMM	iForest
Fault	Pastry	0.74711 (1)	0.66176 (2)	0.72821 (3)	0.66243 (4)	0.58524 (5)
	Z_Scratch	0.74209 (3)	0.71712 (1)	0.77584 (2)	0.73010 (4)	0.55213 (5)
	K_Scatch	0.94678 (3)	0.91684 (5)	0.92212 (4)	0.94783 (2)	0.94893 (1)
	Stains	0.86556 (1)	0.80848 (4)	0.81068 (3)	0.84424 (2)	0.74532 (5)
	Dirtiness	0.73767 (2)	0.70261 (3)	0.78197 (1)	0.65360 (4)	0.60343 (5)
	Bumps	0.73194 (1)	0.62443 (4)	0.65910 (2)	0.64203 (3)	0.51147 (5)
	Other_Faults	0.73409 (3)	0.67300 (4)	0.74380 (2)	0.78232 (1)	0.66463 (5)
Image Segmentation	BRICK	0.54813 (3)	0.46279 (5)	0.93805 (2)	0.95622 (1)	0.48590 (4)
	SKY	0.99984 (3)	1.00000 (1)	0.99994 (2)	0.99626 (4)	0.98901 (5)
	FOLIAGE	0.78690 (4)	0.82876 (3)	0.90069 (1)	0.82879 (2)	0.72881 (5)
	CEMENT	0.53020 (4)	0.49775 (5)	0.83910 (2)	0.95943 (1)	0.58751 (3)
	WINDOW	0.54775 (4)	0.62737 (3)	0.72612 (2)	0.83608 (1)	0.46808 (5)
	PATH	0.64221 (5)	0.72225 (4)	0.97240 (1)	0.91111 (2)	0.74520 (3)
	GRASS	0.99210 (4)	0.99801 (3)	0.99910 (2)	0.99991 (1)	0.92048 (5)
LandSat	red soil	0.91750 (3)	0.84476 (4)	0.98846 (1)	0.96100 (2)	0.78044 (5)
	cotton crop	0.99693 (2)	0.99389 (4)	0.99901 (1)	0.99622 (3)	0.99307 (5)
	grey soil	0.94770 (3)	0.95172 (1)	0.95034 (2)	0.61131 (5)	0.86509 (4)
	damp grey soil	0.63347 (1)	0.46886 (4)	0.57443 (2)	0.52522 (3)	0.39395 (5)
	soil with vegetation stubble	0.87001 (2)	0.74329 (5)	0.79659 (4)	0.90894 (1)	0.79804 (3)
	very damp grey soil	0.73539 (3)	0.65526 (4)	0.83786 (1)	0.58132 (5)	0.73832 (2)
Page	text	0.72955 (2)	0.46192 (5)	0.66032 (3)	0.79199 (1)	0.60372 (4)
	horiz. line	0.92081 (3)	0.89531 (5)	0.95125 (2)	0.96770 (1)	0.91872 (4)
	graphic	0.82382 (4)	0.73967 (5)	0.93934 (3)	0.97463 (1)	0.94970 (2)
	vert. line	0.95057 (1)	0.92444 (4)	0.94351 (2)	0.94242 (3)	0.88595 (5)
	picture	0.92968 (1)	0.91352 (3)	0.91725 (2)	0.87087 (4)	0.83920 (5)
Wave	0	0.83539 (1)	0.69253 (5)	0.78260 (2)	0.76900 (3)	0.72886 (4)
	1	0.90888 (1)	0.80230 (4)	0.80481 (3)	0.74791 (5)	0.81359 (2)
	2	0.89647 (1)	0.83358 (3)	0.82682 (4)	0.76207 (5)	0.83522 (2)
Average Rank		2.5 (2)	3.7 (4)	**2.2 (1)**	2.6 (3)	4.0 (5)

Table 4.5: AUC Scores and Ranks in Multi-Class Datasets: Difficult Scenarios. As mentioned in Chapter 3, three of these five difficult scenarios are already included in one-vs-all or all-vs-one scenarios, namely, the difficult scenarios in the Wave, Page, and Fault datasets

	(Normal-Anomalous) Class Labels	RBFDD	OCSVM	AEN	GMM	iForest
Fault	(Pastry,Z_Scratch,K_Scratch,Stains,Dirtiness,Bumps) -(other_Faults)	0.73409 (3)	0.67300 (4)	0.74380 (2)	0.78232 (1)	0.66463 (5)
Image Segmentation	(BRICK,FOLIAGE,CEMENT,GRASS) -(SKY,WINDOW,PATH)	0.81797 (3)	0.73711 (5)	0.88260 (1)	0.86549 (2)	0.77386 (4)
LandSat	(red soil,cotton crop,grey soil,very damp grey soil) -(damp grey soil,soil with vegetation stubble)	0.71909 (2)	0.53770 (5)	0.60676 (4)	0.73845 (1)	0.63110 (3)
Page	(text) -(horiz. line,graphic,vert. line,picture)	0.93654 (3)	0.90640 (5)	0.94363 (2)	0.95736 (1)	0.92717 (4)
Wave	(1,2) -(0)	0.83539 (1)	0.69253 (5)	0.78260 (2)	0.76900 (3)	0.72886 (4)
Average Rank		2.4 (3)	4.8 (5)	2.2 (2)	**1.6 (1)**	4.0 (4)



(a) All Scenarios (b) One-vs-All (c) All-vs-One

Figure 4.4: The average rank of the algorithms across the experiments. On each box plot the median ranks are shown via the horizontal orange lines and the mean ranks are shown using green triangles.

Table 4.6: Average Rank Across All Experiments

	RBFDD	AEN	GMM	iForest	OCSVM
Average Rank	**2.3 (1)**	2.6 (2)	2.7 (3)	3.5 (4)	3.9 (5)

Fig. 4.4 also shows the distribution of ranks of the different approaches on the one-vs-all and all-vs-one scenarios (distributions for the other scenarios are not included as there are too few results to generate meaningful distributions). While the overall pattern is largely the same for these subsets as for the overall results, it is worth noting that there is a marked difference in the performance of the AEN models on the one-vs-all scenarios—performance is relatively poor—and on the all-vs-one scenarios—performance is quite good. In addition, the poor performance of OCSVM is somewhat surprising, however, it is in line with previous benchmarks, for example (Emmott et al., 2015). As observed in Table 4.6, the empirical results show that the RBFDD network, overall, is out-performing the other benchmark algorithms across all the experiments, suggesting the effectiveness of the proposed RBFDD approach and the cost function, Eq.(4.1), defined for training it.

As a final remark, it is worth noting that the RBFDD network does not perform well in the binary-class datasets where there is one normal and one anomalous class— GMM with a rank of 1.7 performs best (Table 4.2). In one-vs-all scenarios, however, the RBFDD network is performing well and coming in first with a rank of 1.9 (Table 4.3).

The one-vs-all scenarios are particularly important for anomaly detection as it is likely that anomalies will arise from very different data distributions. In all-vs-one scenarios the AEN is coming in first, out-performing RBFDD slightly, with a rank of 2.2 (Table 4.4). Finally, in the difficult scenarios GMM is out-performing other algorithms with a rank of 1.6 (Table 4.5).

4.5 Summary

In this chapter, we have proposed a new method for anomaly detection, which is based on the standard RBF network. In order to modify an RBF network for one-class classification we have proposed a cost function that maximises the output of the network, given the training set comprised of exclusively normal class samples, while keeping the size of all Gaussians as small as possible. This should learn the most compact set of Gaussians possible to cover the normal input subspace and avoid any coverage of the non-normal regions of the input space. We have also regularised the weight vector to avoid over-fitting, and force better positioning of the Gaussians in the input space. Based on the results of the experiments, it seems that the RBFDD network as a novel approach to semi-supervised anomaly detection based on the RBF network, is capable of one-class classification.

Even though RBFDD has shown the best performance across all the experiments, it is worth mentioning that this is not always the case. For instance, for the binary classification datasets, RBFDD is actually performing poorly. However, in the one-vs-all experiments RBFDD out-performs the other approaches. The one-vs-all experiments are particularly important for anomaly detection as it is likely that anomalies will arise from very different flavors of distributions.

Even though, the results show that the proposed RBFDD network has the potential to be a strong anomaly detector there is plenty of room for improvement. The current RBFDD approach makes an assumption that the input features have equal variances, and no correlation with one another. Although this assumption offers the advantage of some computational efficiency, it is quite limiting as Gaussians are forced to always have a radial shape, with no flexibility to either elongate or rotate. In the next chapter, we allow unequal variances per input feature, and also correlation between the input features, which dictates using a non-diagonal covariance matrix. This should make

the model much more flexible as it will no longer be constrained by radial kernels—kernels will be ellipsoids and capable of rotation. This property could lead to more interesting decision boundaries and better performance.

CHAPTER

FIVE

ELLIPTICAL BASIS FUNCTION DATA DESCRIPTOR NETWORKS

5.1 Introduction

In the previous chapter, RBF networks were successfully transformed into a one-class classifier and it was shown that in 61 benchmark experiments, overall, the RBFDD network achieves the best performance compared to a collection of state of the art algorithms. A potential limitation of RBFDD networks, however, is due to their radial kernels. One might argue that if these kernels were to become more flexible in shape, the resultant anomaly detector would become more effective. In this chapter, this issue is addressed by allowing unequal variances per input feature, and also correlation between the input features. This dictates using a non-diagonal covariance matrix in the kernels, which should make the model more flexible as it will no longer be constrained by radial kernels— kernels will be ellipsoids and capable of rotation. This property could lead to more interesting decision boundaries and better performance. We have named the resultant model the Elliptical Basis Function Data Descriptor (EBFDD) network. Thus the research question to be addressed in this chapter is:

- *Will replacing the radial kernels with elliptical kernels in a one-class RBF network make the model a stronger anomaly detector?*

In this Chapter the Elliptical Basis Function Data Descriptor (EBFDD) network

(H. Bazargani and Mac Namee, 2019) is proposed. A set of benchmark experiments show that the added flexibility of EBFDD networks leads to a more effective anomaly detector that out-performs the RBFDD network approach and other current state of the art methods.

This chapter proceeds as follows. Section 5.2 describes the proposed EBFDD anomaly detector. Section 5.3 explains the evaluation experiment that is designed to assess the performance of the EBFDD approach and compare it to the state of the art and Section 5.4 discusses the results of these experiments. Finally, Section 5.5 concludes the chapter.

5.2 The Elliptical Basis Function Data Descriptor (EBFDD) Network

The EBFDD network is a semi-supervised one-class classifier, and just like its predecessor, the RBFDD network, it is based on the Radial Basis Function (RBF) network (Broomhead and Lowe, 1988; Moody and Darken, 1989).

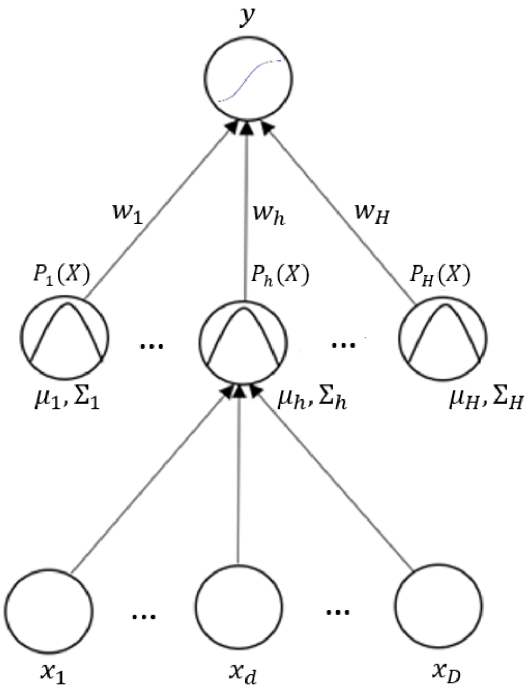


Figure 5.1: The Architecture of the EBFDD network.

Fig. 5.1 illustrates the architecture of the EBFDD network. This is a shallow network

containing one hidden layer and one output layer with a single node. If X_i is the i^{th} input vector of length D, then x_d is the value of the d^{th} dimension of X_i. There are H hidden nodes in the hidden layer, each of which uses a Gaussian activation function. The activation, $P_h(X_i)$, of the h^{th} Gaussian node is defined as:

$$P_h(X_i) = exp\left[-\frac{1}{2}\left(X_i - \mu_h\right)^T {\Sigma_h}^{-1}\left(X_i - \mu_h\right)\right] \tag{5.1}$$

where the parameters μ_h and Σ_h are the mean vector (which is D-dimensional) and the covariance matrix (which is a $D \times D$ matrix) of the h^{th} Gaussian kernel.

Similar to RBFDD networks, in EBFDD networks the outputs of the nodes in the hidden layer are connected to a single output node via a weight vector W, where w_h is the weight parameter connecting the h^{th} hidden node to the output node. In the output node the modified hyperbolic tangent, $tanh$, activation function proposed in (Orr and Müller, 1998) is again used as it avoids saturation:

$$y_i = 1.7159 \times tanh\left(\frac{2z(X_i)}{3}\right) \tag{5.2}$$

where $z(X_i)$ is the weighted sum of the outputs of the hidden layer, when X_i is the input vector:

$$z(X_i) = \sum_{h=1}^{H} w_h \times P_h\left(X_i\right) \tag{5.3}$$

where w_h is the weight parameter connecting the h^{th} Gaussian kernel to the output node. Similar to RBFDD, the training of EBFDD begins with a pre-training phase based on k-means clustering, that provides an initial set of positions for the H Gaussian kernels at the hidden layer. The resulting cluster centres are used to initialise the Gaussian kernel centres, μ_h. The covariance matrices, Σ_h, for each of the H hidden nodes, are initialised to be radial and diagonal— equal variance in all directions is assumed with off-diagonal elements being zero. The number of hidden nodes in the network, H, (which is also the number of clusters found by k-means) is a hyper-parameter set before training starts.

The parameter values (μ_h, Σ_h, and w_h) for each node in the network are then optimised using mini-batch gradient descent (Alpaydin, 2004) (mini-batch sizes are set to 32) and the back-propagation of error algorithm (Alpaydin, 2004). The cost function to

be minimised in this process is the following, which is minimised for mini-batches of size N:

$$E = \sum_{i=1}^{N} \left(\frac{1}{2} \left[(1 - y_i)^2 + \beta R_\Sigma + \lambda R_W \right] \right) \tag{5.4}$$

where y is the output of the network; R_Σ and R_W are regularisation terms defined below in Eq.(5.5) and Eq.(5.6); and β and λ are hyper-parameters that control the influence of the regularisation terms. This is a more general form of the cost function defined in Eq.(4.1).

This cost function is a weighted sum of three main terms that ensure the network learns a compact set of Gaussians that cover just the normal training data, however, unlike RBFDD these elliptical kernels should be able to cover more sophisticated distributions in the normal data. The interpretation of this cost function is quite similar to that of the RBFDD network. The first term in this summation, $(1 - y_i)^2$, encourages the network training process to learn a model that outputs a value as close as possible to 1 for instances belonging to the normal class. The second term in the cost function, R_Σ, regularises the variances of the Gaussian kernels in the hidden layer of the network. This term introduces the optimisation criterion of having the most compact set of Gaussians possible to represent the normal data. As the size of a Gaussian ellipsoid is dictated by the diagonal elements (i.e., the variances) of its covariance matrix, off-diagonal elements are excluded from the regularisation. R_Σ is defined as:

$$R_\Sigma = \sum_{h=1}^{H} \sum_{d=1}^{D} \left(\Sigma_h \left[d, d \right] \right)^2 \tag{5.5}$$

which is the squared L2-norm (Goodfellow et al., 2016) of the variances (i.e., the diagonal elements in each of the H Gaussian covariance matrices). Here, D denotes the dimensionality of the input space and $\Sigma_h \left[d, d \right]$ refers to the d^{th} diagonal element of the convariance matrix for the h^{th} hidden node.

The third term in the cost function, R_W, is the squared L2-norm of the weight vector, W, connecting the hidden layer nodes to the output node. As mentioned in the previous chapter, this discourages the weights from becoming so large that they would actually ignore the outputs from the hidden nodes. Just like in the case of the RBFDD network, regularising the weights also makes the EBFDD network robust to outliers

in the training set (Goodfellow et al., 2016). R_W is defined as:

$$R_W = \sum_{h=1}^{H} w_h^2 \tag{5.6}$$

where w_h is the weight connecting the h^{th} hidden node to the output node.

Similar to its predecessor, we argue that for an EBFDD network, a gradient descent training process based on the minimisation of the cost function in Eq.(5.4) will find a compact set of Gaussians, whose collective output is still high for the normal region of the input space and low anywhere else (i.e., where we believe the anomalies would appear).

Based on the application of the back-propagation of error algorithm using gradient descent, the following update rules for learning the parameters of the EBFDD network have been derived in Eq.(5.7), Eq.(5.8) and Eq.(5.9)[1]. The general structure of these update rules is $a = a - \eta \times \frac{\partial E}{\partial a}$, where a is the learnable parameter, and E is the cost function defined in Eq.(5.4). Each of the weights, w_h, connecting a hidden unit to the output unit is updated using:

$$\begin{aligned}
w_h &\leftarrow w_h - \eta \times \frac{\partial E}{\partial w_h} \\
&\leftarrow w_h - \eta \left[\frac{\partial \frac{1}{2}(1-y_i)^2}{\partial w_h} + \frac{\partial \frac{1}{2}\lambda R_W}{\partial w_h} \right] \\
&\leftarrow w_h - \eta \left[\frac{\partial \frac{1}{2}(1-y_i)^2}{\partial y_i} \times \frac{\partial y_i}{\partial z_h(X_i)} \times \frac{\partial z_h(X_i)}{\partial w_h} + \frac{\partial \frac{1}{2}\lambda R_W}{\partial w_h} \right] \\
&\leftarrow w_h - \eta \left[(y_i - 1) \times 1.1439 \left(1 - tanh\left(\frac{2z_h(X_i)}{3} \right)^2 \right) \times P_h(X_i) + \lambda w_h \right]
\end{aligned} \tag{5.7}$$

where η is the learning rate, and $z_h(X_i)$ is defined as in Eq.(5.3) (although we omit the summation because only the h^{th} hidden unit is relevant). All other terms are defined as before. Each of the kernel centres, μ_h, is updated using:

[1]The update rules are described for a single training instance for ease of reading, but are easily expandable to a mini-batch scenario by summing across the instances in the mini-batch.

$$
\begin{aligned}
\mu_h &\leftarrow \mu_h - \eta \times \frac{\partial E}{\partial \mu_h} \\
&\leftarrow \mu_h - \eta \left[\frac{\partial \frac{1}{2}(1-y_i)^2}{\partial \mu_h} \right] \\
&\leftarrow \mu_h - \eta \left[\frac{\partial \frac{1}{2}(1-y_i)^2}{\partial y_i} \times \frac{\partial y_i}{\partial z_h(X_i)} \times \frac{\partial z_h(X_i)}{\partial P_h(X_i)} \times \frac{\partial P_h(X_i)}{\partial \mu_h} \right] \\
&\leftarrow \mu_h - \eta \left[(y_i - 1) \times 1.1439 \left(1 - tanh \left(\frac{2 z_h(X_i)}{3} \right)^2 \right) \times w_h \times P_h(X_i) \Sigma_h^{-1} (X_i - \mu_h) \right]
\end{aligned}
\tag{5.8}
$$

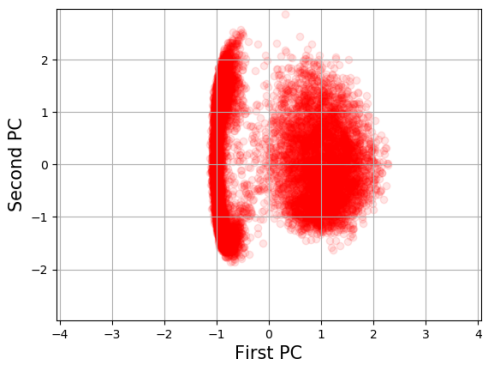
And for the covariance matrix of the h^{th} Gaussian, Σ_h, the update rule is:

$$
\begin{aligned}
\Sigma_h &\leftarrow \Sigma_h - \eta \times \frac{\partial E}{\partial \Sigma_h} \\
&\leftarrow \Sigma_h - \eta \times \left[\frac{\partial \frac{1}{2}(1-y_i)^2}{\partial \Sigma_h} + \frac{\partial \frac{1}{2} \beta R_\Sigma}{\partial \Sigma_h} \right] \\
&\leftarrow \Sigma_h - \eta \times \left[\frac{\partial \frac{1}{2}(1-y_i)^2}{\partial y_i} \times \frac{\partial y_i}{\partial z_h(X_i)} \times \frac{\partial z_h(X_i)}{\partial P_h(X_i)} \times \frac{\partial P_h(X_i)}{\partial \Sigma_h} + \frac{\partial \frac{1}{2} \beta R_\Sigma}{\partial \Sigma_h} \right] \\
&\leftarrow \Sigma_h - \eta \times \left[(y_i - 1) \times 1.1439 \left(1 - tanh \left(\frac{2 z_h(X_i)}{3} \right)^2 \right) \times w_h \right. \\
&\qquad \left. \times P_h(X_i)(-\Sigma_h^{-T}(X_i - \mu_h)(X - \mu_h)^T \Sigma_h^{-T}) + \beta Diag(\Sigma_h) \right]
\end{aligned}
\tag{5.9}
$$

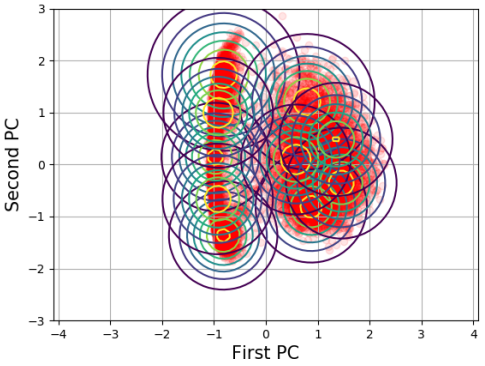
where the function $Diag()$ diagonalises the covariance matrix Σ_h by turning all non-diagonal elements to 0 and returns a diagonal $D \times D$ matrix.

When Eq.(5.9) is applied, the resulting covariance matrix may not be invertible. This is a problem, as the inverse of the covariance matrices are used in both forward and backward propagation in the network. In such cases, we replace the covariance matrix generated with its closest positive semi-definite matrix found using the method proposed by Higham (1988).

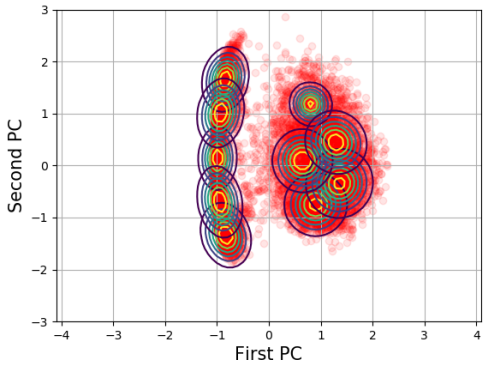
To illustrate the behaviour of an EBFDD network we present a simple example based on the MNIST dataset of handwritten digits (LeCun and Cortes, 2010), where images of digits 0 and 1 are considered as the normal class. We have used 10 kernels in this illustration. So that the behaviour of the network can be visualised, the dimensionality



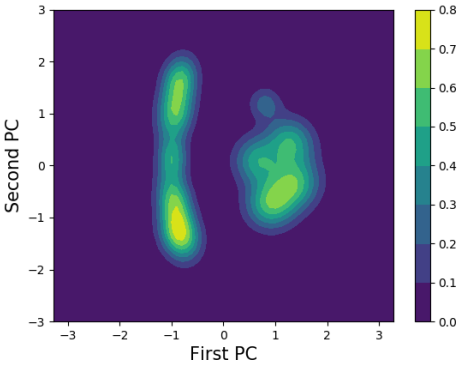
(a) Training Data



(b) Converged k-means



(c) Converged EBFDD



(d) Decision Boundary

Figure 5.2: Considering images of both digits 0 and 1 from the MNIST dataset as class *normal* instances for training and using the first two principal components, we have the visualisations of the transformed training data after applying PCA in red circles, the Gaussians after k-means pre-training, trained EBFDD network, and the decision surface of the trained EBFDD network.

of the input data is reduced to 2 using Principal Component Analysis (PCA) (Alpaydin, 2004). Fig. 5.2a shows the training dataset of normal instances (images of digits 0 and 1). Fig. 5.2b shows the output of the k-means algorithm after the pre-training phase, which provides the EBFDD network with an initial set of well positioned Gaussian kernels. Fig. 5.2c shows the compact set of kernels covering the normal region after the EBFDD network is trained. Finally, Fig. 5.2d shows the decision surface that has been learned, where brighter colors correspond to higher outputs by the EBFDD network. It is interesting to see how the flexibility of the kernels can make them elongate and rotate in such a way that they tightly cover the training data. In addition, if we compare this with a similar visualisation in Fig. 4.2 of Chapter 4 where 15 Gaussians were used with RBFDD networks, here with 10 Gaussians a more clear and descriptive decision boundary for the normal data is learned with the EBFDD network.

Table 5.1: Hyper-parameter ranges for each algorithm

Algorithms	Hyper-parameters	Investigated Hyper-Parameters
EBFDD	number of kernels (H)	[1, 5, 10, ..., D]
	η	[0.01, 0.001, 0.0001]
	β	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
	λ	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
RBFDD	number of kernels (H)	[1, 5, 10, ..., D]
	η	[0.01, 0.001, 0.0001]
	β	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
	λ	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
OCSVM	ν	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
	γ	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
AEN	number of hidden units (H)	[1, 5, 10, ..., D]
	η	[0.01, 0.001, 0.0001]
	hidden layer activation functions	[sigmoid, relu]
	output layer activation functions	[sigmoid, relu, linear]
	cost functions	[mean squared error, cross entropy]
GMM	number of kernels (H)	[1, 5, 10, ..., D]
Isolation Forest	number of estimators	[100, 200, 500, 800, 1000]

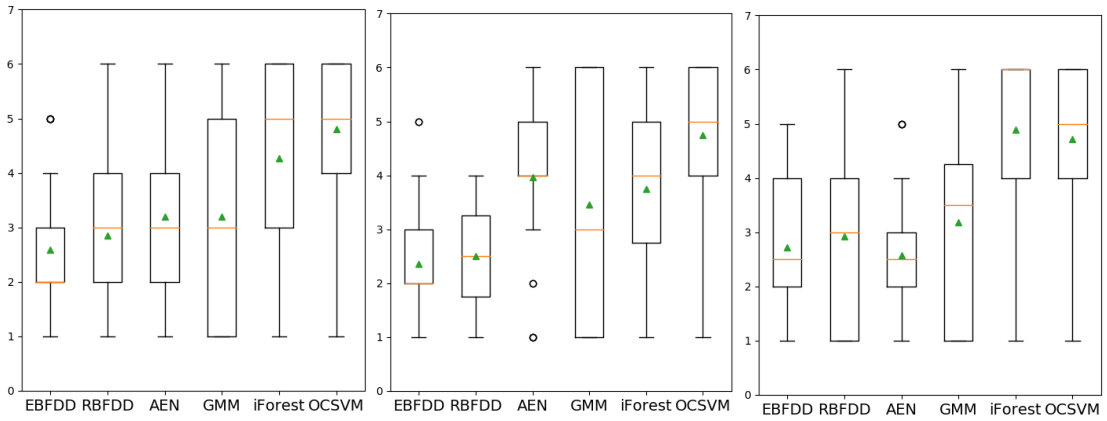
5.3 Evaluation Experiment

To evaluate the performance of the EBFDD network and compare it with state of the art anomaly detection approaches, including its predecessor the RBFDD network, we add the EBFDD network to the evaluation experiment described in Section 4.3 and follow the same experimental design.

The tunable hyper-parameters used to generate these results, for each algorithm across all the datasets and scenarios, are presented in Table 5.1. Note that these hyper-parameters and their searched range are identical to the previous chapter, and the only addition is the hyper-parameters for the EBFDD network. Similar to the previous chapter, the area under the ROC curve has been chosen as the evaluation metric in these experiments.

5.4 Results & Discussion

The performance of each algorithm on each anomaly detection scenario for each dataset was compared and ranked. Tables 5.2 to 5.5 show results on the binary, difficult scenarios, one-vs-all, and all-vs-one scenarios. The distribution of the ranks



(a) All Scenarios (b) One-vs-All (c) All-vs-One

Figure 5.3: The average rank of the algorithms across the experiments. On each box plot the median ranks are shown via the horizontal orange lines and the mean ranks are shown using green triangles.

for each algorithm for all anomaly detection scenarios are shown in Fig. 5.3a, and the average rank scores are summarised in Table 5.6. The average ranks in Table 5.6 show that the EBFDD network has the lowest average rank over all experiments and, on average, out performs the other approaches - including RBFDD. We believe that the EBFDD out-performs RBFDD because of the more flexible nature of the decision boundaries that it is able to learn because of the use of elliptical kernels. It is also clear from Fig. 5.3a that the ranks of EBFDD have low variance indicating its consistent strong performance. It is worth noting that the EBFDD with a rank of 3.3 is not performing well for the binary-class datasets, even though it has improved the performance of the RBFDD network with a rank of 5.3. Similar to RBFDD, it seems that EBFDD too shows a strong performance in the one-vs-all scenarios, and comes on top with a rank of 2.4. In all-vs-one scenarios, the AEN with a rank of 2.6 attains the best rank, and we can see that EBFDD with a rank of 2.7 is coming second while out-performing RBFDD, which has a rank of 2.9. Finally, in the difficult scenario, GMM with a rank of 2.0 proves to be the strongest, while EBFDD has come in second with a rank of 2.4.

Table 5.2: Experiment Results and Scenarios for Binary-Class Datasets

	(Normal-Anomalous) Class Labels	EBFDD	RBFDD	OCSVM	AEN	GMM	iForest
Gamma	('g'-'h')	0.81060 (3)	0.77450 (5)	0.74096 (6)	0.81670 (2)	0.86147 (1)	0.77516 (4)
Spambase	(0-1)	0.67211 (5)	0.63396 (6)	0.70562 (4)	0.82311 (3)	0.85171 (1)	0.84728 (2)
Skin	(2-1)	0.90384 (2)	0.84751 (5)	0.70549 (6)	0.92357 (1)	0.88891 (4)	0.89098 (3)
Average Rank		3.3 (4)	5.3 (5)	5.3 (5)	**2.0 (1)**	**2.0 (1)**	3.0 (3)

Table 5.3: Experiment Results for Multi-Class Datasets: Difficult Scenarios

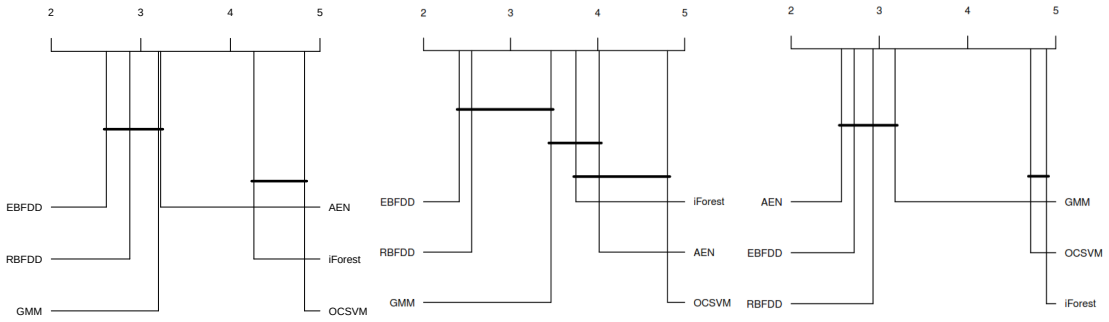
	(Normal-Anomalous) Class Labels	EBFDD	RBFDD	OCSVM	AEN	GMM	iForest
Fault	(Pastry,Z_Scratch,K_Scratch,Stains,Dirtiness,Bumps) -(other_Faults)	0.79641 (1)	0.73409 (4)	0.67300 (5)	0.74380 (3)	0.78232 (2)	0.66463 (6)
Image Segmentation	(BRICK,FOLIAGE,CEMENT,GRASS) -(SKY,WINDOW,PATH)	0.82873 (3)	0.81797 (4)	0.73711 (6)	0.88260 (1)	0.86549 (2)	0.77386 (5)
LandSat	(red soil,cotton crop,grey soil,very damp grey soil) -(damp grey soil,soil with vegetation stubble)	0.70100 (3)	0.71909 (2)	0.53770 (6)	0.60676 (5)	0.73845 (1)	0.63110 (4)
Page	(text) -(horiz. line,graphic,vert. line,picture)	0.93995 (3)	0.93654 (4)	0.90640 (6)	0.94363 (2)	0.95736 (1)	0.92717 (5)
Wave	(1,2) -(0)	0.81664 (2)	0.83539 (1)	0.69253 (6)	0.78260 (3)	0.76900 (4)	0.72886 (5)
Average Rank		2.4 (2)	3.0 (4)	5.8 (6)	2.8 (3)	**2.0 (1)**	5.0 (5)

Table 5.4: Experiment Results for Multi-Class Datasets: One-vs-All Scenarios

	Normal Class Label	EBFDD	RBFDD	OCSVM	AEN	GMM	iForest
Fault	Pastry	0.87555 (2)	0.87388 (3)	0.85343 (6)	0.86261 (5)	0.88637 (1)	0.87033 (4)
	Z_Scratch	0.97772 (1)	0.97667 (2)	0.97224 (4)	0.96593 (5)	0.97570 (3)	0.92684 (6)
	K_Scatch	0.98856 (2)	0.98721 (3)	0.96846 (6)	0.98017 (4)	0.98888 (1)	0.98001 (5)
	Stains	0.99561 (2)	0.99480 (3)	0.99184 (5)	0.99325 (4)	0.98924 (6)	0.99766 (1)
	Dirtiness	0.99376 (1)	0.98214 (2)	0.93395 (5)	0.93515 (4)	0.97002 (3)	0.91264 (6)
	Bumps	0.84157 (1)	0.82843 (4)	0.82729 (5)	0.82484 (6)	0.83205 (2)	0.83173 (3)
	Other_Faults	0.62293 (2)	0.62015 (3)	0.59953 (5)	0.59474 (6)	0.65096 (1)	0.61955 (4)
Image Segmentation	BRICK	0.99535 (3)	0.99394 (4)	0.97861 (6)	0.99328 (5)	0.99920 (1)	0.99815 (2)
	SKY	1.00000 (1)	1.00000 (1)	1.00000 (1)	1.00000 (1)	0.99881 (5)	0.99810 (6)
	FOLIAGE	0.92810 (3)	0.93342 (2)	0.87113 (6)	0.91278 (4)	0.95734 (1)	0.90678 (5)
	CEMENT	0.93665 (3)	0.94126 (1)	0.90427 (5)	0.92766 (4)	0.94099 (2)	0.90318 (6)
	WINDOW	0.91953 (3)	0.91362 (4)	0.86126 (6)	0.90077 (5)	0.95195 (1)	0.93139 (2)
	PATH	0.99605 (3)	0.99823 (2)	0.98943 (5)	0.99440 (4)	0.99974 (1)	0.98015 (6)
	GRASS	0.99993 (2)	1.00000 (1)	0.99887 (5)	0.99946 (4)	0.99848 (6)	0.99986 (3)
LandSat	red soil	0.99355 (2)	0.98994 (4)	0.98820 (5)	0.99423 (1)	0.98729 (6)	0.99136 (3)
	cotton crop	0.99577 (1)	0.99469 (2)	0.98829 (4)	0.98255 (5)	0.96337 (6)	0.98876 (3)
	grey soil	0.97577 (2)	0.97932 (1)	0.97379 (4)	0.97434 (3)	0.95708 (6)	0.97054 (5)
	damp grey soil	0.92691 (2)	0.92693 (1)	0.92038 (5)	0.92378 (4)	0.87460 (6)	0.92636 (3)
	soil with vegetation stubble	0.90505 (2)	0.91687 (1)	0.87032 (6)	0.88969 (4)	0.87151 (5)	0.90125 (3)
	very damp grey soil	0.96158 (2)	0.96146 (3)	0.95407 (6)	0.95853 (4)	0.95448 (5)	0.96252 (1)
Page	text	0.93995 (3)	0.93654 (4)	0.90640 (6)	0.94363 (2)	0.95736 (1)	0.92717 (5)
	horiz. line	0.71468 (5)	0.84573 (3)	0.61670 (6)	0.74304 (4)	0.93233 (1)	0.87221 (2)
	graphic	0.79569 (5)	0.81629 (4)	0.92336 (3)	0.93125 (2)	0.94769 (1)	0.57743 (6)
	vert. line	0.96094 (3)	0.97299 (1)	0.91514 (6)	0.94404 (5)	0.96387 (2)	0.95872 (4)
	picture	0.81344 (4)	0.83315 (2)	0.81884 (3)	0.87817 (1)	0.79179 (6)	0.81032 (5)
Wave	0	0.87200 (1)	0.86507 (2)	0.86207 (3)	0.84739 (5)	0.84634 (6)	0.86151 (4)
	1	0.91652 (2)	0.91489 (3)	0.91239 (4)	0.90627 (5)	0.89499 (6)	0.92074 (1)
	2	0.91854 (3)	0.91360 (4)	0.91934 (2)	0.90830 (5)	0.90010 (6)	0.92011 (1)
Average Rank		**2.4 (1)**	2.5 (2)	4.7 (6)	4.0 (5)	3.5 (3)	3.8 (4)

Table 5.5: Experiment Results for Multi-Class Datasets: All-vs-One Scenarios

	Anomalous Class Label	EBFDD	RBFDD	OCSVM	AEN	GMM	iForest
Fault	Pastry	0.74429 (2)	0.74711 (1)	0.66176 (5)	0.72821 (3)	0.66243 (4)	0.58524 (6)
	Z_Scratch	0.77937 (1)	0.74209 (3)	0.71712 (5)	0.77584 (2)	0.73010 (4)	0.55213 (6)
	K_Scatch	0.94033 (4)	0.94678 (3)	0.91684 (6)	0.92212 (5)	0.94783 (2)	0.94893 (1)
	Stains	0.86140 (2)	0.86556 (1)	0.80848 (5)	0.81068 (4)	0.84424 (3)	0.74532 (6)
	Dirtiness	0.72816 (3)	0.73767 (2)	0.70261 (4)	0.78197 (1)	0.65360 (5)	0.60343 (6)
	Bumps	0.70685 (2)	0.73194 (1)	0.62443 (5)	0.65910 (3)	0.64203 (4)	0.51147 (6)
	Other_Faults	0.79641 (1)	0.73409 (4)	0.67300 (5)	0.74380 (3)	0.78232 (2)	0.66463 (6)
Image Segmentation	BRICK	0.65811 (3)	0.54813 (4)	0.46279 (6)	0.93805 (2)	0.95622 (1)	0.48590 (5)
	SKY	0.99974 (4)	0.99984 (3)	1.00000 (1)	0.99994 (2)	0.99626 (5)	0.98901 (6)
	FOLIAGE	0.84694 (2)	0.78690 (5)	0.82876 (4)	0.90069 (1)	0.82879 (3)	0.72881 (6)
	CEMENT	0.73917 (3)	0.53020 (5)	0.49775 (6)	0.83910 (2)	0.95943 (1)	0.58751 (4)
	WINDOW	0.56663 (4)	0.54775 (5)	0.62737 (3)	0.72612 (2)	0.83608 (1)	0.46808 (6)
	PATH	0.89445 (3)	0.64221 (6)	0.72225 (5)	0.97240 (1)	0.91111 (2)	0.74520 (4)
	GRASS	0.99435 (4)	0.99210 (5)	0.99801 (3)	0.99910 (2)	0.99991 (1)	0.92048 (6)
LandSat	red soil	0.93566 (3)	0.91750 (4)	0.84476 (5)	0.98846 (1)	0.96100 (2)	0.78044 (6)
	cotton crop	0.99854 (2)	0.99693 (3)	0.99389 (5)	0.99901 (1)	0.99622 (4)	0.99307 (6)
	grey soil	0.90203 (4)	0.94770 (3)	0.95172 (1)	0.95034 (2)	0.61131 (6)	0.86509 (5)
	damp grey soil	0.65422 (1)	0.63347 (2)	0.46886 (5)	0.57443 (3)	0.52522 (4)	0.39395 (6)
	soil with vegetation stubble	0.87574 (2)	0.87001 (3)	0.74329 (6)	0.79659 (5)	0.90894 (1)	0.79804 (4)
	very damp grey soil	0.71354 (4)	0.73539 (3)	0.65526 (5)	0.83786 (1)	0.58132 (6)	0.73832 (2)
Page	text	0.54191 (5)	0.72955 (2)	0.46192 (6)	0.66032 (3)	0.79199 (1)	0.60372 (4)
	horiz. line	0.92208 (3)	0.92081 (4)	0.89531 (6)	0.95125 (2)	0.96770 (1)	0.91872 (5)
	graphic	0.86888 (4)	0.82382 (5)	0.73967 (6)	0.93934 (3)	0.97463 (1)	0.94970 (2)
	vert. line	0.94563 (2)	0.95057 (1)	0.92444 (5)	0.94351 (3)	0.94242 (4)	0.88595 (6)
	picture	0.92755 (2)	0.92968 (1)	0.91352 (4)	0.91725 (3)	0.87087 (5)	0.83920 (6)
Wave	0	0.81664 (2)	0.83539 (1)	0.69253 (6)	0.78260 (3)	0.76900 (4)	0.72886 (5)
	1	0.86285 (2)	0.90888 (1)	0.80230 (5)	0.80481 (4)	0.74791 (6)	0.81359 (3)
	2	0.87352 (2)	0.89647 (1)	0.83358 (4)	0.82682 (5)	0.76207 (6)	0.83522 (3)
Average Rank		2.7 (2)	2.9 (3)	4.7 (5)	**2.6 (1)**	3.2 (4)	5.0 (6)



(a) All Scenarios (b) One-vs-All (c) All-vs-One

Figure 5.4: Critical difference plots from Friedman's test with Finner p-value correction with significance level $\alpha = 0.05$

Table 5.6: Average Rank Across All Experiments

	EBFDD	RBFDD	AEN	GMM	iForest	OCSVM
Average Rank	**2.6 (1)**	2.9 (2)	3.2 (3)	3.2 (3)	4.3 (5)	4.8 (6)

Fig. 5.3 also shows the distribution of ranks of the different approaches on the one-vs-all and all-vs-one scenarios (distributions for the other scenarios are not included as there are too few results to generate meaningful distributions). While the overall pattern is largely the same for these subsets as for the overall results, it is worth noting that there is a marked difference in the performance of the AEN models on the one-vs-all scenarios—performance is relatively poor—and on the all-vs-one scenarios—performance is quite good.

A Friedman test with Finner p-value correction was performed to further investigate the pairwise differences in the performance of the algorithms. Fig. 5.4 summarises the results of these tests (tests were performed for all scenarios and independently for the one-vs-all and all-vs-one scenarios). The complete results of the test are shown in the summary Tables 5.7, 5.8 and 5.9. The lower triangle shows the corrected p-values from the Friedman test, showing whether the null hypothesis can be rejected with a desired significance level, denoted with α. Also, to compare the performances directly, a simple win/lose/tie count is given in the upper triangle of the tables.

For example, let us consider EBFDD and OCSVM in Table 5.7. Looking at the lower triangle of the table, we can see that the p-value is very small, effectively 0, which suggests that we can reject the null hypothesis at every significance level. This means that overall, EBFDD performed significantly better than OCSVM. Looking at the upper triangle, we can see the win/lose/tie information as 8/52/1. This means that the AUC score for the OCSVM has been higher than the EBFDD network 8 times (OCSVM

won), lower than the EBFDD network 52 times (OCSVM lost), and equal to the EBFDD network 1 time (a draw), across all the experiments, summing up to 61 experiments in total[2].

Table 5.7: All Scenarios: Win/Lose/Tie for Friedman Aligned Test with Significance Levels $\alpha : * = 0.1, ** = 0.05, * * * = 0.01$

	RBFDD	OCSVM	EBFDD	AEN	GMM	iForest
RBFDD		51/9/1	26/34/1	36/24/1	32/29/0	44/17/0
OCSVM	0.0000 ***		8/52/1	10/50/1	18/43/0	25/36/0
EBFDD	0.4614	0.0000 ***		36/24/1	34/27/0	49/12/0
AEN	0.3706	0.0000 ***	0.1193		32/29/0	38/23/0
GMM	0.3866	0.0000 ***	0.1259	0.9421		43/18/0
iForest	0.0001 ***	0.1273	0.0000 ***	0.0040 ***	0.0035 ***	

Table 5.8: One-vs-All scenarios: Win/Lose/Tie for Friedman Aligned Test with Significance Levels $\alpha : * = 0.1, ** = 0.05, * * * = 0.01$

	RBFDD	OCSVM	EBFDD	AEN	GMM	iForest
RBFDD		25/2/1	11/16/1	23/4/1	17/11/0	19/9/0
OCSVM	0.0001 ***		3/24/1	7/20/1	11/17/0	9/19/0
EBFDD	0.7751	0.0000 ***		22/5/1	16/12/0	21/7/0
AEN	0.0127 **	0.1549	0.0065 ***		13/15/0	12/16/0
GMM	0.1010	0.0220 **	0.0648 *	0.3232		16/12/0
iForest	0.0355 **	0.0648 *	0.0220 **	0.6200	0.6200	

Table 5.9: All-vs-One scenarios: Win/Lose/Tie for Friedman Aligned Test with Significance Levels $\alpha : * = 0.1, ** = 0.05, * * * = 0.01$

	RBFDD	OCSVM	EBFDD	AEN	GMM	iForest
RBFDD		22/6/0	14/14/0	12/16/0	15/13/0	23/5/0
OCSVM	0.0009 ***		4/24/0	3/25/0	7/21/0	16/12/0
EBFDD	0.7200	0.0002 ***		13/15/0	17/11/0	24/4/0
AEN	0.5847	0.0001 ***	0.7751		17/11/0	23/5/0
GMM	0.6988	0.0040 ***	0.4797	0.3456		23/5/0
iForest	0.0003 ***	0.7453	0.0001 ***	0.0001 ***	0.0013 ***	

In all cases the performance of a group of the best performing algorithms cannot be separated in a statistically significant way. This group includes EBFDD, RBFDD, GMM, and (except for the one-vs-all scenario subset) AEN. The poorer performances of the OCSVM and iForest approaches are statistically significantly different to the performance of the other approaches.

[2]Three out of five difficult scenarios in Table 5.3 are already included in either one-vs-all or all-vs-one scenarios. Hence, the number of unique experiments reduces from 64 to 61

5.4.1 The Number of Trainable Parameters

It is worth noting that the cost of the good performance of the EBFDD algorithm is the very large number of parameters that must be learned in this model. For an EBFDD network, as the full covariance matrices are learnable, the number of trainable parameters is quite high:

$$(H \times D) + \left(H \times \frac{D(D+1)}{2}\right) + H \tag{5.10}$$

where H is the number of hidden nodes, and D is the dimensionality of the input. In comparison, given its use of radial kernels, the number of trainable parameters for an RBFDD network is:

$$(H \times D) + (H \times D) + H \tag{5.11}$$

This means that training EBFDD networks can take longer than training equivalently sized RBFDD networks.

There is a minor difference in the number of trainable parameters in a GMM compared to an EBFDD network. As the weights used to combine distributions in a GMM sum to 1, for H Gaussians only $(H-1)$ weights need to be learned as the last weight can be computed by subtracting the sum of those weights from 1. As a result, the number of trainable parameters in a GMM is equal to:

$$(H \times D) + \left(H \times \frac{D(D+1)}{2}\right) + (H-1) \tag{5.12}$$

In addition the number of trainable parameters for the AEN is:

$$(H \times D) + (H \times D) \tag{5.13}$$

It is not possible to talk about the number of trainable parameters for the iForest and OCSVM models in a meaningful way and so these are not compared. Moreover, as the implementations of algorithms used in these experiments come from different packages with differing levels of optimisation we do not believe that detailed comparisons of training times are appropriate.

As observed in Table 5.6, the empirical results show that the added complexity of the EBFDD network has made it a stronger anomaly detector compared to the simpler RBFDD network. In addition, even though the EBFDD network and GMM have al-

most the same number of trainable parameters, it seems that the optimisation of the proposed cost function in the EBFDD network, which utilises gradient descent, allows it to find better solutions than the expectation maximisation approach used in a GMM.

5.5 Summary

This chapter presented the Elliptical Basis Function Data Descriptor (EBFDD), a modification of RBFDD that introduces elliptical kernels. This modification involved the definition of a new cost function and adapting other aspects of the network. EBFDD utilises elliptical kernels that can elongate and rotate to allow it to learn sophisticated decision surfaces. An evaluation experiment conducted using a large collection of datasets compared the performance of the EBFDD network with state of the art anomaly detection algorithms, as well as its predecessor the RBFDD network. Based on the average ranks, EBFDD has always out-performed RBFDD. We believe that this is because of the more flexible nature of the decision boundaries that EBFDD is able to learn, due to the use of elliptical kernels, compared to RBFDD. Although statistical significance is not shown in all cases, the empirical results show that the EBFDD network has a better overall performance across all of the experiments with a rank of 2.6. Similarly to RBFDD, EBFDD performs especially well in the one-vs-all scenarios. The one-vs-all scenarios are particularly important for anomaly detection as it is likely that anomalies will arise from very different data distributions.

As a result, to answer the research question in this chapter, it seems that adding elliptical kernels to the RBFDD network has turned it into a stronger anomaly detector. This improved performance, however, is at a cost of much increased training time as there are so many more model parameters to fit. This is particularly challenging for high-dimensional datasets.

So far, only datasets with well-behaved sets of features have been used in the experiments. However, what will happen if we were to apply anomaly detection on high-dimensional raw data, such as, images or time series sensor data? Inevitably, one might think of making the RBFDD/EBFDD networks deeper to extract meaningful features from the raw data. Of course the concerns are 2 fold: 1) How should one approach deepening these networks? 2) Will the proposed cost functions be able to produce a strong enough signal to train a deep network? 3) Finally, will adding deep

layers to the RBFDD/EBFDD networks bring us a worthwhile advantage in terms of the performance compared to the shallow versions of these networks? In the next chapter, these concerns are addressed and investigated in detail.

CHAPTER

SIX

DEEP RADIAL BASIS FUNCTION DATA DESCRIPTORS

6.1 Introduction

RBFDD networks use radial kernels, and thus, might lack the necessary flexibility to learn certain distributions. To overcome this limitation of RBFDD, we previously proposed the Elliptical Basis Function Data Descriptor (EBFDD) network (H. Bazargani and Mac Namee, 2019) in Chapter 5, where we make the anomaly detector more flexible by replacing the radial kernels with elliptical kernels. EBFDD was shown to perform better than RBFDD, however, at significantly increased computational cost (EBFDD requires a covariance matrix inversion which is a computationally very expensive operation). We believe, however, that the same flexibility can be achieved by adding computational layers before the RBFDD layer to transform the representation into a space where RBFDD can be applied effectively and, due to the lower computation time of RBFDD compared to EBFDD, also efficiently. Thus, we avoid the computational complexity of the EBFDD networks by adding more layers while retaining the capacity of the deep model to learn complex distributions in the normal data. Moreover, although the RBFDD network is an effective anomaly detector when used with well-behaved sets of input features, it does not perform well on high-dimensional raw data representations (e.g., pixel values in images or raw sensor data). These are the main motivations for deepening the structure of the RBFDD network so that we can apply it to anomaly detection problems with raw high-dimensional input data. In this chapter, alternatives for deepening the structures of the RBFDD models will be inves-

tigated. We identify three ways to adapt RBFDD networks to work with raw input data:

1. Based on transfer learning using the latent representation from a generic pre-trained network as input to the RBFDD network.
2. Extending the first approach to include fine-tuning the pre-trained network as part of training the RBFDD.
3. Adding multiple computational layers before the RBFDD network that are fully trained as part of training the network in an *end-to-end* fashion.

In this chapter we explore the effectiveness of these three approaches, and show that the final approach that trains the entire network from random initialisation—referred to as Deep RBFDD (D-RBFDD)—out-performs the other approaches. We propose a deep anomaly detection model that can be trained in a fully end-to-end fashion, and is suitable for use with raw, high-dimensional data sources such as images and time series data from sensors. The proposed model, the *Deep Radial Basis Function Data Descriptor* (D-RBFDD) network, is based on the Radial Basis Function Data Descriptor (RBFDD) network. We use two well-known labelled image classification datasets, MNIST (LeCun and Cortes, 2010) and Fashion MNIST (Xiao et al., 2017), to explore the effectiveness of the three approaches to deepening RBFDD networks. We also use an ECG dataset, the MIT-BIH Arrhythmia Database which is a real-world highly-imbalanced anomaly detection time series dataset, to compare the performance of the D-RBFDD network with state of the art algorithms. The research question to be answered in this chapter is:

- *Is it possible to deepen the one-class RBF network architecture to make an effective anomaly detector applicable to high-dimensional raw data inputs, and what is the best way to do this deepening?*

In this chapter the Deep RBFDD (D-RBFDD) network (H. Bazargani et al., under review) is proposed as an effective approach to deepening RBFDD networks that allows

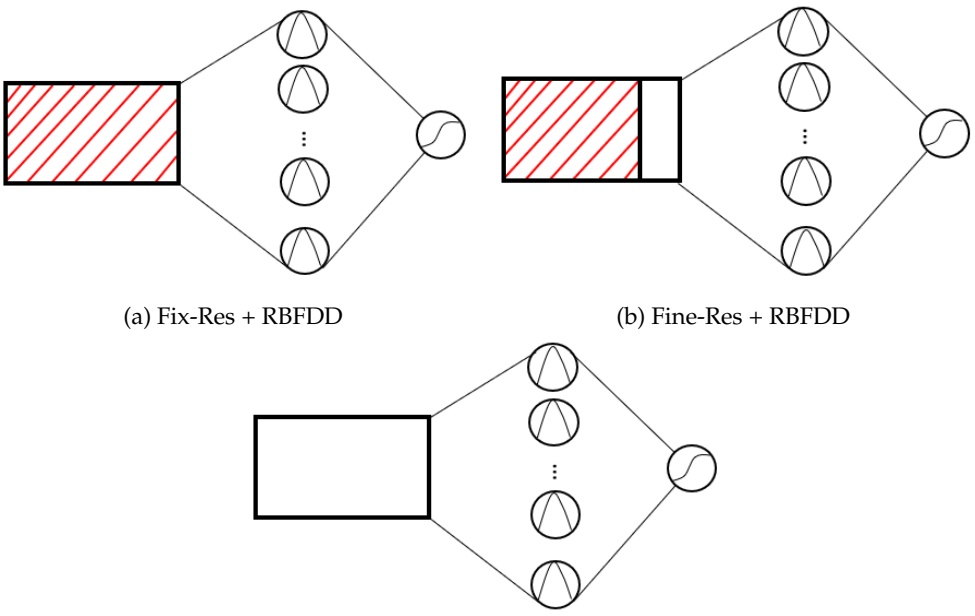
them to work with high-dimensional, raw input data. In a set of benchmark experiments, it is shown that when dealing with raw high-dimensional data, on average, the D-RBFDD network out-performs its predecessor the RBFDD network, while either out-performing the state of the art or demonstrating competitive performance with it.

This chapter proceeds as follows. Section 6.2 describes the proposed D-RBFDD anomaly detector. Section 6.3 explains the evaluation experiment that is designed to assess the performance of the D-RBFDD approach and compare it to the state of the art and Section 6.4 discusses the results. Finally, Section 6.5 concludes the chapter.

6.2 Deep Radial Basis Function Data Descriptor (D-RBFDD) Networks

In the previous chapter, it was shown that the RBFDD network can become an even more effective anomaly detector, if we eliminated the restriction of having radial kernels and instead, used elliptical kernels, that is EBFDD. Although RBFDD and EBFDD networks are effective, they are shallow neural networks with one hidden layer and are unlikely to perform well when trained on raw data. So far, all the datasets used in this thesis are composed of high-level, information rich features. The RBFDD and EBFDD models are likely to suffer in scenarios where we are dealing with raw high-dimensional datasets. In such scenarios, deep models can be used to learn to extract useful features from raw data for the final layers of a model to work with. We explore three approaches to add depth to RBFDD networks (illustrated in Fig. 6.1):

- *Transfer learning* (Weiss et al., 2016) is exploited and the latent representation produced at the final layer of a network that is already trained on a large generic dataset is presented to the RBFDD network.
- The latent representation from a pre-trained network, such as that described above, can be *fine-tuned* using the cost function optimised when training an RBFDD network.
- Computational layers placed before the RBFDD network can be trained from random initialisation as part of the end-to-end deep RBFDD training process.



(a) Fix-Res + RBFDD

(b) Fine-Res + RBFDD

(c) Deep RBFDD

Figure 6.1: Three approaches to deepening RBFDD networks. The red lines highlight the fixed portion of each model.

For the approach using transfer learning, referred to as *Fix-Res + RBFDD*, we use a fixed, pre-trained ResNet-18 model (He et al., 2016) trained on the ImageNet (He et al., 2016) dataset, and extract the latent representation after its last convolutional layer for each data instance. This representation is then passed to a standard RBFDD model. In the second approach, referred to as *Fine-Res + RBFDD*, again we connect a pre-trained ResNet-18 model to an RBFDD model. In this case, however, we fine-tune the last 4 convolutional layers, and the last 4 batch-norm layers of the pre-trained ResNet-18 model as part of training the RBFDD model using the RBFDD cost function in Eq.(6.1), the details of which have been presented in Chapter 4.

$$E = \sum_{i=1}^{N} \left(\frac{1}{2} \left[(1 - y_i)^2 + \beta \sum_{h=1}^{H} s_h^2 + \lambda \sum_{h=1}^{H} w_h^2 \right] \right) \tag{6.1}$$

This fine-tuning step is expected to make the latent representation passed to the RBFDD model more appropriate for anomaly detection, and improve its overall performance.

The final approach, that we refer to as the Deep RBFDD (D-RBFDD) network, attaches randomly initialised computational layers before the RBFDD layer and trains the entire network in an end-to-end fashion based on minimising the cost function in Eq.(6.1). Provided that the cost function can generate sufficient signal to train the

entire deep model, the advantage of this method is that by using end-to-end training the latent representation that this network passes to the RBFDD layer will be more suited to anomaly detection than the representation generated by the pre-trained classification network, even when that representation is fine-tuned. In D-RBFDD, we add layers preceding the RBFDD layer, following the simple LeNet-5 network (the first deep RBF network) architecture (LeCun et al., 1998) as it has been shown to be a simple, but effective architecture. In LeNet-5 an RBF network classifier is mounted on the model and has proven effective for multi-class classification, which motivates the use of LeNet-5 architecture for D-RBFDD network and the task of one-class classification. The overall D-RBFDD network architecture is illustrated in Fig. 6.2. On the outputs of both convolutional layers, *C1* and *C3*, first the hyperbolic tangent, $tanh$, non-linearity and then average pooling sub-sampling are applied. Also, on the output of the fully-connected layer, *F5*, the hyperbolic tangent, $tanh$, non-linearity is applied.

To facilitate the application of k-means clustering in the pre-training phase of training an RBFDD network we apply the hyperbolic tangent, $tanh$, activation function proposed in (Orr and Müller, 1998) to the latent representations coming into the RBFDD layer. This ensures that the k-means algorithm is provided with a bounded latent representation and leads to better model initialisation.

The training process of D-RBFDD is similar to that of RBFDD outlined in Chapter 4, with a small difference in pre-training. After the parameters of the computational layers before the RBFDD layer are randomly initialised, the entire training data has to pass through them first. Then, using k-means clustering, the transformed training data is used to initialise the mean, μ, and the spread, s, of the Gaussian kernels in the RBFDD layer.

Deepening the RBFDD network should improve the performance of the RBFDD network and make it a stronger anomaly detector, in particular, when dealing with raw high-dimensional input data.

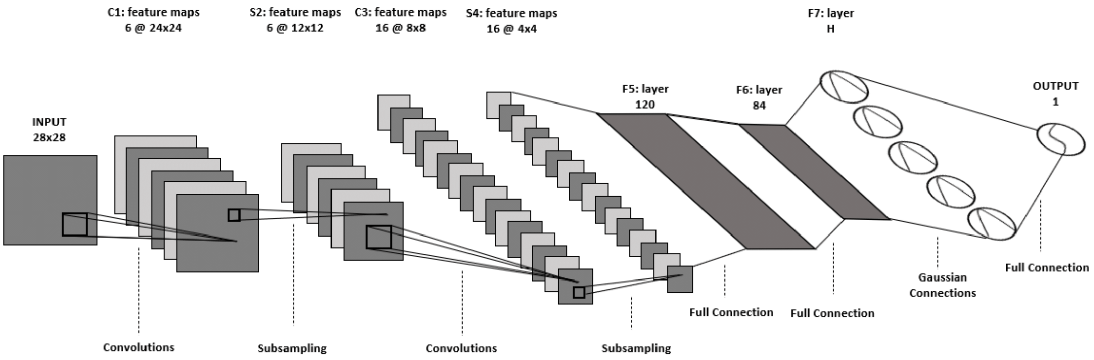


Figure 6.2: The Deep RBFDD Anomaly Detector

6.3 Evaluation Experiment

The evaluation experiment is designed to assess the performance of the D-RBFDD approach and compare it to the state of the art, as well as the approaches that are based on transfer learning, where possible. The raw datasets used in the experiment are two well-known image datasets MNIST and Fashion-MNIST, as well as one real-world highly-imbalanced anomaly detection time series dataset, the MIT-BIH Arrhythmia Database (See Chapter 3 for details). We have designed an experiment to evaluate the performance of the different approaches to deepening RBFDD networks, and to compare these to state of the art anomaly detection approaches. The following state of the art anomaly detection approaches are included in this experiment: One-class Deep Support Vector Data Description (DeepSVDD-OC) networks (Ruff et al., 2018), Soft-boundary Deep Support Vector Data Description (DeepSVDD-SB) networks (Ruff et al., 2018), One-Class Support Vector Machines (OCSVMs), Isolation Forest (iForest), RBFDD networks, and deep Convolutional Auto Encoders (CAEs) (Ribeiro et al., 2018). In all cases only *normal* data is used during model training.

To further investigate how effectively representations can be transferred from pre-trained classification networks to anomaly detection tasks, in the case of the OCSVM and iForest models (as well as RBFDD networks), we have also considered the scenario where the latent representation learned by a pre-trained classification network is used as input to these networks. We also use the latent representation learned by the deepened version of RBFDD that fine-tunes the pre-trained classification model representation as input to these shallow models to further investigate the impact of

fine-tuning. For each algorithm, dataset and scenario combination we use the experimental design described in Section 3.4. We also perform hyper-parameter tuning using a grid search for each hyper-parameter combination according to Section 3.4. The ranges of searched hyper-parameters for each algorithm used are listed in Table 6.1.

Table 6.1: The hyper-parameter ranges explored for each algorithm, where D denotes the dimensionality of the input data to the RBFDD layer in both the RBFDD network and the D-RBFDD network. Below η is the learning rate, and for DeepSVDD, λ is the weight decay coefficient, and ν controls the trade-off between violations of the boundary and the volume of the hyper-sphere. The rest of the hyper-parameters have already been introduced in Chapter 4 in Table 4.1
.

Algorithms	Hyper-parameters	Investigated Hyper-Parameters
RBFDD	number of kernels (H)	[0.01D, 0.05D, ..., 0.25D, 0.50D, 0.75D, 2D]
	η	[0.01, 0.001, 0.0001]
	β	[0.9, 0.5, 0.01, 0.0001]
	λ	[0.9, 0.5, 0.01, 0.0001]
D-RBFDD	number of kernels (H)	[0.25D, 0.50D, 0.75D, 2D]
	η	[0.01, 0.001, 0.0001]
	β	[0.9, 0.5, 0.01, 0.0001]
	λ	[0.9, 0.5, 0.01, 0.0001]
DeepSVDD	η	[0.01, 0.0001, 0.00001]
	learning rate milestone for the scheduler	0.75
	λ	[0.9, 0.5, 0.01, 0.0001]
	ν (Only for Soft-Boundary)	[0.9, 0.5, 0.01, 0.0001]
OCSVM	ν	[0.9, 0.5, 0.01, 0.0001]
	γ	[0.9, 0.5, 0.01, 0.0001]
CAE-1/CAE-2	η	[0.01, 0.001, 0.0001]
	cost functions	mean squared error, cross-entropy
Isolation Forest	number of estimators	[100, 200, 500, 800, 1000]

Each state of the art approach compared in this experiment is tuned to achieve its best possible performance. For all OCSVM models we use Gaussian kernels, as recommended in (Khan and Madden, 2013). The hyper-parameters tuned for OCSVM models are ν, and γ. For iForest, the only hyper-parameter to be tuned is the number of estimators.

To explore their performance we use different CAE architectures, each with similar capacity to the D-RBFDD model. For the image classification datasets CAE-1 has two 2D convolutional layers in the encoder and two transposed 2D convolutional layers in the decoder, while CAE-2 has three convolutional layers in the encoder and three transposed convolutional layers in the decoder. For the ECG dataset, CAE-1 has two 1D convolutional layers in the encoder and two transposed 1D convolutional layers in the decoder. CAE-2 has the same structure, however, the second 1D convolutional

layer has twice the number of convolutional filters compared to CAE-1. For all CAEs, rectified linear activation functions and max-pooling are used at each layer.

For the RBFDD network and the D-RBFDD network, the hyper-parameters that are tuned are the number of Gaussians in the hidden layer, and the coefficients of the cost function (Eq. (4.1)): β and λ (whose values fall in the range of $(0, 1]$). The D-RBFDD network, is based on the LeNet-5 network architecture (LeCun et al., 1998). For the ECG dataset we replace the 2D convolutions in this architecture with 1D convolutions.

The DeepSVDD network proposed by Ruff et al. (2018) uses a LeNet-type network architecture, and we use this for the image datasets. For the ECG dataset we replace this with the 1D LeNet-5 architecture used in the D-RBFDD network. In both versions of DeepSVDD the weight decay coefficient λ is a tuned hyper-parameter. For DeepSVDD-SB, ν, is also a tunable hyper-parameter, whose role is to control the trade-off between violations of the boundary and the volume of the hypersphere. Following the training method in (Ruff et al., 2018), the training of DeepSVDD models also includes a learning rate scheduler that reduces the learning rate by a factor of 10 after 75% of the specified training epochs have been completed.

In the experiments using the image classification datasets, we use a pre-trained ResNet-18 model (He et al., 2016) trained on the ImageNet (Deng et al., 2009) dataset[1] for transfer learning. No transfer learning is used for the ECG dataset, as reliable large-scale pre-trained models for ECG sensor data are not available.

6.4 Results and Discussions

The results of the experiments based on the MIT-BIH Arrhythmia and image classification datasets are detailed in Tables 6.2 and 6.3. For each anomaly detection scenario the different approaches have been ranked and the average ranks for each approach are summarised in the last column of each table.

The results in Table 6.3 show the effectiveness of deepening RBFDD for raw datasets, and allow us to compare the three different strategies for deepening described in Section 6.2. The fact that the D-RBFDD model has out-performed the RBFDD model, in the majority of cases, demonstrates the value of using the deep model to gener-

[1]The ResNet-18 implementation used is available at: `https://github.com/pytorch/vision/tree/master/torchvision/models/resnet.py`

ate a latent representation suitable for use by RBFDD. Moreover, it is interesting to note that none of the models that use the latent representation output by the fixed, pre-trained ResNet-18 model out-perform their equivalent models trained on the raw, high-dimensional representation. This is the case for the RBFDD models as well as for the OCSVM and iForest models. The fixed pre-trained ResNet-18 model has been trained for a multi-class classification objective and the latent representations generated by this network seem to be too entangled with that task to be very useful for anomaly detection.

Table 6.2: Experiment results on the MIT-BIH Arrhythmia dataset. The label of each anomalous class is given at the top of the columns (for the label descriptions see Section 3.2.2). The values of each cell are AUC scores followed by the relative rank in parentheses. The average rank per algorithm is given in the last column.

	L	R	V	/	One-vs-All	Avg. Rank
iForest	0.5743 (8)	0.7118 (8)	0.6819 (8)	0.7713 (8)	0.6808 (8)	8.0 (8)
OCSVM	0.6684 (5)	0.7582 (6)	0.8647 (5)	0.8591 (6)	0.7830 (5)	5.4 (5)
RBFDD	0.7002 (4)	0.8182 (4)	0.8722 (4)	0.8947 (5)	0.8043 (4)	4.2 (4)
CAE-1	0.6331 (6)	0.7525 (7)	0.7416 (6)	0.8139 (7)	0.7174 (6)	6.4 (7)
CAE-2	0.5994 (7)	0.8139 (5)	0.7407 (7)	0.9395 (1)	0.7023 (7)	5.4 (5)
DeepSVDD-OC	0.7700 (3)	0.8352 (3)	0.9241 (3)	0.9187 (4)	0.8324 (3)	3.2 (3)
DeepSVDD-SB	0.7835 (1)	0.8596 (1)	0.9363 (1)	0.9316 (2)	0.8346 (2)	**1.4 (1)**
D-RBFDD	0.7723 (2)	0.8458 (2)	0.9361 (2)	0.9261 (3)	0.8507 (1)	2.0 (2)

Table 6.3: Experiment results on the MNIST and Fashion MNIST datasets. Each column is labelled N-A, where N = normal class and A = anomalous class. The values in each cell are AUC scores followed by relative rank in parentheses. The average rank per algorithm is given in the last column. The labels for Fashion MNIST are: T: T-shirts/tops, B: Ankle boots, S:Shirts, Sn:Sneakers and Sa:Sandals.

	MNIST						Fashion MNIST				
	0 - 1	7 - 1	4 - 9	7 - 9	9 - 4	9 - 7	T- B	T - S	Sa - Sn	B - Sa	Avg. Rank
iForest	0.9648 (7)	0.7725 (12)	0.6296 (7)	0.7948 (5)	0.7484 (6)	0.7355 (6)	0.9963 (3)	**0.8182** (1)	0.5394 (10)	0.9536 (7)	6.4 (6)
Fix-Res + iForest	0.4795 (14)	0.4608 (14)	0.5904 (12)	0.5939 (14)	0.5595 (14)	0.5941 (14)	0.6212 (14)	0.5614 (13)	0.5304 (11)	0.4782 (14)	13.4 (14)
Fine-Res + iForest	0.7609 (11)	0.9698 (5)	0.5807 (14)	0.6910 (9)	0.6898 (10)	0.6566 (11)	0.9761 (10)	0.4585 (14)	0.4957 (14)	0.7753 (10)	10.8 (12)
OCSVM	0.9962 (3)	0.9623 (6)	0.8320 (2)	**0.9209** (1)	0.9245 (2)	0.9125 (2)	0.9967 (2)	0.7872 (6)	0.5708 (9)	0.9708 (5)	3.8 (3)
Fix-Res + OCSVM	0.5506 (13)	0.6924 (13)	0.5808 (13)	0.6098 (13)	0.6035 (11)	0.6268 (13)	0.8981 (12)	0.5992 (12)	0.5100 (12)	0.6011 (13)	12.5 (13)
Fine-Res + OCSVM	0.8542 (9)	0.9746 (3)	0.5975 (11)	0.7181 (7)	0.7119 (8)	0.6850 (9)	0.9780 (8)	0.6429 (11)	0.5036 (13)	0.7232 (12)	9.1 (9)
RBFDD	**0.9988** (1)	0.9722 (4)	0.7585 (3)	0.8187 (4)	0.8069 (5)	0.8583 (5)	0.9954 (4)	0.7898 (5)	0.6562 (6)	0.9830 (4)	4.1 (4)
Fix-Res + RBFDD	0.7923 (10)	0.8332 (11)	0.6273 (8)	0.6157 (12)	0.6941 (9)	0.6714 (10)	0.9740 (11)	0.6881 (9)	0.6197 (8)	0.7299 (11)	9.9 (11)
Fine-Res + RBFDD	0.9422 (8)	0.9119 (9)	0.6217 (9)	0.6612 (11)	0.7236 (7)	0.7152 (7)	0.9901 (7)	0.8055 (2)	**0.7310** (1)	0.8431 (9)	7.0 (7)
CAE-1	0.7000 (12)	0.8964 (10)	0.7137 (6)	0.8934 (3)	0.5688 (13)	0.6419 (12)	0.8426 (13)	0.6823 (10)	0.6379 (7)	0.8875 (8)	9.4 (10)
CAE-2	0.9914 (5)	0.9514 (7)	0.6110 (10)	0.6614 (10)	0.5843 (12)	0.7139 (8)	0.9764 (9)	0.7378 (8)	0.7013 (4)	0.9647 (6)	7.9 (8)
DeepSVDD-OC	0.9906 (6)	**0.9943** (1)	0.7455 (4)	0.7071 (8)	0.9140 (3)	0.8950 (4)	0.9950 (6)	0.8001 (4)	0.6567 (5)	0.9871 (3)	4.4 (5)
DeepSVDD-SB	0.9957 (4)	0.9916 (2)	0.7365 (5)	0.7417 (6)	0.9132 (4)	0.8969 (3)	0.9951 (5)	0.8020 (3)	0.7023 (3)	**0.9896** (1)	3.6 (2)
D-RBFDD	0.9981 (2)	0.9512 (8)	**0.8450** (1)	0.8971 (2)	**0.9480** (1)	**0.9137** (1)	**0.9987** (1)	0.7459 (7)	0.7161 (2)	0.9887 (2)	**2.7** (1)

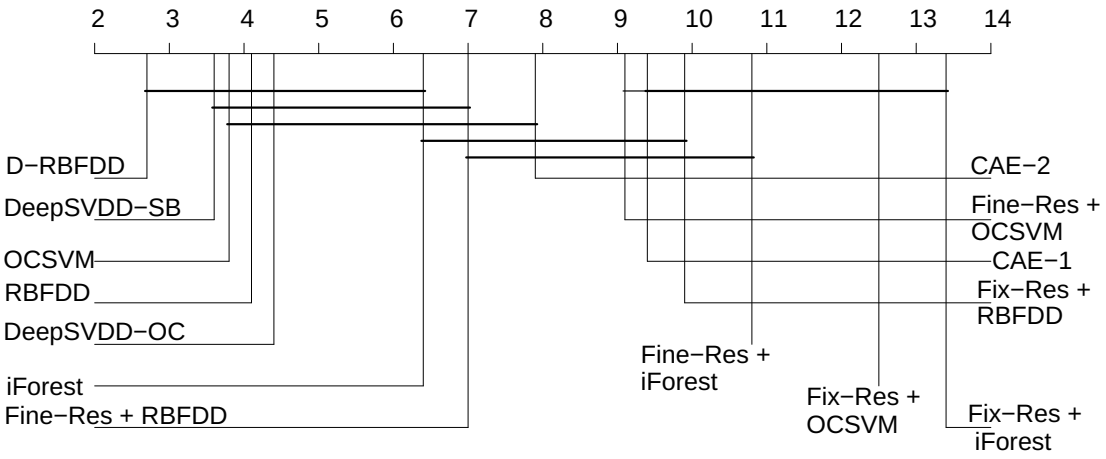
This is further underlined by the fact that, in almost all cases, the performance of the models (RBFDD, OCSVM, and iForest) using the latent representations that arise from the version of the ResNet-18 model fine-tuned using the RBFDD network improves over the versions of the models trained using the representations from the fixed ResNet-18 model. However, it is important to note that in most cases this performance was still not better than the models working on raw data.

Together these results show that deepening RBFDD networks allows them to work effectively with raw inputs, and that the D-RBFDD approach is the best way to do this out of those compared. This conclusion is reinforced by the results based on the ECG dataset shown in Table 6.2. In experiments using this dataset, D-RBFDD out-performs RBFDD in all cases.

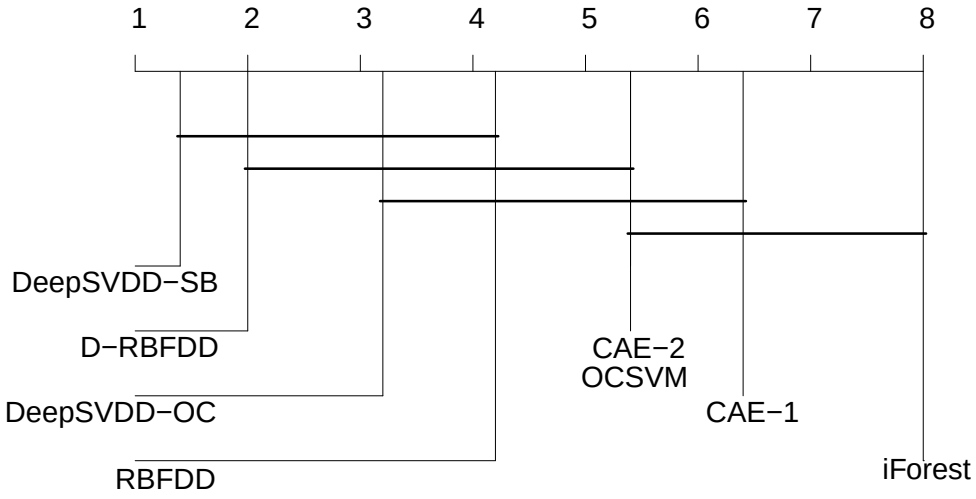
By examining the results for the image classification datasets in Table 6.3 and those based on the ECG dataset in Table 6.2 together, we can evaluate how D-RBFDD compares to other state of the art anomaly detection algorithms. In the image classification dataset cases, the results show that, overall, the D-RBFDD network out-performs the other algorithms as it has the best average rank. On the ECG dataset, DeepSVDD-SB has a slightly better average rank than D-RBFDD, although D-RBFDD performs better in the *One-vs-All* scenario which is particularly important for anomaly detection as it is likely that anomalies will arise from very different data distributions.

To further investigate and understand the overall differences between the performances of the different algorithms and the effectiveness of D-RBFDD, we perform non-parametric statistical significance tests for multiple classifier comparison. Following García et al. (2010) a Friedman test followed by a Finner p-value correction was performed on the results in both Tables 6.2 and 6.3. This test analyses the difference in performance between each pair of algorithms with respect to the different anomaly detection scenarios.

The statistical test results for the image classification and ECG datasets are summarised in the critical difference plots (with significance level of $\alpha = 0.05$) in Fig. 6.3a and Fig. 6.3b respectively. In these figures, any two algorithms not connected with bold horizontal lines are significantly different. The p-values of the statistical tests and the pairwise win/lose/tie results are shown in Table 6.4 for the MNIST and Fashion MNIST datasets, and in Table 6.5 for the MIT-BIH Arrhythmia dataset.



(a) On cases of MNIST and Fashion MNIST datasets



(b) On cases of MIT-BIH Arrhythmia dataset

Figure 6.3: Critical difference plots from Friedman's aligned test results using a significance level of 0.05 on the different anomaly detection scenarios. Algorithms *not* connected with horizontal bars are significantly different.

In Tables 6.4 and 6.5 the lower diagonal shows the p-values of the post-hoc Friedman test (with the Finner p-value correction) with the corresponding significance level with which the null-hypothesis can be rejected. A simple win/lose/tie count with respect to the MNIST, Fashion MNIST, and MIT-BIH Arrhythmia datasets for each pair of algorithms is shown in the upper diagonal of Tables 6.4 and 6.5. For example in Table 6.4, when comparing DeepSVDD-SB and D-RBFDD, we see the value (7/3/0). This means that D-RBFDD had better scores in 7 cases and worse scores in 3 cases out of a total of 10 cases.

Table 6.4: Results of Friedman test with Finner p-value adjustment and pairwise win/lose/tie counts over the different cases for the MNIST and Fashion MNIST datasets. Upper diagonal: win/lose/tie. Lower diagonal: Friedman test with p-values after the Finner adjustment. * $\alpha = 0.1$, ** $\alpha = 0.05$ and *** $\alpha = 0.01$.

	D-RBFDD	DeepSVDD-SB	OCSVM	RBFDD	DeepSVDD-OC	iForest	Fine-Res + RBFDD	CAE-2	Fine-Res + OCSVM	CAE-1	Fix-Res + RBFDD	Fine-Res + iForest	Fix-Res + OCSVM	Fix-Res + iForest
D-RBFDD		7/3/0	7/3/0	7/3/0	8/2/0	9/1/0	8/2/0	9/1/0	9/1/0	10/0/0	10/0/0	9/1/0	10/0/0	10/0/0
DeepSVDD-SB	0.6870		4/6/0	6/4/0	7/3/0	7/3/0	8/2/0	10/0/0	10/0/0	9/1/0	10/0/0	10/0/0	10/0/0	10/0/0
OCSVM	0.6175	0.9149		5/5/0	6/4/0	9/1/0	8/2/0	9/1/0	9/1/0	9/1/0	9/1/0	9/1/0	10/0/0	10/0/0
RBFDD	0.5251	0.8075	0.8812		4/6/0	8/2/0	8/2/0	9/1/0	9/1/0	9/1/0	10/0/0	10/0/0	10/0/0	10/0/0
DeepSVDD-OC	0.4489	0.7068	0.7758	0.8812		7/3/0	8/2/0	8/2/0	9/1/0	9/1/0	10/0/0	10/0/0	10/0/0	10/0/0
iForest	0.0926 *	0.2028	0.2387	0.2924	0.3660		8/2/0	6/4/0	9/1/0	6/4/0	8/2/0	9/1/0	10/0/0	10/0/0
Fine-Res + RBFDD	0.0495 **	0.1223	0.1475	0.1923	0.2387	0.7758		6/4/0	8/2/0	7/3/0	9/1/0	8/2/0	10/0/0	10/0/0
CAE-2	0.0149 **	0.0495 **	0.0605 *	0.0854 *	0.1154	0.5006	0.6870		6/4/0	8/2/0	8/2/0	7/3/0	9/1/0	10/0/0
Fine-Res + OCSVM	0.0032 ***	0.0110 **	0.0135 **	0.0200 **	0.0300 **	0.2202	0.3418	0.5860		5/5/0	6/4/0	9/1/0	9/1/0	9/1/0
CAE-1	0.0019 ***	0.0073 ***	0.0096 ***	0.0135 **	0.0200 **	0.1764	0.2713	0.5006	0.8812		5/5/0	5/5/0	8/2/0	10/0/0
Fix-Res + RBFDD	0.0009 ***	0.0033 ***	0.0046 ***	0.0073 ***	0.0110 **	0.1154	0.1923	0.3660	0.7068	0.8075		6/4/0	10/0/0	10/0/0
Fine-Res + iForest	0.0001 ***	0.0009 ***	0.0012 ***	0.0019 ***	0.0032 ***	0.0441 **	0.0854 *	0.1923	0.4489	0.5251	0.6870		7/3/0	7/3/0
Fix-Res + OCSVM	0.0000 ***	0.0000 ***	0.0000 ***	0.0001 ***	0.0001 ***	0.0046 ***	0.0110 **	0.0339 **	0.1223	0.1615	0.2387	0.4489		8/2/0
Fix-Res + iForest	0.0000 ***	0.0000 ***	0.0000 ***	0.0000 ***	0.0000 ***	0.0012 ***	0.0032 ***	0.0110 **	0.0495 **	0.0676*	0.1154	0.2387	0.6870	

Table 6.5: Results of Friedman test with Finner p-value adjustment and pairwise win/lose/tie counts over the different cases for the MIT-BIH Arrhythmia dataset. Upper diagonal: win/lose/tie. Lower diagonal: Friedman test with p-values after the Finner adjustment. * $\alpha = 0.1$, ** $\alpha = 0.05$ and *** $\alpha = 0.01$.

	DeepSVDD-SB	D-RBFDD	DeepSVDD-OC	RBFDD	OCSVM	CAE-2	CAE-1	iForest
DeepSVDD-SB		4/1/0	5/0/0	5/0/0	5/0/0	4/1/0	5/0/0	5/0/0
D-RBFDD	0.7116		5/0/0	5/0/0	5/0/0	4/1/0	5/0/0	5/0/0
DeepSVDD-OC	0.3395	0.5369		5/0/0	5/0/0	4/1/0	5/0/0	5/0/0
RBFDD	0.1573	0.2707	0.5738		5/0/0	4/1/0	5/0/0	5/0/0
OCSVM	0.0450 **	0.0851 *	0.2707	0.5369		3/2/0	5/0/0	5/0/0
CAE-2	0.0450 **	0.0851 *	0.2707	0.5369	1.0000		2/3/0	5/0/0
CAE-1	0.0116 **	0.0250 **	0.0960 *	0.2707	0.5738	0.5738		5/0/0
iForest	0.0006 ***	0.0015 ***	0.0135 **	0.0487 **	0.1902	0.1902	0.3951	

Fig. 6.3a shows that D-RBFDD performs significantly and consistently better than the following algorithms: Fine-Res + RBFDD, CAE-2, Fine-Res + OCSVM, CAE-1, Fix-Res + RBFDD, Fix-Res + OCSVM, Fix-Res + iForest, Fine-Res + iForest. In the case of DeepSVDD-SB, OCSVM, RBFDD, DeepSVDD-OC and iForest, the null-hypothesis of the test could not be rejected with a significance level of $\alpha = 0.05$, however, D-RBFDD performed better in average rank. On the other hand, in a simple and direct pairwise win/lose/tie based comparison, D-RBFDD won in at least 70% and up to 100% of the cases when compared to the other algorithms. This indicates that D-RBFDD typically performs as good as or better than the benchmark algorithms it is compared to.

From Fig. 6.3b we can see that DeepSVDD-SB attained the best average rank of 1.4 on the ECG dataset. D-RBFDD achieved similar performance with an average rank of 2.0. In most of the scenarios DeepSVDD-SB has performed slightly better than D-RBFDD, however, interestingly D-RBFDD achieved the best performance in the *one-vs-all* case. Although, from an overall point of view for DeepSVDD-SB and D-RBFDD, the null hypothesis could not be rejected in any of the significance levels. D-RBFDD performed better than iForest and CAE-1 at the significance level of $\alpha = 0.01$ and $\alpha = 0.05$ respectively, and performed better than OCSVM and CAE-2 with a significance level of $\alpha = 0.1$.

Overall, these results indicate that adding extra computational layers to RBFDD makes it a much more effective anomaly detector for problems with raw data representations. Also, selecting D-RBFDD will lead to at least similar or many times better performance than the other approaches, making it an attractive solution for anomaly detection for these types of datasets. We believe that this strong performance, coupled with the easy interpretability and adaptability of approaches based on RBF networks make D-RBFDD a compelling approach. In particular, the effectiveness of end-to-end approaches optimised using a loss function with a direct anomaly detection objective, rather than one based on reconstruction error, motivates our proposed D-RBFDD network. This makes a deep end-to-end anomaly detector based on RBF networks, such as D-RBFDD, an attractive approach. D-RBFDD is a fast, effective anomaly detector, trained in an end-to-end fashion and capable of learning latent representations directly aligned with an anomaly detection objective, and that lends itself to easy interpretation and adaptation to changing concepts of normality.

In order to provide some insight regarding the training time complexity of D-RBFDD,

we need to use a method that can be applied to all the models in our experiments for a fair comparison. One could look at this in terms of number of trainable parameters, however, it is not possible to talk about the number of trainable parameters for the iForest and OCSVM models in a meaningful way. In addition the Big-O complexity is not reported for all the models in the literature. For instance, the creators of DeepSVDD have not been specific regarding the time complexity of their model.

Hence, for a fair comparison and in order to gain some insight regarding the comparative computational complexity of the models, we choose one experiment scenario from MNIST and one from MIT-BIH Arrhythmia dataset. We would train the models using their best performing hyper-parameters, and report the training time per model. Since the pre-trained ResNet-18 is not applicable in the MIT-BIH Arrhythmia dataset, we will not include the models based on ResNet-18 in here. For all artificial neural networks, the total training time, as well as, the training time per epoch are reported. The time for computing the centres of the Gaussians in the case of D-RBFDD, and the time for computing the centres in the case of DeepSVDD-OC and DeepSVDD-SB are reported as well. Finally, The total number of epochs are set based on the best performing value per model.

Table 6.6: Total training time, in seconds, of the models with their best performing hyper-parameters

	MNIST	MIT-BIH Arrhythmia
	7-9	One-vs-All
iForest	7.78s	201.37s
OCSVM	6.02s	36.33s
RBFDD	264.0s (K-means 14s + 0.50 s/epoch $\times$ 500 epochs)	50.76s (K-means 0.76s + 2.50 s/epoch $\times$ 20 epochs)
CAE-1	25.0s (0.25 s/epoch $\times$ 100 epochs)	70.60s (3.53 s/epoch $\times$ 20 epochs)
CAE-2	26.0s (0.26 s/epoch $\times$ 100 epochs)	1825.0s (3.65 s/epoch $\times$ 500 epochs)
DeepSVDD-OC	29.0s (Centre initialisation 3s + 0.26 s/epoch $\times$ 100 epochs)	295.0s (Centre initialisation 10s + 2.85 s/epoch $\times$ 100 epochs)
DeepSVDD-SB	30.0s (Centre initialisation 3s + 0.27 s/epoch $\times$ 100 epochs)	298.0s (Centre initialisation 10s + 2.88 s/epoch $\times$ 100 epochs)
D-RBFDD	31.15s (K-means 5.15s + 0.26 s/epoch $\times$ 100 epochs)	69.52s (K-means 12.12s + 2.87 s/epoch $\times$ 20 epochs)

In particular, in the case of D-RBFDD, DeepSVDD-OC and DeepSVDD-SB, it can be seen that the per epoch training times are very close. In the MIT-BIH Arrhythmia dataset, both DeepSVDD-OC and DeepSVDD-SB achieve their best performance after 100 epochs, whereas DRBFDD does so after 20 epochs, and hence the big gap in their total training time (i.e., 295.0s, 298.0s, and 69.52s respectively).

6.5 Summary

In this chapter, we have proposed a deep one-class neural network, the D-RBFDD network, that adds convolutional layers before an RBFDD network. The D-RBFDD network is trained in an end-to-end fashion on an objective that is designed specifically for anomaly detection. We have shown that this network has successfully turned the shallow RBFDD network into a deep one-class classifier, suitable for anomaly detection on high-dimensional raw data such as images and sensor data. Unlike some of the state of the art algorithms, in particular approaches based on SVM, D-RBFDD networks are scalable in size and can work with large datasets and high-dimensional data.

In a set of benchmark experiments, using image datasets the D-RBFDD network has shown superior performance to state of the art one-class classifiers—DeepSVDD, OCSVM, iForest, and CAEs. In the case of the ECG dataset, the D-RBFDD network has out-performed all other approaches and produced competitive results to those of the DeepSVDD-SB algorithm, and out-performed it in the *One-vs-All* scenario, which is particularly important for anomaly detection as it is likely that anomalies will arise from very different data distributions. We have also observed that our proposed D-RBFDD model, has indeed out-performed its shallower version, the RBFDD network, in almost all of our benchmark experiments. This suggests that, when dealing with raw data we have benefited from the introduction of depth in the D-RBFDD network.

Our experiments show that transfer learning using a pre-trained ResNet-18 with fixed weights, does not work well for anomaly detection. We believe that the reason for this is that ResNet-18 has been trained for a multi-class classification task and that the latent representations that it generates are too entangled with that task to be useful for anomaly detection. Interestingly, we see that if the final layers of the ResNet-18 model are fine-tuned using the RBFDD network cost function, the performance improves in most cases. Overall, it can be concluded that, selecting D-RBFDD for an anomaly detection task on raw data would lead to performance that is at least as good as or significantly better than current state of the art algorithms.

CHAPTER

SEVEN

CONCLUSION

This chapter concludes the thesis by summarising the main contributions, and discussing some directions for future work.

7.1 Summary of Contributions

This thesis studies the problem of anomaly detection and proposes three variants of one-class RBF networks for the task of anomaly detection, which address the following research questions (restated from Chapter 1):

RQ 1 **Is it possible to turn the Radial Basis Function (RBF) network into an effective anomaly detector that can learn only from the normal class?**

RQ 2 **Will replacing the radial kernels with elliptical kernels in a one-class RBF network make the model a stronger anomaly detector?**

RQ 3 **Is it possible to deepen the one-class RBF network architecture to make an effective anomaly detector applicable to high-dimensional raw data inputs, and what is the best way to do this deepening?**

The significant contributions made within the thesis in response to these research questions are summarised in the rest of this section.

7.1.1 The RBFDD Network

A novel approach to semi-supervised anomaly detection based on the Radial Basis Function (RBF) network (Alpaydin, 2004; Bishop, 2007) is proposed in Chapter 4. This

is the Radial Basis Function Data Descriptor (RBFDD) network (H. Bazargani and Mac Namee, 2018), which relies on a newly defined cost function that allows it to learn from only instances of the normal class.

RBFDD networks inherit certain attractive features of RBF networks. Some of these attractive features are the ability of RBF networks to be trained using gradient descent, which allows them to be deepened and trained in an end-to-end fashion, their interpretability (Jin and Sendhoff, 2003), their adaptability to changing concepts (Liu et al., 2020; Han et al., 2011; Rapaka et al., 2003), and finally their fast and efficient training. This makes RBFDD networks an attractive one-class algorithm for the task of anomaly detection that would have the potential to benefit/inherit these properties from the RBF networks.

In a series of benchmark experiments using over 50 scenarios, RBFDD networks have been shown to out-perform state of the art anomaly detection techniques, and show particular promise for the one-vs-all anomaly detection scenario common in many real-world problems. The results of these experiments show that RBFDD networks are capable of learning the distribution of the normal data and they can be used to effectively identify anomalies. This contribution addresses *RQ1*.

7.1.2 The EBFDD Network

The use of radial kernels in RBFDD networks, however, is a potential limitation. In Chapter 5 the Elliptical Basis Function Data Descriptor (EBFDD) network (H. Bazargani and Mac Namee, 2019) is proposed. The EBFDD network addresses the limitation of RBFDD networks by allowing unequal variances per input feature, and also correlation between the input features. This dictates using a non-diagonal covariance matrix in the kernels of the EBFDD network, which makes the model more flexible—unlike the RBFDD network, in an EBFDD network, kernels are of elliptical shape and capable of rotation. In a series of benchmark experiments, the EBFDD network has out-performed its predecessor the RBFDD network as well as current state of the art approaches. We have paid a price for this good performance, however, as the computational cost of EBFDD is high due to matrix inversion operations required. This contribution addresses research question *RQ2*.

7.1.3 The D-RBFDD Network

Another potential limitation of RBFDD networks comes from the fact that they are shallow neural networks, hence, they are designed to work with datasets composed of well-behaved features. When it comes to raw high-dimensional input data, however, the performance of RBFDD networks (similar to any other shallow neural network) will suffer. In Chapter 6, this issue is addressed and three methods for deepening RBFDD networks are investigated. The first two rely on transfer learning, where the latent representations of a pre-trained model are fed into the RBFDD network, whereas the last method places computational layers before the RBFDD network, where the entire network can be trained from random initialisation in an *end-to-end* fashion.

It seems that the latent representations generated by pre-trained models are too entangled with the task of multi-class classification to be very useful for anomaly detection. The third proposed method of deepening, however, appears to be an effective way of deepening RBFDD networks, resulting in Deep RBFDD (D-RBFDD) (H. Bazargani et al., under review) models. In a set of benchmark experiments using two image datasets and one real-world ECG anomaly detection dataset, it is shown that D-RBFDD networks have improved upon the performance of RBFDD networks, while either out-performing state of the art approaches or showing competitive performance with them. This contribution addresses research question *RQ3*.

7.2 Future Work

The directions for the future work can be summarised as follows:

Learning from streams: According to the findings in this thesis, all flavors of one-class RBF networks (RBFDD, EBFDD, and D-RBFDD) have demonstrated a strong performance on static datasets. This is the minimum we would expect from an effective anomaly detector before trying to extend its abilities to more challenging scenarios of anomaly detection, in particular, learning from streams. The main challenge with streams is the fact that the distribution of the *incoming* data may change as a function of time, and may change the very concept of *normality*, which is known as concept drift (Widmer and Kubat, 1996). In fu-

ture work, methods of transforming RBFDD networks (and their EBFDD and D-RBFDD variants) into on-line anomaly detectors will be explored. This is an interesting direction since RBF networks have already been shown to be highly adaptable to changing concepts (Liu et al., 2020; Han et al., 2011; Rapaka et al., 2003). An additional component could be used for detecting concept drift in the incoming stream which, upon detecting a drift in concept, could trigger an RBFDD network (or its EBFDD and D-RBFDD variants) to adapt its structure, as well as, parameters. This could be done by pruning or adding Gaussian kernels in order to forget or learn the old or new concepts of normality and adapt fast to novel emerging distributions and concepts of normality or anomaly in the incoming data stream. Clearly, the learnable parameters of the model need to be tuned upon the emergence of a new concept.

Interpretability: The RBFDD, EBFDD, and D-RBFDD networks have shown strong performance in detecting anomalies. It is important to understand why they flag a certain data instance as anomalous in order to be able to trust the model. Interpretability becomes particularly important in sensitive applications such as cancer detection (Cruz-Roa et al., 2013) where the cost of a wrong decision can be catastrophic. For example, an anomaly detector developed for cancerous cell detection might make accurate decisions, however, for all the wrong reasons, and interpretability can help us detect such potentially wrong reasoning before deploying the model to avoid putting lives in danger. Since RBF networks easily lend themselves to interpretation (Jin and Sendhoff, 2003), we believe it is possible to extend this property to RBFDD, EBFDD, and D-RBFDD networks to gain valuable insights from predictions made by the models. We have already seen in Chapters 4 and 5 that the learned parameters of one-class RBF networks have potential for being interpretable. As a part of future work, maximising this interpretability is certainly desirable. In particular, techniques such as Activation Maximisation (AM) (Erhan et al., 2009), and generating counter-factual explanations (Goyal et al., 2019) will be explored.

One-class hyper-parameter Tuning: Due to the rarity—if not a total lack—of anomalous data in real-life anomaly detection datasets, it would be interesting to see whether hyper-parameter tuning could be done without any need for real anomalous data instances. In future work, a method of augmenting the dataset

with random proxy negative examples (Oliveria et al., 2004a) will be used to see whether hyper-parameter tuning could be done for one-class RBF networks without actual anomalous data but their random proxy representatives. Clearly, the distribution of proxy representatives plays an important role, and needs to be investigated. By defining an appropriate distance measure, we could generate random proxy representatives with different distances from the normal training data and investigate the efficacy of measures as well as the effect of their magnitude on performance of one-class RBF networks. The strategy of data augmentation can also be extended to the normal class, where by applying different manipulations to the normal instances we could add a variety of normal samples in the data. For instance, for image data, augmenting the dataset by introducing different levels of color distortion and random image cropping have proven to result in learning very rich representations in deeper layers in using contrastive learning (Chen et al., 2020). Using contrastive learning on one-class RBF networks using a carefully augmented dataset, has the potential to enrich the learned representations and boost performance.

Transfer learning and anomaly detection: Building on the results described in Chapter 6, we will further explore why the learned representations in pre-trained classification networks did not transfer effectively to the task of anomaly detection. It could be the case that these representations are too entangled with the classification task to be effectively transferred.

BIBLIOGRAPHY

Aggarwal, C.C., 2013. Outlier Analysis. Springer. URL: https://doi.org/10.1007/978-1-4614-6396-2, doi:10.1007/978-1-4614-6396-2.

Alpaydin, E., 2004. Introduction to Machine Learning. MIT Press.

An, J., Cho, S., 2015. Variational autoencoder based anomaly detection using reconstruction probability. Special Lecture on IE 2, 1–18.

Angiulli, F., Pizzuti, C., 2002. Fast outlier detection in high dimensional spaces, in: Elomaa, T., Mannila, H., Toivonen, H. (Eds.), Principles of Data Mining and Knowledge Discovery, 6th European Conference, PKDD 2002, Helsinki, Finland, August 19-23, 2002, Proceedings, Springer. pp. 15–26. URL: https://doi.org/10.1007/3-540-45681-3_2, doi:10.1007/3-540-45681-3_2.

Ashley, E.A., Niebauer, J., 2004. Cardiology explained. Remedica.

Awad, M., Khanna, R., 2015. Support vector machines for classification, in: Efficient Learning Machines. Springer, pp. 39–66.

Baldi, P., 2012. Autoencoders, unsupervised learning, and deep architectures, in: Guyon, I., Dror, G., Lemaire, V., Taylor, G.W., Silver, D.L. (Eds.), Unsupervised and Transfer Learning - Workshop held at ICML 2011, Bellevue, Washington, USA, July 2, 2011, JMLR.org. pp. 37–50. URL: http://proceedings.mlr.press/v27/baldi12a.html.

Berthold, M.R., Diamond, J., 1994. Boosting the performance of RBF networks with dynamic decay adjustment, in: Tesauro, G., Touretzky, D.S., Leen, T.K. (Eds.), Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994], MIT Press. pp. 521–528. URL: http://papers.nips.cc/paper/946-boosting-the-performance-of-rbf-networks-with-dynamic\-decay-adjustment.

Bezerra, M.E.R., Oliveira, A.L.I., Meira, S.R.L., 2007. A constructive RBF neural network for estimating the probability of defects in software modules, in: Proceedings of the International Joint Conference on Neural Networks, IJCNN 2007, Celebrating 20 years of neural networks, Orlando, Florida, USA, August 12-17, 2007, IEEE. pp. 2869–2874. URL: https://doi.org/10.1109/IJCNN.2007.4371415, doi:10.1109/IJCNN.2007.4371415.

Bishop, C.M., 2007. Pattern Recognition and Machine Learning. Springer.

Broomhead, D.S., Lowe, D., 1988. Multivariable functional interpolation and adaptive networks. Complex Syst. 2. URL: http://www.complex-systems.com/abstracts/v02_i03_a05.html.

Campbell, C., Bennett, K.P., 2000. A linear programming approach to novelty detection, in: Leen, T.K., Dietterich, T.G., Tresp, V. (Eds.), Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA, MIT Press. pp. 395–401. URL: https://proceedings.neurips.cc/paper/2000/hash/0e087ec55dcbe7b2d7992d6b69b519fb-Abstract.html.

Chalapathy, R., Menon, A.K., Chawla, S., 2018. Anomaly detection using one-class neural networks. CoRR abs/1802.06360. URL: http://arxiv.org/abs/1802.06360, arXiv:1802.06360.

Chandola, V., Banerjee, A., Kumar, V., 2009. Anomaly detection: A survey. ACM Comput. Surv. 41, 15:1–15:58. URL: https://doi.org/10.1145/1541880.1541882, doi:10.1145/1541880.1541882.

Chen, T., Kornblith, S., Norouzi, M., Hinton, G.E., 2020. A simple framework for contrastive learning of visual representations, in: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, PMLR. pp. 1597–1607. URL: http://proceedings.mlr.press/v119/chen20j.html.

Clifton, L.A., Yin, H., Zhang, Y., 2006. Support vector machine in novelty detection for multi-channel combustion data, in: Wang, J., Yi, Z., Zurada, J.M., Lu, B., Yin, H. (Eds.), Advances in Neural Networks - ISNN 2006, Third International Symposium on Neural Networks, Chengdu, China, May 28 - June 1, 2006, Proceedings, Part III, Springer. pp. 836–843. URL: https://doi.org/10.1007/11760191_122, doi:10.1007/11760191_122.

Crathorne, A.R., 1933. The American Mathematical Monthly 40, 353–355. URL: http://www.jstor.org/stable/2301530.

Cruz-Roa, A.A., Ovalle, J.E.A., Madabhushi, A., Osorio, F.A.G., 2013. A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection, in: Mori, K., Sakuma, I., Sato, Y., Barillot, C., Navab, N. (Eds.), Medical Image Computing and Computer-Assisted Intervention - MICCAI 2013 - 16th International Conference, Nagoya, Japan, September 22-26, 2013, Proceedings, Part II, Springer. pp. 403–410. URL: https://doi.org/10.1007/978-3-642-40763-5_50, doi:10.1007/978-3-642-40763-5_50.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., Li, F., 2009. Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA, IEEE Computer Society. pp. 248–255. URL: https://doi.org/10.1109/CVPR.2009.5206848, doi:10.1109/CVPR.2009.5206848.

Doersch, C., 2016. Tutorial on variational autoencoders. CoRR abs/1606.05908. URL: http://arxiv.org/abs/1606.05908, arXiv:1606.05908.

Dua, D., Graff, C., 2017. UCI machine learning repository. URL: http://archive.ics.uci.edu/ml.

Emmott, A., Das, S., Dietterich, T.G., Fern, A., Wong, W., 2015. Systematic construction of anomaly detection benchmarks from real data. CoRR abs/1503.01158. URL: http://arxiv.org/abs/1503.01158, arXiv:1503.01158.

Erfani, S.M., Rajasegarar, S., Karunasekera, S., Leckie, C., 2016. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. Pattern Recognition 58, 121–134.

Erhan, D., Bengio, Y., Courville, A., Vincent, P., 2009. Visualizing higher-layer features of a deep network. University of Montreal 1341, 1.

Eskin, E., Wenke Lee, Stolfo, S.J., 2001. Modeling system calls for intrusion detection with dynamic window sizes, in: Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01, pp. 165–175 vol.1. doi:10.1109/DISCEX.2001.932213.

Fawcett, T., Provost, F.J., 1999. Activity monitoring: Noticing interesting changes in behavior, in: Fayyad, U.M., Chaudhuri, S., Madigan, D. (Eds.), Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 15-18, 1999, ACM. pp. 53–62. URL: https://doi.org/10.1145/312129.312195, doi:10.1145/312129.312195.

García, S., Fernández, A., Luengo, J., Herrera, F., 2010. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information sciences 180, 2044–2064.

Gardner, A.B., Krieger, A.M., Vachtsevanos, G.J., Litt, B., 2006. One-class novelty detection for seizure analysis from intracranial EEG. J. Mach. Learn. Res. 7, 1025–1044. URL: http://jmlr.org/papers/v7/gardner06a.html.

Ghosh, A.K., Wanken, J., Charron, F., 1998. Detecting anomalous and unknown intrusions against programs, in: 14th Annual Computer Security Applications Conference (ACSAC 1998), 7-11 December 1998, Scottsdale, AZ, USA, IEEE Computer Society. pp. 259–267. URL: https://doi.org/10.1109/CSAC.1998.738646, doi:10.1109/CSAC.1998.738646.

Ghosh, S., Reilly, D.L., 1994. Credit card fraud detection with a neural-network, in: 27th Annual Hawaii International Conference on System Sciences (HICSS-27), January 4-7, 1994, Maui, Hawaii, USA, IEEE Computer Society. pp. 621–630.

Gilroy, E.J., 1995. Statistical methods for groundwater monitoring (robert d. gibbons). SIAM Rev. 37, 628–629. URL: https://doi.org/10.1137/1037145, doi:10.1137/1037145.

Goldberger, A.L., Amaral, L.A., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.K., Stanley, H.E., 2000. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. circulation 101, e215–e220.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. The MIT Press, Cambridge, Massachusetts, London, England.

Goodfellow, I.J., Shlens, J., Szegedy, C., 2015. Explaining and harnessing adversarial examples, in: Bengio, Y., LeCun, Y. (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. URL: http://arxiv.org/abs/1412.6572.

Gopalan, P., Sharan, V., Wieder, U., 2019. Pidforest: Anomaly detection via partial identification, in: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp. 15783–15793. URL: https://proceedings.neurips.cc/paper/2019/hash/eb6dc8aba23375061b6f07b137617096-Abstract.html.

Goyal, Y., Wu, Z., Ernst, J., Batra, D., Parikh, D., Lee, S., 2019. Counterfactual visual explanations, in: Chaudhuri, K., Salakhutdinov, R. (Eds.), Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, PMLR. pp. 2376–2384. URL: http://proceedings.mlr.press/v97/goyal19a.html.

H. Bazargani, M., Mac Namee, B., 2018. Radial basis function data descriptor (rbfdd) network: An anomaly detection approach, in: ODD v5.0 workshop: Outlier Detection De-constructed London, UK.

H. Bazargani, M., Mac Namee, B., 2019. The elliptical basis function data descriptor (EBFDD) network: A one-class classification approach to anomaly detection, in: Brefeld, U., Fromont, É., Hotho, A., Knobbe, A.J., Maathuis, M.H., Robardet, C. (Eds.), Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part I, Springer. pp. 107–123. URL: https://doi.org/10.1007/978-3-030-46150-8_7, doi:10.1007/978-3-030-46150-8_7.

H. Bazargani, M., Pakrashi, A., Mac Namee, B., under review. The deep radial basis function data descriptor (d-rbfdd) network: A one-class neural network for anomaly detection. IEEE Access .

Han, H., Chen, Q., Qiao, J., 2011. An efficient self-organizing RBF neural network for water quality prediction. Neural Networks 24, 717–725. URL: https://doi.org/10.1016/j.neunet.2011.04.006, doi:10.1016/j.neunet.2011.04.006.

Hautamäki, V., Kärkkäinen, I., Fränti, P., 2004. Outlier detection using k-nearest neighbour graph, in: 17th International Conference on Pattern Recognition, ICPR 2004, Cambridge, UK, August 23-26, 2004, IEEE Computer Society. pp. 430–433. URL: https://doi.org/10.1109/ICPR.2004.1334558, doi:10.1109/ICPR.2004.1334558.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, IEEE Computer Society. pp. 770–778. URL: https://doi.org/10.1109/CVPR.2016.90, doi:10.1109/CVPR.2016.90.

Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J., Scholkopf, B., 1998. Support vector machines. IEEE Intelligent Systems and their applications 13, 18–28.

Hecht-Nielsen, R., 1988. Theory of the backpropagation neural network. Neural Networks 1, 445–448. URL: https://doi.org/10.1016/0893-6080(88)90469-8, doi:10.1016/0893-6080(88)90469-8.

Hickinbotham, S.J., Austin, J., 2000. Novelty detection in airframe strain data, in: 15th International Conference on Pattern Recognition, ICPR'00, Barcelona, Spain, September 3-8, 2000, IEEE Computer Society. pp. 2536–2539. URL: https://doi.org/10.1109/ICPR.2000.906130, doi:10.1109/ICPR.2000.906130.

Higham, N.J., 1988. Computing a nearest symmetric positive semidefinite matrix. Linear Algebra and its Applications 103, 103–118. doi:10.1016/0024-3795(88)90223-6.

Hinton, G.E., Osindero, S., Teh, Y.W., 2006. A fast learning algorithm for deep belief nets. Neural Computation 18, 1527–1554. URL: https://doi.org/10.1162/neco.2006.18.7.1527, doi:10.1162/neco.2006.18.7.1527.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Computation 9, 1735–1780. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735, doi:10.1162/neco.1997.9.8.1735.

Hryniowski, A., Wong, A., 2019. Deeplabnet: End-to-end learning of deep radial basis networks with fully learnable basis functions. CoRR abs/1911.09257. URL: http://arxiv.org/abs/1911.09257, arXiv:1911.09257.

Japkowicz, N., Shah, M., 2011. Evaluating learning algorithms: a classification perspective. Cambridge University Press.

Jin, Y., Sendhoff, B., 2003. Extracting interpretable fuzzy rules from RBF networks. Neural Process. Lett. 17, 149–164. URL: https://doi.org/10.1023/A:1023642126478, doi:10.1023/A:1023642126478.

Karczmarek, P., Kiersztyn, A., Pedrycz, W., Al, E., 2020. K-means-based isolation forest. Knowledge-Based Systems 195, 105659. URL: http://www.sciencedirect.com/science/article/pii/S0950705120301064, doi:https://doi.org/10.1016/j.knosys.2020.105659.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, IEEE. pp. 1942–1948.

Khan, S.S., Madden, M.G., 2013. One-class classification: Taxonomy of study and review of techniques. CoRR abs/1312.0049. URL: "http://arxiv.org/abs/1312.0049", arXiv:1312.0049.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.

LeCun, Y., Cortes, C., 2010. MNIST handwritten digit database URL: http://yann.lecun.com/exdb/mnist/.

Li, J., Mao, Z., Lu, Y., 2008. Adapting radial basis function neural networks for one-class classification, in: Proceedings of the International Joint Conference on Neural Networks, IJCNN 2008, part of the IEEE World Congress on Computational Intelligence, WCCI 2008, Hong Kong, China, June 1-6, 2008, IEEE. pp. 3766–3770. URL: https://doi.org/10.1109/IJCNN.2008.4634339, doi:10.1109/IJCNN.2008.4634339.

Li, N., Chang, F., 2019. Video anomaly detection and localization via multivariate gaussian fully convolution adversarial autoencoder. Neurocomputing 369, 92–105. URL: https://doi.org/10.1016/j.neucom.2019.08.044, doi:10.1016/j.neucom.2019.08.044.

Liu, T., Chen, S., Liang, S., Gan, S., Harris, C.J., 2020. Fast adaptive gradient RBF networks for online learning of nonstationary time series. IEEE Trans. Signal Process. 68, 2015–2030. URL: https://doi.org/10.1109/TSP.2020.2981197, doi:10.1109/TSP.2020.2981197.

Ma, R., Liu, Y., Lin, X., Wang, Z., 2008. Network anomaly detection using RBF neural network with hybrid QPSO, in: Proceedings of the IEEE International Conference on Networking, Sensing and Control, ICNSC 2008, Hainan, China, 6-8 April 2008, IEEE. pp. 1284–1287. URL: https://doi.org/10.1109/ICNSC.2008.4525415, doi:10.1109/ICNSC.2008.4525415.

Marchi, E., Vesperini, F., Eyben, F., Squartini, S., Schuller, B.W., 2015. A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural networks, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015, IEEE. pp. 1996–2000. URL: https://doi.org/10.1109/ICASSP.2015.7178320, doi:10.1109/ICASSP.2015.7178320.

Marir, N., Wang, H., Feng, G., Li, B., Jia, M., 2018. Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark. IEEE Access 6, 59657–59671.

Markou, M., Singh, S., 2003a. Novelty detection: a review - part 1: statistical approaches. Signal Process. 83, 2481–2497. URL: https://doi.org/10.1016/j.sigpro.2003.07.018, doi:10.1016/j.sigpro.2003.07.018.

Markou, M., Singh, S., 2003b. Novelty detection: a review - part 2: : neural network based approaches. Signal Process. 83, 2499–2521. URL: https://doi.org/10.1016/j.sigpro.2003.07.019, doi:10.1016/j.sigpro.2003.07.019.

Moody, G.B., Mark, R.G., 2001. The impact of the mit-bih arrhythmia database. IEEE Engineering in Medicine and Biology Magazine 20, 45–50. doi:10.1109/51.932724.

Moody, J.E., Darken, C.J., 1989. Fast learning in networks of locally-tuned processing units. Neural Comput. 1, 281–294. URL: https://doi.org/10.1162/neco.1989.1.2.281, doi:10.1162/neco.1989.1.2.281.

Moya, M.M., Koch, M.W., Hostetler, L.D., 1993. One-class classifier networks for target recognition applications. STIN 93, 24043.

Nguyen, A.M., Yosinski, J., Clune, J., 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, IEEE Computer Society. pp. 427–436. URL: https://doi.org/10.1109/CVPR.2015.7298640, doi:10.1109/CVPR.2015.7298640.

Oliveira, A.L.I., de Lima Neto, F.B., de Lemos Meira, S.R., 2004b. A method based on RBF-DDA neural networks for improving novelty detection in time series, in:

Barr, V., Markov, Z. (Eds.), Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA, AAAI Press. pp. 670–675. URL: http://www.aaai.org/Library/FLAIRS/2004/flairs04-115.php.

Oliveria, A., Neto, F., Meira, S., 2004a. Combining mlp and rbf neural networks for novelty detection in short time series, in: Lecture Notes in Computer Science (LNCS), Springer. pp. 844–853.

Orr, G.B., Müller, K. (Eds.), 1998. Neural Networks: Tricks of the Trade. volume 1524 of *Lecture Notes in Computer Science*. Springer. URL: https://doi.org/10.1007/3-540-49430-8, doi:10.1007/3-540-49430-8.

Pickands III, J., et al., 1975. Statistical inference using extreme order statistics. the Annals of Statistics 3, 119–131.

Rapaka, A., Novokhodko, A., Wunsch, D., 2003. Intrusion detection using radial basis function network on sequences of system calls, in: Proceedings of the International Joint Conference on Neural Networks, 2003., IEEE. pp. 1820–1825.

Reynolds, D.A., 2009. Gaussian mixture models, in: Li, S.Z., Jain, A.K. (Eds.), Encyclopedia of Biometrics. Springer US, pp. 659–663. URL: https://doi.org/10.1007/978-0-387-73003-5_196, doi:10.1007/978-0-387-73003-5_196.

Ribeiro, M., Lazzaretti, A.E., Lopes, H.S., 2018. A study of deep convolutional auto-encoders for anomaly detection in videos. Pattern Recognition Letters 105, 13–22. URL: https://doi.org/10.1016/j.patrec.2017.07.016, doi:10.1016/j.patrec.2017.07.016.

Rojas, R., 1996. Neural Networks - A Systematic Introduction. Springer. URL: http://page.mi.fu-berlin.de/%7Erojas/neural/.

Rosner, B., 1983. Percentage points for a generalized esd many-outlier procedure. Technometrics 25, 165–172. URL: http://www.jstor.org/stable/1268549.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S.A., Binder, A., Müller, E., Kloft, M., 2018. Deep one-class classification, in: Dy, J., Krause, A. (Eds.), Proceedings of the 35th International Conference on Machine Learning, PMLR. pp. 4393–4402. URL: http://proceedings.mlr.press/v80/ruff18a.html.

Šabić, E., Keeley, D., Henderson, B., Nannemann, S., 2020. Healthcare and anomaly detection: using machine learning to predict anomalies in heart rate data. AI & SOCIETY , 1–10.

Schlegl, T., Seeböck, P., Waldstein, S.M., Schmidt-Erfurth, U., Langs, G., 2017. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery, in: Niethammer, M., Styner, M., Aylward, S.R., Zhu, H., Oguz, I., Yap, P., Shen, D. (Eds.), Information Processing in Medical Imaging - 25th International Conference, IPMI 2017, Boone, NC, USA, June 25-30, 2017, Proceedings, Springer. pp. 146–157. URL: https://doi.org/10.1007/978-3-319-59050-9_12, doi:10.1007/978-3-319-59050-9_12.

Schölkopf, B., 2000. The kernel trick for distances, in: Leen, T.K., Dietterich, T.G., Tresp, V. (Eds.), Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA, MIT

Press. pp. 301–307. URL: https://proceedings.neurips.cc/paper/2000/hash/4e87337f366f72daa424dae11df0538c-Abstract.html.

Scholkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C., 2001. Estimating the support of a high-dimensional distribution. Neural Computation 13, 1443–1471.

Spence, C., Parra, L., Sajda, P., 2001. Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model, in: Proceedings IEEE workshop on mathematical methods in biomedical image analysis (MMBIA 2001), IEEE. pp. 3–10.

Susto, G.A., Beghi, A., McLoone, S., 2017. Anomaly detection through on-line isolation forest: An application to plasma etching, in: 2017 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC), IEEE. pp. 89–94.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R., 2014. Intriguing properties of neural networks, in: Bengio, Y., LeCun, Y. (Eds.), 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings. URL: http://arxiv.org/abs/1312.6199.

Tax, D.M.J., Duin, R.P.W., 2000. Data description in subspaces, in: Proceedings 15th International Conference on Pattern Recognition. ICPR-2000, pp. 672–675 vol.2. doi:10.1109/ICPR.2000.906164.

Tax, D.M.J., Duin, R.P.W., 2004. Support vector data description. Mach. Learn. 54, 45–66. URL: https://doi.org/10.1023/B:MACH.0000008084.60811.49, doi:10.1023/B:MACH.0000008084.60811.49.

Ting, K.M., Liu, F.T., Zhou, Z., 2008. Isolation forest, in: 2008 Eighth IEEE International Conference on Data Mining(ICDM), pp. 413–422. URL: doi.ieeecomputersociety.org/10.1109/ICDM.2008.17, doi:10.1109/ICDM.2008.17.

Tsamardinos, I., Rakhshani, A., Lagani, V., 2014. Performance-estimation properties of cross-validation-based protocols with simultaneous hyper-parameter optimization, in: Likas, A., Blekas, K., Kalles, D. (Eds.), Artificial Intelligence: Methods and Applications - 8th Hellenic Conference on AI, SETN 2014, Ioannina, Greece, May 15-17, 2014. Proceedings, Springer. pp. 1–14. URL: https://doi.org/10.1007/978-3-319-07064-3_1, doi:10.1007/978-3-319-07064-3_1.

Vidnerová, P., Neruda, R., 2018. Deep networks with RBF layers to prevent adversarial examples, in: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (Eds.), Artificial Intelligence and Soft Computing - 17th International Conference, ICAISC 2018, Zakopane, Poland, June 3-7, 2018, Proceedings, Part I, Springer. pp. 257–266. URL: https://doi.org/10.1007/978-3-319-91253-0_25, doi:10.1007/978-3-319-91253-0_25.

Wang, Y., Cai, W., Wei, P., 2016. A deep learning approach for detecting malicious javascript code. Security and Communication Networks 9, 1520–1534. URL: https://doi.org/10.1002/sec.1441, doi:10.1002/sec.1441.

Weiss, K., Khoshgoftaar, T.M., Wang, D., 2016. A survey of transfer learning. Journal of Big data 3, 9.

Widmer, G., Kubat, M., 1996. Learning in the presence of concept drift and hidden contexts. Machine Learning 23, 69–101. URL: https://doi.org/10.1007/BF00116900, doi:10.1007/BF00116900.

Xiao, H., Rasul, K., Vollgraf, R., 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR abs/1708.07747. URL: http://arxiv.org/abs/1708.07747, arXiv:1708.07747.

Xiong, Y., Zuo, R., 2016. Recognition of geochemical anomalies using a deep autoencoder network. Computers & Geosciences 86, 75 – 82. URL: http://www.sciencedirect.com/science/article/pii/S0098300415300728, doi:https://doi.org/10.1016/j.cageo.2015.10.006.

Xu, D., Yan, Y., Ricci, E., Sebe, N., 2017. Detecting anomalous events in videos by learning deep representations of appearance and motion. Computer Vision and Image Understanding 156, 117–127.

Xu, H., Chen, W., Zhao, N., Li, Z., Bu, J., Li, Z., Liu, Y., Zhao, Y., Pei, D., Feng, Y., Chen, J., Wang, Z., Qiao, H., 2018. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications, in: Champin, P., Gandon, F.L., Lalmas, M., Ipeirotis, P.G. (Eds.), Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018, ACM. pp. 187–196. URL: https://doi.org/10.1145/3178876.3185996, doi:10.1145/3178876.3185996.

Xu, S.S., Mak, M., Cheung, C., 2019. Towards end-to-end ECG classification with raw signal extraction and deep neural networks. IEEE J. Biomed. Health Informatics 23, 1574–1584. URL: https://doi.org/10.1109/JBHI.2018.2871510, doi:10.1109/JBHI.2018.2871510.

Yan, W., Yu, L., 2019. On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach. CoRR abs/1908.09238. URL: http://arxiv.org/abs/1908.09238, arXiv:1908.09238.

Zadeh, P.H., Hosseini, R., Sra, S., 2018. Deep-rbf networks revisited: Robust classification with rejection. CoRR abs/1812.03190. URL: http://arxiv.org/abs/1812.03190, arXiv:1812.03190.

Zhang, Q., Benveniste, A., 1992. Wavelet networks. IEEE transactions on Neural Networks 3, 889–898.

Zhou, C., Paffenroth, R.C., 2017. Anomaly detection with robust deep autoencoders, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA. pp. 665–674. URL: http://doi.acm.org/10.1145/3097983.3098052, doi:10.1145/3097983.3098052.

Zhou, S., Shen, W., Zeng, D., Fang, M., Wei, Y., Zhang, Z., 2016. Spatial-temporal convolutional neural networks for anomaly detection and localization in crowded scenes. Signal Process. Image Commun. 47, 358–368. URL: https://doi.org/10.1016/j.image.2016.06.007, doi:10.1016/j.image.2016.06.007.

Zidelmal, Z., Amirou, A., Adnane, M., Belouchrani, A., 2012. QRS detection based on wavelet coefficients. Comput. Methods Programs Biomed. 107, 490–496. URL:

https://doi.org/10.1016/j.cmpb.2011.12.004, doi:10.1016/j.cmpb.2011.12.004.

Zong, B., Song, Q., Min, M.R., Cheng, W., Lumezanu, C., Cho, D., Chen, H., 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net. URL: https://openreview.net/forum?id=BJJLHbb0-.