

bradscholars

Online Anomaly Detection for Time Series. Towards Incorporating Feature Extraction, Model Uncertainty and Concept Drift Adaptation for Improving Anomaly Detection

Item Type	Thesis
Authors	Tambuwal, Ahmad I.
Rights	<p>
The University of Bradford theses are licenced under a Creative Commons Licence.</p>
Download date	2026-05-18 03:44:28
Link to Item	https://bradscholars.brad.ac.uk/handle/10454/19806.2

ONLINE ANOMALY DETECTION FOR TIME SERIES

A.I. TAMB UWAL

PHD

2021

Online Anomaly Detection for Time Series

Towards Incorporating Feature Extraction, Model Uncertainty and Concept Drift Adaptation for Improving Anomaly Detection

Ahmad Idris TAMB UWAL

Submitted for the degree of

Doctor of Philosophy

Department of Computer Science
Faculty of Engineering and Informatics
University of Bradford

2021

Abstract

Ahmad Idris Tambuwal

Online Anomaly Detection for Time Series

Towards Incorporating Feature Extraction, Model Uncertainty, and Concept Drift Adaptation for Improving Anomaly Detection

Keywords: Time Series, Online Anomaly Detection, Concept Drift, Prediction Interval, Deep Neural Networks, Uncertainty in Deep Learning, Quantile Regression.

Time series anomaly detection receives increasing research interest given the growing number of data-rich application domains. Recent additions to anomaly detection methods in research literature include deep learning algorithms. The nature and performance of these algorithms in sequence analysis enable them to learn hierarchical discriminating features and time-series temporal nature. However, their performance is affected by the speed at which the time series arrives, the use of a fixed threshold, and the assumption of Gaussian distribution on the prediction error to identify anomalous values. An exact parametric distribution is often not directly relevant in many applications and it's often difficult to select an appropriate threshold that will differentiate anomalies with noise. Thus, implementations need the Prediction Interval (PI) that quantifies the level of uncertainty associated with the Deep Neural Network (DNN) point forecasts, which helps in making a better-informed decision and mitigates against false anomaly alerts. To achieve this, a new anomaly detection method is proposed that computes the uncertainty in estimates using quantile regression and used the quantile interval to identify anomalies. Similarly, to handle the speed at which the data arrives, an online anomaly detection method is proposed where a model is trained incrementally to adapt to the concept drift that improves prediction. This is implemented using a window-based strategy, in which a time series is broken into sliding windows of sub-sequences as input to the model. To adapt to concept drift, the model is updated when changes occur in the new arrival instances. This is achieved by using anomaly likelihood which is computed using the Q-function to define the abnormal degree of the current data point based

on the previous data points. Specifically, when concept drift occurs, the proposed method will mark the current data point as anomalous. However, when the abnormal behavior continues for a longer period of time, the abnormal degree of the current data point will be low compared to the previous data points using the likelihood. As such, the current data point is added to the previous data to retrain the model which will allow the model to learn the new characteristics of the data and hence adapt to the concept changes thereby redefining the abnormal behavior. The proposed method also incorporates feature extraction to capture structural patterns in the time series. This is especially significant for multivariate time-series data, for which there is a need to capture the complex temporal dependencies that may exist between the variables. In summary, this thesis contributes to the theory, design, and development of algorithms and models for the detection of anomalies in both static and evolving time series data.

Several experiments were conducted, and the results obtained indicate the significance of this research on offline and online anomaly detection in both static and evolving time-series data. In chapter 3, the newly proposed method (Deep Quantile Regression Anomaly Detection Method) is evaluated and compared with six other prediction-based anomaly detection methods that assume a normal distribution of prediction or reconstruction error for the identification of anomalies. Results in the first part of the experiment indicate that DQR-AD obtained relatively better precision than all other methods which demonstrates the capability of the method in detecting a higher number of anomalous points with low false positive rates. Also, the results show that DQR-AD is approximately 2 – 3 times better than the DeepAnT which performs better than all the remaining methods on all domains in the NAB dataset. In the second part of the experiment, sMAP dataset is used with 4-dimensional features to demonstrate the method on multivariate time-series data. Experimental result shows DQR-AD have 10% better performance than AE on three datasets (SMAP1, SMAP3, and SMAP5) and equal performance on the remaining two datasets. In chapter 5, two levels of experiments were conducted basis of false-positive rate and concept drift adaptation. In the first level of the experiment, the result shows that online DQR-AD is 18% better than both DQR-AD and VAE-LSTM on five NAB datasets. Similarly, results in the second level of the experiment show that the online DQR-AD method has better performance than five counterpart methods with a relatively 10% margin on six out of the seven NAB datasets. This result demonstrates how concept drift adaptation strategies adopted in the proposed online DQR-AD improve the performance of anomaly detection in

time series.

Declaration

The candidate confirmed that the research work presented in this thesis is his own and has not been submitted for the award of any other degree. Wherever information from other sources have been utilized, an appropriate credit and reference has been made to acknowledge the authors efforts.

Ahmad Idris Tambuwal

Acknowledgements

All praise be to Allah, the Lord of the Universe, the most gracious, and the most merciful. May His peace and blessings go to His noble Messenger, Prophet Muhammad S.A.W. I couldn't have reached this far without Allah's blessings and help. I also want to show my appreciation to many people whom without I couldn't have the feet to stand and cope with the great challenges during my Ph.D. period. First, I want to start with my parents who are my bedrock and had made huge sacrifices for my success. I wouldn't be in a position of enrolling and completing a Ph.D. without their support. I also want to appreciate my two wives and four kids who have always provided a welcome distraction that motivates and gives me the strength to carry on during difficult times. To the rest of my family and friends, I will say thank you for your support, advice, and well wishes.

Secondly, I wish to express my profound gratitude to my principal supervisor Professor Daniel Neagu and my associate supervisor Professor Marian Gheorghe for their endless support throughout my Ph.D. journey. I couldn't have endured completing my Ph.D. without their support and understanding during good and challenging periods. Special thanks to the panel of examiners; Professor Keshav Dahal, Dr. ARA Abdullatif, and Dr. D Bryant for their comments and feedback which help a lot in enhancing the quality of the thesis.

I wish to thank Petroleum Technology Development Fund (PTDF) for given the required scholarship and grants to carry out my Ph.D. research. I am also indebted to the Usmanu Danfodiyo University Sokoto for giving me study leave that support me throughout my studies. I also wish to thank the staff and PGR students of the Department of Computer Science and the entire Faculty of Engineering and Informatics.

Publications and Presentations

Journal Papers

- A. I. Tambuwal and D. Neagu, “Deep quantile regression for unsupervised anomaly detection in time series,” *SN Computer Science*, vol. 2, no. 6, pp. 1–16, 2021.

Conference proceedings

- A. I. Tambuwal, D. Neagu, and M. Gheorghe, “An experimental comparison of ensemble classifiers for evolving data streams,” in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 2017, pp. 156–162.
- A. I. Tambuwal and A. M. Bello, “Deepconad: Deep and confidence prediction for unsupervised anomaly detection in time series,” in *Science and Information Conference*. Springer, 2020, pp. 232–244.

Presentations and posters

- A comparative study on ensemble classifiers for evolving data streams. Presented at 1st Annual Innovative Engineering Research Conference (AIERC 2017) in Faculty of Engineering and Informatics, University of Bradford.
- Deep and Confidence Prediction for Unsupervised Anomaly Detection in time series. Presented at 3rd Annual Innovative Engineering Research Conference (AIERC 2019) in Faculty of Engineering and Informatics, University of Bradford.
- Concept Drift Analysis for Improving Anomaly Detection in Sensor Data Streams. A seminar presented at 2017 PGR seminar series in Faculty of Engineering and Informatics, University of Bradford.

List of Abbreviations

- AI** Artificial Intelligence
- ANN** Artificial Neural Network
- AR** AutoRegression
- AD** Anomaly Detection
- ADAM** Anomaly Detection And Mitigation
- ARMA** AutoRegression Moving Average
- ARIMA** AutoRegression Integrated Moving Average
- AUROC** Area Under the Receiver Operating Characteristic
- AUC** Area Under the ROC Curve
- BPTT** Back Propagation Through Time
- CNN** Convolutional Neural Network
- CVFDT** Concept-adaptive Very Fast Decision Tree
- CD** Critical Difference
- DNN** Deep Neural Network
- DQR-AD** Deep Quantile Regression Anomaly Detection
- DTW** Dynamic Time Warping
- DWT** Discrete Wavelet Transform
- DEAR** Deep Evidential Action Recognition
- DBMS** Database Management System

ECU Electronic Control Unit
ECG Electrocardiogram
ELM Extreme Learning Machine
GRU Gated Recurrent Units
GMM Gaussian Mixture Model
GP Gaussian Process
HMM Hidden Markov Model
 h_w History Window
HTM Hierarchical Temporal Memory
IoT Internet of Things
IOLIN Incremental On-Line Information Network
k-NN k Nearest Neighbour
KLD Kullback Le
LSTM Long Short Term Memory
LQ Lower Quantile
MA Moving Average
MTS Multivariate Time Series
MOA Massive Online Analysis
NN Neural Network
NAB Numenta Anomaly Benchmark
NSHW Non Seasonal Holt Winters
OC-SVM One Class Support Vector Machine
OCPC One Class Principal Component Classification
PI Prediction Interval
PTDF Petroleum Technology Development Fund

PCA Principal Component Analysis
PLS Partial Least Square
 p_w Prediction Window
QL Quantile Loss
QI Quantile Interval
RBF Radial Basic Function
RNN Recuurent Neural Network
SAX Symbolic Aggregate Approximation
SVM Support Vector Machine
SGD Stochastic Gradient Descent
SMAP Soil Moisture Active Passive
TEDA Typicality and Eccentricity Data Analytics
UQ Upper Quantile
UCR University of California
VAMSE Validational Mean Sequare Error
VFDT Very Fast Decision Tree

Contents

Abstract	i
Declaration	iv
Acknowledgements	v
Publications	vi
List of Abbreviations	vii
Table of Contents	x
List of Figures	xiii
List of Tables	xv
1 Introduction	1
1.1 Research Motivation	3
1.2 Thesis Statement and Research Questions	4
1.3 Methodology	5
1.4 Thesis Contributions	6
1.5 Thesis Outlines	7
2 Theoretical Background on Time series Anomaly Detection	8
2.1 Introduction	8
2.2 Theoretical Background	8
2.2.1 Data Stream and time series	8
2.2.2 Concept Change	9
2.2.2.1 Concept Shift	9

2.2.2.2	Concept Drift	10
2.2.3	Concept Drift Adaptation	10
2.2.4	Uncertainty Estimation	11
2.2.5	Quantile Regression	11
2.2.6	Anomaly	12
2.2.6.1	Point Anomaly	12
2.2.6.2	Contextual Anomalies	12
2.2.6.3	Collective Anomalies	13
2.3	Problem Concept for Anomaly Detection	13
2.3.1	Nature of Input Data	13
2.3.2	Types of Anomaly	14
2.3.3	Data Labels	14
2.3.3.1	Supervised Anomaly Detection	15
2.3.3.2	Semi-supervised Anomaly Detection	15
2.3.3.3	Unsupervised Anomaly Detection	15
2.3.4	Output of Anomaly Detection	16
2.3.4.1	Anomaly Scores	16
2.3.4.2	Class Labels	16
2.4	Anomaly Detection Techniques for time series	16
2.4.1	Detecting Point Anomalies	17
2.4.2	Detecting Anomalous Sub-sequence	17
2.4.3	Detecting Contextual Anomalies	17
2.4.4	Detecting Anomalous time series in time series Database	18
2.5	Review of time series Anomaly Detection Techniques	18
2.5.1	Distance Based Techniques	18
2.5.1.1	Window Based	19
2.5.1.2	Proximity Based	20
2.5.1.3	Transformation Based	21
2.5.1.4	Statistical Distance-based	21
2.5.2	Prediction Based Techniques	21
2.5.2.1	Classical Regression Based Models	22
2.5.2.2	Machine Learning Based	24
2.5.3	Deep Learning Based	25
2.5.3.1	Recurrent Neural Network (RNN)	26
2.5.3.2	Gated Recurrent Units (GRU)	28
2.5.3.3	Long Short Term Memory (LSTM)	28
2.5.3.4	Autoencoder	30
2.5.3.5	Deep Learning-based Anomaly Detection Techniques	30
2.6	Anomaly Detection for Stream Data	35
2.7	Summary	38

3	Comparison of Ensemble Classifiers on Drifting Data Streams	40
3.1	Introduction	40
3.2	Literature Review	41
3.2.1	Online Learning	41
3.2.2	Concept Drift Detection	42
3.3	Datasets Description	42
3.4	Experiments	43
3.4.1	Experimental Settings	43
3.4.2	Experimental Results	44
3.5	Summary	46
 4	 Unsupervised Anomaly Detection Method for Multivariate time series	 49
4.1	Introduction	49
4.2	Literature Review	50
4.3	The Proposed DQR-AD Model	52
4.3.1	time series Segmentation	52
4.3.2	time series Prediction	54
4.3.2.1	LSTM Network Architecture	55
4.3.2.2	Quantile Loss	57
4.3.3	Anomaly Detection and Classification	58
4.4	Dataset Description	59
4.4.1	NAB (Numenta Anomaly Benchmark) Dataset	59
4.4.2	ECG Dataset	60
4.4.3	sMAP Dataset	61
4.5	Experiments	61
4.5.1	Part I Experiment: NAB Dataset	61
4.5.1.1	Experimental Setups	61
4.5.1.2	Training and Test Data Construction	62
4.5.1.3	Model Training and Validation	62
4.5.1.4	Model Testing and Prediction	64
4.5.1.5	Anomaly Detection and Classification	64
4.5.1.6	Experimental Results and Discussions	64
4.5.2	Part II Experiment: ECG Dataset	68
4.5.2.1	Experimental Setup	68
4.5.2.2	Experimental Results and Discussion	69
4.5.3	Part III Experiment: sMAP Dataset	69
4.5.3.1	Experimental Setup	69
4.5.3.2	Experimental Results and Discussion	71
4.6	Summary	71

5	Online Anomaly Detection Method for Multivariate Sensor Streams	75
5.1	Introduction	75
5.2	Literature Review	76
5.3	Proposed Approach	80
5.3.1	Autoencoding	81
5.3.2	Deep Quantile Regression (DQR)	81
5.3.3	Online Anomaly Detection	83
5.3.4	Concept Drift Adaptation	83
5.4	Dataset Description	85
5.4.1	Yahoo Dataset:	85
5.4.2	NAB Dataset:	85
5.5	Experiments	88
5.5.1	Model Training	89
5.5.2	Anomaly Detection	91
5.5.3	Experimental Results and Discussions	91
5.6	Summary	95
6	Conclusions	97
	References	102
	Appendix A Results of the Experiment conducted using UCR Time Series Anomaly Benchmark Datasets.	118

List of Figures

2.1	Recurrent Neural Network Architecture	27
2.2	LSTM Architecture	29
2.3	Autoencoder Architecture	31
3.1	Prequential AUC for Concept change datasets	45
3.2	Prequential AUC for Imbalance datasets	46
3.3	Prequential AUC for real datasets	47
3.4	AUC comparison of ensemble classifiers with Nemenyi test	47
4.1	Schematic representation of DQR-AD method that involves time series segmentation, prediction, and anomaly detection	53
4.2	DQR model network architecture for time series prediction	57
4.3	A Sample of Normal time series (blue line) with anomalous point (vertical red dashed line) from Machine Temperature Dataset	60
4.4	A Sample of Normal time series (blue line) with anomalous point (vertical red dashed line) from Ambient Temperature Dataset	60
4.5	Autocorrelation of 10 weeks depth for nyc_taxi time series data	63
4.6	DQR Model training and validation loss for nyc_taxi time series data	63
4.7	A trade-off between precision and recall used for setting up a threshold value for nyc_taxi time series data	65
4.8	For nyc_taxi dataset, actual time series values (red) plotted against the QI (blue) to show periods of uncertainties (circled in green) in the test set	66
4.9	Using a threshold value of 17000 for anomaly detection in nyc_taxi time series data	66
4.10	Confusion Matrix for DQR-AD Method	67
4.11	Confusion Matrix for LSTM-AD Method	67
4.12	Confusion Matrix for DQR-AD Method on MBA_ECG14046_data_8	70

LIST OF FIGURES

4.13 Confusion Matrix for LSTM-AD Method on MBA_ECG14046_data_8	70
5.1 Schematic representation of proposed online DQR-AD method	81
5.2 Neural network Architecture, with autoencoding phase using LSTM encoder-decoder, followed by DQR model whose input is reconstructed sequence from the autoencoding	82
5.3 A1Benchmark time series Sample	86
5.4 A2Benchmark time series Sample	86
5.5 A3Benchmark time series Sample	87
5.6 A4Benchmark time series Sample	87
5.7 NYC_Taxi Sample time series with Anomalies	88
5.8 EC2 Request Sample time series with Anomalies	88
5.9 Training and validation loss	89
5.10 Sample of actual sequence from nyc_taxi dataset	90
5.11 Sample of reconstructed sequence from nyc_taxi dataset	90
5.12 Predicted test set sample from nyc_taxi data. Actual values are shown in blue with prediction interval shown in orange	91
5.13 MAE and Relative Uncertainty for A1Benchmark	93
5.14 MAE and Relative Uncertainty for A2Benchmark	93
5.15 MAE and Relative Uncertainty for A3Benchmark	94
5.16 MAE and Relative Uncertainty for A4Benchmark	95
A.1 Detected vs Actual Anomalies on apneaecg1 time series	119
A.2 Detected vs Actual Anomalies on apneaecg2 time series	119
A.3 Detected vs Actual Anomalies on apneaecg3 time series	120
A.4 Detected vs Actual Anomalies on ECG1 time series	120
A.5 Detected vs Actual Anomalies on ECG2 time series	121
A.6 Detected vs Actual Anomalies on ECG3 time series	121
A.7 Detected vs Actual Anomalies on ECG4 time series	122
A.8 Detected vs Actual Anomalies on ECG5 time series	122
A.9 Detected vs Actual Anomalies on EPG1 time series	123
A.10 Detected vs Actual Anomalies on EPG3 time series	123
A.11 Detected vs Actual Anomalies on EPG5 time series	124
A.12 Detected vs Actual Anomalies on Gaithunt time series	124
A.13 Detected vs Actual Anomalies on MARS1 time series	125
A.14 Detected vs Actual Anomalies on MARS5 time series	125
A.15 Detected vs Actual Anomalies on Power Demand2 time series	126
A.16 Detected vs Actual Anomalies on Power Demand3 time series	126
A.17 Detected vs Actual Anomalies on Power Demand4 time series	127
A.18 Detected vs Actual Anomalies on Temperature time series	127

List of Tables

3.1	Datasets and their characteristics	43
3.2	Ensemble Classifiers for non-stationary data streams	44
3.3	Average Prequential AUC	44
4.1	Comparative evaluation of DQR-AD with four anomaly detection methods (NumentaTM, ContextOSE, DeepAnT, and VAE-LSTM) on 20 NAB time series from different domains. Precision and Recall is reported in this table	68
4.2	Comparative evaluation of DQR-AD with five other anomaly detection methods (ContextOSE, EXPoSE, NumentaTM, DeepAnT, and VAE-LSTM) applied to entire NAB dataset using mean F_score for each domain	68
4.3	Comparative evaluation of DQR-AD with two anomaly detection methods (LSTM-AD and Deep LSTM-AD) on 47 ECG time series. F_Score, Precision and Recall is reported in this table	73
4.4	Comparative evaluation of DQR-AD and AE ($h = 2, h = 10$) on Five sMAP Datasets (SMAP1 to SMAP5). An average AUROC is reported on each dataset.	74
5.1	MAE and its Relative Confidence	92
5.2	Comparative evaluation of Online DQR-AD with two other anomaly detection methods (DQR-AD and VAE-LSTM) on 7 Real Known Cause Anomaly time series. F-Score, Precision and Recall is reported in this table	94
5.3	Comparative evaluation of Online DQR-AD with five other online anomaly detection methods (NumentaTM, Online RNN-AD, Data-driven-AD, AE-AD, and SGP-Q) applied to Real Known Cause anomaly dataset using AUC	96

Introduction

Rapid advancement in industry 4.0 technologies (such as IoT, cloud computing, AI, and machine learning) have made it common for many application domains such as health care, food Supply chain, transportation, etc., to install a large number of sensors on many devices [39]. Sensor data collected in such domains is in the form of time series or streams that demonstrates temporal characteristics. In an automotive system, for example, data collected from the ECU of the car is in the form of time series which come from various sensor channels. However, a fault in those sensors will lead to anomalous points in the time series collected. In addition, the health sector can use time series signals from ECG recordings to identify an abnormal heart condition for a particular patient.

As such, industries often collect and exploit such readings for anomaly or fault detection. This indicates the need for anomaly detection methods for early fault detection, with potential contributions to avoid total system failure. This includes providing early evidence for the detection of mechanical faults [152] and sensor faults [130] in automotive vehicles during usage.

Anomaly detection often defined as the process of identifying an abnormal pattern in the data, is a popular research problem in the literature, that is widely explored in different application domains [32, 105, 113, 152]. In such domains, anomalies correspond to critical events that are not observed in the data coming from normal system behavior. time series anomaly detection problems have a wider and different scope due to the nature of the input data and how anomaly is defined. Input data can be in the form of univariate, in which each observation is a univariate symbol or real value, or multivariate, in which each observation is a multivariate vector consisting of a symbolic, continuous, or heterogeneous mixture of univariate observations. Each format of the input data possesses its own unique characteristics that need to be handled in a unique manner. In time series anomaly can be defined in different ways depending on the application domain [29]. In this thesis, an anomaly is defined as an unexpected

point in time series (e.g. a sudden sensor drift) while a continuous change in the sensor readings refers to as a concept change. Anomaly detection methods are needed for early fault detection, with potential contributions to avoid total system failure. This includes providing early evidence for the detection of mechanical faults [152] and sensor faults [130] in many application domains.

Several traditional anomaly detection methods exist in literature which assumed the data to be univariate or multivariate records without considering the sequential nature of the time series data. These methods are generally categorized based on their output [29] into labeling and scoring techniques. The labeling techniques, also called classification-based methods, directly assign class labels (e.g. normal or abnormal) to the test instances using a trained model [51, 136, 142]. For supervised-learning purposes, such techniques rely on pre-labeled data sets. However, labeling large volumes of data is often difficult, costly, and sometimes incomplete due to big data resource challenges. In contrast, the advantage brought by scoring techniques that assign anomaly scores based on the degree the data instance shows anomaly profile is similar to applications using unsupervised or semi-supervised learning. These techniques include statistical-based [74, 76] and distance-based methods [5, 8, 23, 77, 118, 169]. Statistical distance-based methods [70, 161] use the distribution of the data, but could be difficult to use in some applications [77]. Distance-based anomaly detection techniques compute anomaly scores by calculating the distance between sample points thereby assuming that normal data points exist in the same area and are far apart from the abnormal data points. Moreover, due to the temporal nature of time series, distance-based anomaly detection methods are difficult to be used due to their sensitivity to noise and distance calculation function. In addition, they are not suitable for cases without sufficient relevant data, that is relative to the task at hand.

On the other hand, classical regression models that comprises MA [31], AR [31, 52], ARMA [116], ARIMA [107], Kalman filters [88], and general regression [49, 127] are used for prediction-based anomaly detection methods. Neither of these methods uses sequential models that exploit the temporal nature of time series data which makes their predictions less accurate. Often, anomalies in time series data can only be identified by analyzing data instances in sequential order which is not supported by traditional anomaly detection methods. Neither of these methods uses sequential models that exploit the temporal nature of time series data which makes their predictions less accurate.

Recent advancement in deep learning methods applications to big data collections opens also opportunities to study their applicability to anomaly detection [28]. These methods used sequential models, and their performance in sequence analysis reported in multimedia applications [35, 64, 65, 154] enable them to learn the hierarchical discriminating features and time series temporal nature [67]. In addition, for each time series point, an anomaly score is calculated which helps in handling the type of anomaly detection problem. Deep learning-based anomaly detection techniques using

LSTM [4, 97, 103] and other forms RNN [85, 131], CNN [110], and auto-encoder [5, 96, 99, 102, 126, 134, 173] demonstrate higher performance over the previously mentioned prediction-base anomaly detection techniques. Despite their performance, they are affected by the speed at which the time series arrives, the use of a fixed threshold, and the assumption of Gaussian distribution on the prediction error to identify anomalous values. However, an exact parametric distribution is often not directly relevant in many applications and it's often difficult to select an appropriate threshold that will differentiate anomalies with noise. Thus, implementations need the PI that quantifies the level of uncertainty associated with the DNN point forecasts, which helps in making a better-informed decision and mitigates against false anomaly alerts. To achieve this, a new anomaly detection method is proposed that computes uncertainty in estimates using quantile regression and used the quantile interval to identify anomalies. Similarly, to handle the speed at which the data arrives, an online anomaly detection method is proposed where a model is trained incrementally to adapt to the concept drift that improves prediction. This is implemented using a window-based strategy, in which a time series is broken into sliding windows of sub-sequences as input to the model. To adapt to concept drift, the model is updated when changes occur in the new arrival instances. This is achieved by using anomaly likelihood which is computed using the Q-function to define the abnormal degree of the current data point based on the previous data points. Specifically, when concept drift occurs, the proposed method will mark the current data point as anomalous. However, when the abnormal behavior continues for a longer period, the abnormal degree of the current data point will be low compared to the previous data points using the likelihood. As such, the current data point is added to the previous data to retrain the model which will allow the model to learn the new characteristics of the data and hence adapt to the concept changes thereby redefining the abnormal behavior.

1.1 Research Motivation

This research is funded by PTDF in Nigeria. PTDF believes that Nigeria's oil and gas industries can be enhanced through human and institutional capacity development and the promotion of research and the acquisition of relevant technologies. This mandate enables the agency to provide scholarships that will train Nigerian graduates within the country and abroad for relevant areas in the oils and gas industry. As a beneficiary of this scholarship, one of my fundamental motivations for conducting this research is attaining this mandate.

The enormous amount of data generated from sensors due to fast advances in industry 4.0 technologies and the increasing need to analyze this data for a number of industrial applications were also great motivations. The generated data are mostly time-ordered measurements processable as time series and exploring such data may

1.2 Thesis Statement and Research Questions

lead to the discovery of a number of industrial applications. In the automotive industry, for example, outlier detection in sensor data will be needed for early fault detection with the potential contribution to avoiding total system failure. This includes providing early evidence for the detection of mechanical faults [152] and sensor faults [130] in automotive vehicles during usage.

Again, the performance of deep learning algorithms and their recent applications in big data collections open also opportunities to study their applicability to anomaly detection [28]. These methods used sequential models, and their performance in sequence analysis reported in multimedia applications [35, 64, 65, 154], enable them to learn the hierarchical discriminatory features and time series temporal nature [67]. In addition, for each time series point, an anomaly score is calculated which helps in handling the type of anomaly detection problem. We then posit that uncertainty in estimates from deep learning algorithms can provide valuable information that can enhance the process of anomaly detection. The target is to develop a model that forecasts the next time series point taking uncertainty into account. One of the possible ways to achieve this is through quantile regression. This motivated us to develop a deep quantile regression method for anomaly detection in multivariate time series.

Finally, one of the key features of time series data collected from sensors is its dynamic nature. Such time series are observed to have non-stationary behavior due to periodic changes. Therefore, the common approach of using an offline method of detecting anomalies will fail when dealing with the dynamic time series. This is the fundamental motivation for further studies to develop an online anomaly detection method. This is achieved by updating the deep quantile regression anomaly detection method and the proposed online method that handled the dynamic nature of time series. This is achieved by incorporating concept drift adaptation in the anomaly detection process. These are the major motivations for this study.

1.2 Thesis Statement and Research Questions

This thesis aims at advancing the state of the art in time series anomaly detection problems. The thesis studies the problem of both static and dynamic time series with a focus on detecting if a given point in the test data is anomalous with respect to training data of both normal and abnormal sequences. Specifically, our research work focuses on unsupervised prediction-based anomaly detection for both static and dynamic time series.

However, most of the prediction-based anomaly detection methods that used an unsupervised learning approach compute anomaly scores by assuming a distribution usually of Gaussian type on the prediction error, which is either ranked or threshold to label data instances as anomalous or not. But, an exact parametric distribution is often not directly relevant in some applications, and the assumption of any distribution will

lead to false anomaly alerts due to high prediction variance. This problem then leads to the novel research question (RQ1) as follows:

- **Research Question 1 (RQ1):** will the estimation of uncertainty in prediction using quantile regression increase the prediction performance and reduced false anomaly alerts?

Similarly, most anomaly detection methods used fixed thresholds to identify anomalous values. However, it is difficult to select an appropriate threshold that will differentiate anomalies with noise. As such, choosing a single threshold will result in an imbalance of true and false positive rates. To address this problem, research question 2 (RQ2) is formulated as follows:

- **Research Question 2 (RQ2):** will obtain probabilistic threshold using confidence interval improve anomaly detection performances and reduces false positive rate?

Moreover, data characteristics remain an important factor in developing an anomaly detection method. In the big data context, the volume and speed at which the time series data arrive demonstrate the need for feature extraction to capture structural regularities in time series and adaptation of concept changes in data. This will improve the quality of predictions and as such, an interesting research question (RQ3) is formulated as follows:

- **Research Question 3 (RQ3):** will feature extraction and concept drift adaptation improve the performance of online anomaly detection and reduce the rate of false positive alerts?

1.3 Methodology

In this thesis, quantitative research methodology is adopted as described in [69]. Specifically, the methodology involves a literature review of existing methods to identify the problem, analysis of the problem, model design, algorithms development, experiments, and testing to evaluate the model and algorithms developed. In each chapter, a literature review is carried out to find out the main challenges and make use of recent research developments in a specific area of contribution. Algorithms and models are then developed which are implemented in sequence, starting from simple to more complicated tasks. The algorithms and models are evaluated using both real and synthetic datasets to ascertain their efficacy. Throughout this research, publicly available sensor data which comprises time series and data streams will be processed. We utilized both statistical and deep learning techniques for anomaly detection in time series data. This is achieved using the following steps:

1. *Identify different definitions and problem formulations of anomalies in time series data.* There are often different definitions of anomalies and different formulations of anomaly detection problems depending on the application domain. We identify these definitions and study different aspects of anomaly detection problems in relation to the nature of the time series with a specific focus on multivariate time series.
2. *An experimental comparison of ensemble classifiers on drifting data streams.* As a result of a preliminary study conducted to investigate the challenges faced by online learning algorithms when there is concept drift in the data. It was discovered that online ensemble methods are the most popular approach used in handling concept drift due to their stability-elasticity property that makes it easy for them to incorporate new data into the model. As such, an experimental comparison of online ensemble classifiers was conducted using drifting data streams.
3. *Proposed a new anomaly detection method called DQR-AD for time series.* To address the challenges in the assumption of Gaussian distribution for identifying anomalies, a new anomaly detection method is proposed that computes the uncertainty in estimates using quantile regression and used the quantile interval to identify anomalies.
4. *Proposed an online anomaly detection method for dynamic time series.* One key characteristic of time series collected from sensors is its dynamic structure. Such time series are observed to have non-stationary behavior due to periodic changes. Therefore, the common approach of using an offline method of detecting anomalies will fail when dealing with the above-characterized time series. As such, we update our anomaly detection method and proposed an online method that handled anomaly detection for dynamic time series in real-time. This is implemented using a window-based strategy, in which a time series is broken into sliding windows of sub-sequences. Each window is treated as a unit of analysis. We also incorporate feature extraction to capture structural patterns in the time series. This is especially significant for multivariate time series data, for which there is a need to capture the complex temporal dependencies that may exist between the variables.

1.4 Thesis Contributions

The thesis contributions to the field of computer science and anomaly detection include:

- Identification of different definitions and problem formulations of anomalies in time series data.
- An experimental comparison of ensemble classifiers on drifting data streams.
- Development of a new deep quantile regression model for anomaly detection in time series.
- Development of an online anomaly detection method that incorporates feature extraction and concept drift adaptation to improve anomaly detection in data streams.

1.5 Thesis Outlines

This thesis is structured into six independent chapters with each chapter covering a specific topic of the thesis. This chapter introduces the research problem and motivations, research questions, methodology, contributions, and structure of the thesis.

- **Chapter 2. Theoretical Background on time series Anomaly Detection:** This chapter provides the theoretical background of the related concepts, the problem concept for anomaly detection, and a review of anomaly detection techniques for time series and data streams.
- **Chapter 3. Comparison of Ensemble Classifiers on Drifting Data Streams:** This chapter provides an experimental analysis of concept drift and its corresponding challenges on online learning algorithms. It is targeted at experimental comparison and performance evaluation of online ensemble classifiers using both real and synthetic data streams.
- **Chapter 4. Unsupervised Anomaly Detection Method for Multivariate time series:** Proposes a new unsupervised anomaly detection method called DQR-AD for multivariate time series.
- **Chapter 5. Online Anomaly Detection for time series:** This chapter proposes a new online anomaly detection method that investigates how concept drift adaptation improves the performance of anomaly detection in dynamic time series.
- **Chapter 6. Conclusions:** Contains the summary of previous chapters and the directions for the future works.

Theoretical Background on Time series Anomaly Detection

2.1 Introduction

An anomaly or outlier refers to an unexpected pattern that deviates from the normal pattern in time series data. Outlier and anomaly are two interchangeable words that are frequently used in the field of computer science [29, 68]. The process of identifying anomalies or outliers is referred to as anomaly detection. Anomaly detection has been extensively used in many areas of applications such as health care, health monitoring systems, activity monitoring, detection of fraudulent activities, and fault detection. This chapter provides a background study for anomaly detection in both static and dynamic time series data. It starts with the theoretical background on time series, data streams, concept drift, and quantile regression. This is then followed by different concepts of the anomaly detection problem and a review of time series anomaly detection techniques.

2.2 Theoretical Background

2.2.1 Data Stream and time series

A dynamic time series or data stream is defined as a sequence of digital signals that are used to transmit or receive information continuously. It is a temporarily ordered, and potentially infinite sequence of instances that arrives at a high speed that does not warrant storage in the memory [63]. A time series is an ordered series of observations that involves a sequence of values obtained over repeated measurements of time. For example; $x_i(t); [i = 1, \dots, n; t = 1, \dots, m]$, is a time series made sequentially

through a time where i indexes the measurements made at each time point t [166]. It is called a univariate time series when $n = 1$, and multivariate time series (MTS) when $n \geq 2$. Multivariate time series datasets have become popular in many real-world applications such as weather predictions and data analysis, multimedia, an automotive system, monitoring of the industrial process, medical, and finance where many sensors are used that generate values at a given time interval.

Similarities and Differences

1. A time series can be formed from a data stream where order and time are the fundamental concepts that define its features. While data stream can be received over time, it does not mean it can be analyzed as time series because the time a data is received may not represent the time of measurements which is the characteristic of time series.
2. In data analytics, we refer to time series data as past observations that are already stored and the order of the data is the focus of its analysis which differs from data streams that required real-time collection and processing.
3. In order to analyze and discover patterns from data streams, there is a need to develop an online model with incremental learning behavior. Although data streams can have a time series format, such data will be of high volume mounting to terabytes or petabytes. As a result single-scan, online data analysis methodology should also be critically important for such data.

2.2.2 Concept Change

A data stream is a dataset that comprises observations with time stamps that indicates the total or partial order between those observations. To define a concept change, there is a need to first understand what the concept is all about. Consider the process that generates streams to be a joint distribution over a random variable Y and $X = X_1, \dots, X_n$, where $y \in Y$ is the class labels and $x \in X$ represent the data values. The concept is defined in different ways, but a more probabilistic definition is given in [162, 163], where it is defined as a joint probability distribution between the random variables X and Y . This can be express in the following form $P_t(X, Y) = P_t(Y)P_t(X|Y) = P_t(X)P_t(Y|X)$. For two time stamps t and $t + 1$, a concept is stable when all instances are generated with the same probability distribution that is $P_t(X, Y) = P_{t+1}(X, Y)$ while a concept change when $P_t(X, Y) \neq P_{t+1}(X, Y)$.

2.2.2.1 Concept Shift

This involves changes in the input data X or changes in the distribution of the data which occurs when a model is trained using data drawn from a particular distribution

is applied to the data from a different distribution i.e. $P_t(X) \neq P_{t+1}(X)$. For example, an increase in anomalous activity resulted in changes in the distribution of the data used in training the model.

2.2.2.2 Concept Drift

Concept drift is defined as change due to the dependent variable where the relationship between X and Y is changed without any change in the data distribution. For example, in the anomaly detection model, there may be changes in the definition of what is considered an anomaly over time. Concept drift occurs due to changes in any component of the concept which are categorized into Real and Virtual drifts. **Real Drift:** This involves a change in posterior probability distribution (i.e. $P(Y|X)$). This change affects the decision boundaries and thus poses a threat to the learning system **Virtual Drift:** This involves changes in prior probability (i.e. $P(Y)$) and change in class conditional probability distribution (i.e. $P(X|Y)$). These changes are generally termed as changes in an attribute value or class distribution that does not affect decision boundaries.

Concept drift can also be categorized based on the frequency, magnitude, transition speed, and recurrence as illustrated in the figure below [162]

2.2.3 Concept Drift Adaptation

In literature, three main approaches are used for handling concept drift in the data stream, which include: window-based approaches, weight-based approaches, and ensemble classifiers [47, 56, 120]. These methods are generally categorized into active and passive approaches [44]. Active approaches such as drift detectors specifically aim at detecting concept drift and use mechanism that detects the presence of the change. There are many drift detection methods used for detecting concept drift which is intensively reviewed in [60]. One of the key challenges faced by active approaches is failing to detect the change or wrongly detecting non-existing change (false alarm). The passive approach manages the loopholes in active approaches through the use of an adaptation mechanism that allows the model to be updated every time new data arrives regardless of whether concept drift occurs or not. Passive approaches include models that are developed based on updating a single classifier and models that involve adding, removing and modification of ensemble members.

Although passive approaches that involve updating a single classifier have advantages over ensemble classifiers in terms of speed and low computational cost, they suffer many challenges when the concept changes suddenly and rapidly. As a result ensemble classifiers are used due to their utmost predictive accuracy and stability-elasticity property. This property makes it easy for them to incorporate new data into the model, by training and adding new members to the ensemble, and naturally, forget irrelevant knowledge by removing the old members from the ensemble [44]. In

today’s literature, ensemble learners remain the most popular approach to handling concept drift in data stream mining. Two main approaches involve which includes Chunk-based approach (that involves adding a new classifier that is trained on a recently arrived chunk of data) and the Online-based ensemble approach (that involves updating the base classifier in the ensemble) [120].

2.2.4 Uncertainty Estimation

Uncertainty estimation in prediction is the process of quantifying uncertainty (or confidence) in prediction models where instead of estimating an optimal value, the model can estimate probability distributions. It is the process of demonstrating Bayesian inference, which aims at finding the posterior distribution over model parameters $p(\omega|X, Y)$, given a set of sub-sequence X and Y where X represents h_w and Y represent p_w . With a new data point $x \in X$, the distribution of the prediction is obtained by marginalizing out the posterior distribution, as shown in (2.1), where ω is the collection of model parameters.

$$p(y|x) = \int_{\omega} p(y|x, \omega)p(\omega|X, Y)d\omega \quad (2.1)$$

Taking the variance of this distribution will quantify the prediction uncertainty, which is further elaborated using the law of total variance, as shown in (2.2).

$$Var(y|x) = Var[E(y|\omega, x)] + E[Var(y|\omega, x)] = Var(\omega(x)) + \delta^2 \quad (2.2)$$

From the last part of (2.2), we can see that the variance is decomposed into two terms: (i) $Var(\omega(x))$, which represent model uncertainty that reflects our ignorance over model parameters ω ; and (ii) δ^2 which represent the noise level during the data generation process, referred to as inherent noise or data uncertainty (aleatoric). Neural Network uncertainty can be categorized into two; uncertainty in the data called aleatoric and uncertainty in prediction called epistemic. Aleatoric uncertainty can be learned directly from the data while several methods are used for inferring epistemic uncertainty which includes Bayesian neural networks and sampling during inference. In this thesis, we quantify aleatoric uncertainty using quantile regression.

2.2.5 Quantile Regression

In a classical regression problem, the most commonly used loss function is mean square error. The regression model tries to minimize this loss that corresponds to normal distribution. Quantile regression aims at forecasting conditional quantiles are given an input sequence [3]. In this context, models are trained to minimize quantile

loss which is defined in (2.3).

$$L(\xi_i|\alpha) = \begin{cases} \alpha\xi_i & \text{if } \xi_i \geq 0 \\ (\alpha-1)\xi_i & \text{if } \xi_i < 0 \end{cases} \quad (2.3)$$

where α is the required quantile with values between 0 and 1 and $\xi_i = y_i - f(x)_i$ with $f(x)_i$ as the predicted quantile and y_i as the observed value for the corresponding input x . To create a quantile loss for the entire dataset, the average of the quantile loss for an individual point is taken, as shown in (2.4).

$$L(y, f|\alpha) = \frac{1}{N} \sum_{i=0}^N L(y_i - f(x_i)|\alpha) \quad (2.4)$$

Models developed using quantile regression are useful in areas of application where it is difficult to define a parametric distribution on the residuals [3].

2.2.6 Anomaly

Anomaly is an unexpected pattern in the data that deviate from the normal behavior of the data [29]. In time series, an anomaly can be defined as an unexpected point in time series (e.g., a sudden sensor drift), an anomalous sub-sequence within the time series (e.g., a continuous change in the sensor readings), or points that are anomalous based on defined context, or an anomalous time series within the entire time series database [150]. Anomalies might be induced in the data for a variety of reasons, such as malicious activity, for example, credit card fraud, cyber-intrusion, terrorist activity or break- down of a system, but all the reasons have the common characteristic that they are interesting to the analyst. In time series, an anomaly can be classified as a point, contextual, or collective anomaly.

2.2.6.1 Point Anomaly

Point anomaly is the most common and simplest type of anomaly that dominated the research literature on anomaly detection. An example is in a credit card fraud detection system where a transaction that is high than the normal range of expenditure of a person is classified as a point anomaly.

2.2.6.2 Contextual Anomalies

Contextual anomaly is referred to as a conditional anomaly where a point is classified as anomaly based on specific context [29]. Current literature [71, 72] shows how contextual anomaly is widely explored in time series. In a contextual anomaly problem setting, the data instance is defined using the following sets of attributes:

2.3 Problem Concept for Anomaly Detection

1. **Contextual attribute:** Contextual attribute is used to define the context or neighborhood of a data instance. In time series, for example, a timestamp that defines the position of a data point is referred to as a contextual attribute.
2. **Behavioural attributes:** These are attributes that define the pattern or non-contextual characteristics of a data point. For example, in the average rainfall time series, the amount of rainfall at any location is a behavioral attribute.

To determine the behavior of an anomalous point, the values for the behavioral attributes within a specific context are used. Literature [9] shows that all anomaly detection problems can be converted to contextual anomaly when context information is incorporated when setting up the problem. This behavior demonstrates the need for identifying contextual and behavioral attributes when developing any contextual anomaly detection method.

2.2.6.3 Collective Anomalies

A collection of related data points that are anomalous with respect to the other points in the entire data set is referred to as a collective anomaly. In collective anomaly, a single data point may not be anomalies on its own but become anomalous when it is combined with other sets of data points. Research in literature [9, 22] has explored Collective anomalies for sequence data. Unlike point anomalies, collective anomalies occur in datasets with related data points.

2.3 Problem Concept for Anomaly Detection

In this section, we will review different concepts for the formulation of anomaly detection problems. Factors such as problem domain, data characteristics, and availability of class labels will justify the choice and applicability of a particular anomaly detection technique.

2.3.1 Nature of Input Data

The performance of any anomaly detection technique depends on the nature of its input data [29]. time series is a collection of time-ordered points which are described using a set of attributes. The attribute can be binary, categorical, or continuous with a single (univariate) or multiple attributes (multivariate). In univariate time series, only one variable is varying over time which means there is only one observation over a period of time [13]. For example, data collected from a sensor measuring the temperature of a room every second. Therefore, each second, you will only have a one-dimensional value, which is the temperature. Multivariate time series consists of more

2.3 Problem Concept for Anomaly Detection

than one observation collected over time which means multiple variables are changing simultaneously over time [13]. Analysis of multivariate time series is more challenging than univariate time series due to the number of variables to deal with. Univariate time series forecasting treats prediction as essentially a single-variable problem, whereas multivariate time series may use many time-concurrent variables for prediction. As such designing and correlating multiple variables across hierarchical levels varies from system to system. To handle these challenges in machine learning, transformation methods such as Principal Component Analysis (PCA) and factor analysis are used to reduce the attribute space from large numbers to smaller numbers of factors. In addition, the time series attribute type determines the applicability of any anomaly detection technique. For instance, statistical anomaly detection techniques are commonly applied in time series with continuous and categorical attributes. In a similar context, the type of attribute would determine the distance measure to be used for nearest-neighbor-based methods. In these approaches, the pairwise distance between the data instances is computed using distance and similarity measures. This is different from statistical and classification-based techniques that require original data instances.

The relationship among data instances is another key factor that is used to categorize input data [68]. For example, time series, genome sequences, and protein sequences, which are linearly ordered based on time are referred to as sequence data. On the other hand, graph data represents data instances as vertices that are connected to each others using edges.

2.3.2 Types of Anomaly

Knowing the type of anomaly that will be detected is the basis for any anomaly detection method. For example, The availability of contextual attributes determines whether applying a contextual anomaly detection method will be meaningful or not. In some cases, defining context is easy which enhances the use of contextual anomaly detection techniques. While in other cases, it is a difficult task to define a context which makes it difficult to use those techniques. Similarly, contextual information must be included as part of the problem definition which can be made through the data set. A data point might be an anomaly in a given context, while the same point with the same attribute's behavior could be normal in a different context.

2.3.3 Data Labels

Data Labels refer to normal or anomalous class labels associated with data points. Dataset labeling is usually carried out manually which required a substantial effort from the domain experts to fully labeled the dataset. Anomalous data points are rare which makes them even more difficult to obtain than normal data instances. For example, in an aircraft monitoring system, an anomaly may refer to dangerous events

2.3 Problem Concept for Anomaly Detection

which are very rare. This makes it difficult to label the dataset with all possible types of anomalous behavior. Similarly, anomalous behavior in time series is dynamic in nature which results in changes in their concept. Depending on the availability of class labels, Anomaly detection techniques can be divided into the following three modes:

2.3.3.1 Supervised Anomaly Detection

This is a binary classification problem where the model is trained in supervised mode with pre-labeled datasets of normal and anomaly classes. In this scenario, a predictive model is trained with both normal and abnormal data points to predict the class of unseen data points. The output of the model is compared with an unseen data instance to determine which class it belongs to. Supervised anomaly detection techniques are faced with the challenge of imbalance class distribution where anomalous points are fewer than normal points in the training data. Some data mining and machine learning literature [33, 82, 83, 115, 158] have addressed this issue through either sampling, one-class learning or feature selection but, obtaining accurate and representative anomaly class labels is usually challenging. A number of techniques have been proposed that inject artificial anomalies in a normal data set to obtain a labeled training data set [1, 143, 151] but these artificial classes cannot fully represent the actual anomalies. In addition, labeling large volumes of data is often difficult, costly, and sometimes incomplete due to big data resource challenges.

2.3.3.2 Semi-supervised Anomaly Detection

In a situation where normal data is available, semi-supervised anomaly detection techniques can be used. These techniques assume that the training data is made up of only normal classes. Semi-supervised anomaly detection techniques are more applicable than supervised techniques because they don't require anomaly class which is often rare. For example, in spacecraft fault detection [52, 53], an anomaly scenario that indicates an accident will not be easy to model. These techniques trained the model using data corresponding to normal behavior which is later used to identify anomalies existing in the test data. On the other hand, literature [40, 41] some anomaly detection methods that assume the availability of only anomaly class during training. These techniques are not commonly used, due to difficulties in obtaining a dataset that contains all possible anomaly behavior.

2.3.3.3 Unsupervised Anomaly Detection

Unsupervised anomaly detection techniques are widely used because of the non-requirement of labeled training data. In these techniques, a model is trained in an unsupervised mode with the assumption that normal data points are more common than abnormal

2.4 Anomaly Detection Techniques for time series

data points. This approach assigned anomaly scores based on the degree to which the data point shows the anomaly profile. Researches in literature [5, 94, 126] shows the use of this technique in static and dynamic time series. Semi-supervised techniques when trained with unlabeled data can be converted to work as unsupervised techniques. This is done by the assumption of test data contain very few anomalies and that the model learned to be robust.

2.3.4 Output of Anomaly Detection

Depending on the nature of the data used for model training, the output produced by anomaly detection techniques is of two types.

2.3.4.1 Anomaly Scores

In this type of output approach, anomaly detection techniques generate an anomaly score that shows the degree to which a data point is anomalous. The output generated comprises a list of anomalies ranked based on their scores. The list will then be analyzed where a fixed threshold can be used to determine when to generate an anomaly alert.

2.3.4.2 Class Labels

Anomaly detection techniques in this category assign a label (normal or anomalous) to each test instance. In this approach, an anomaly score is generated which is used to assign a label to the data point. These techniques differ from Scoring based anomaly detection techniques by providing binary labels as output. As such they do not allow an analysis of the degree to which a data point is anomalous even though it can be indirectly controlled through model parameter settings.

2.4 Anomaly Detection Techniques for time series

The previous section discusses different understandings and concepts of anomaly detection problems. In this section, we will review different anomaly detection techniques proposed in the literature to suit a particular anomaly detection problem. Different anomaly detection techniques were proposed in literature [29] which are called traditional methods of anomaly detection [150]. Traditional anomaly detection methods mostly focus on identifying anomalous data points without considering the sequence or temporal nature of time series data. This limits their applicability in time series which is the focus of this research work. A survey by Gupta et al.[68], reviews anomaly detection techniques for time series data which are classified based on how

they identify anomalies. In time series, anomaly detection problem is viewed in different ways. The question is; are we looking for an unusual data point in the time series (e.g. a sudden sensor drift), an anomalous sub-sequence within the time series (e.g. a continuous change in the sensor readings), or time series instances that are anomalous in a specific context, or an anomalous time series within the entire time series database? In this section, we will review these four different possible formulations of anomaly detection problems.

2.4.1 Detecting Point Anomalies

In this type of problem setting, the goal is to identify individual data points which are anomalies within the entire sequence or sub-sequence of the time series. Point-based anomaly detection methods are also used for identifying sensor drift. For example, a sensor reading at a particular second might be different when compared with other recorded readings per second. Point anomalies are widely explored in time series data where many detection techniques were proposed in literature [68].

2.4.2 Detecting Anomalous Sub-sequence

This is another anomaly detection problem setting that identifies anomalous sub-sequence within a given time series. Anomalous sub-sequence is also referred to as collective anomalies [22]. In collective anomalies, a single data point may not be an anomaly but the occurrence of that point with other related points might be an anomaly. In sensor readings, for example, a low or high sensor value that is observed over a period of time, might not be an anomaly unless it occurs in several other places. The anomalous sub-sequences are also called discords. “Discords are the sub-sequences of a longer time series that are maximally different from the rest of the sequence” [86]. The concept of discords was also used for identifying anomalous sub-sequences within a time series [50, 87].

2.4.3 Detecting Contextual Anomalies

In this type of anomaly detection problem setting, the anomalies are defined as data points that are anomalous in a specific context. Contextual anomaly detection problems can be handled in the same way as point anomaly detection problems when separate anomaly detection methods are applied to the same dataset within different contexts. Contextual anomaly detection problems have been widely in time series data [9, 71, 72]. However, they face the challenge of defining the context of normal and abnormal data apriori which may not be possible within many big data applications.

2.4.4 Detecting Anomalous time series in time series Database

In this setting, an anomaly detection method tries to determine anomalous points in the test data with respect to a database of training data [30]. The training data consist of only normal data points in a semi-supervised setting or an unsupervised setting where both normal and anomalous points are present in the training data with an assumption that the majority are normal data points. This thesis will on the unsupervised anomaly detection problem setting for multivariate time series and data streams.

2.5 Review of time series Anomaly Detection Techniques

This section will review various anomaly detection techniques for time series data which are existing in the literature. Anomaly detection techniques differ in the way they identify anomalies which involve the following steps:

1. Computation of anomaly score for individual data point or sub-sequence of a given test time series. This score is computed as the measure of the discrepancy between the output of the detection technique and the actual time series. This can be based on the rate of dissimilarity between the sub-sequences, prediction, or reconstruction errors.
2. Aggregation of anomaly scores for individual data points to determine the total score of the entire time series. This can be achieved via computing the average of all the anomaly scores, top k anomaly scores, or log of anomaly scores.

In order to detect anomalies, a fixed threshold value is used where data points are labeled as anomalies when their anomaly scores exceed that value. These techniques are categorized into distance-based, prediction-based, and deep learning-based anomaly detection techniques.

2.5.1 Distance Based Techniques

Distance-based anomaly detection involves the use of distance and similarity measures in the detection of anomalies. These techniques are different in the way they choose the distance measures for the computation of the anomaly score. Distance/similarity measures such as correlation, Euclidean, Mahalanobis, Cosine, and DTW, etc. can be used are used for the computation of the distance between the sample time series and the center of the detection model. When the distance is not more than a pre-defined threshold, the sample time series is considered to be normal and anomalous otherwise. Several distance-based anomaly detection techniques exist in the literature which are categorized into window-based, proximity-based, and transformation-based as discussed in the following subsections:

2.5.1.1 Window Based

Window-based anomaly detection divides time series into fixed-size windows (sub-sequences) to identify anomalies. The windows are extracted by shifting one or more time steps at a time. Once the windows are extracted, an anomaly detection technique is used to assign an anomaly score to each window which is accumulated to determine the anomaly score of the test time series. The anomaly detection process in window-based techniques is formally described as follows :

1. Given a training dataset, $X_{training} = X_1, X_2, \dots, X_n$, extract w windows of each time series X_i as $x_{i1}, x_{i2}, \dots, x_{iw}$, where w can be calculated as $|X_i| + l_1$ by sliding each size l window, one step ahead. Similar is done to the test sequence where $X_{test} = T_1, T_2, \dots, T_n$, is also divided into $|T_i| + l_1$ windows as $t_{i1}, t_{i2}, \dots, t_{iw}$. The choice of window size mostly depends on the nature of the data. However, several methods are adopted in literature which include weighted sum [12], adaptive windowing [159], and auto-correlation [146]. An alternative approach is to manually select window size by running the model with many possible window sizes and choosing the window size with minimum prediction errors or using cross-validation and choosing the window that generalizes best.
2. The anomaly score for each test window ($a(t_{ij})$) is calculated using its similarity to the training window. This similarity function can be distance measures like Euclidean, Manhattan Correlation values, etc. For example, the anomaly score of a test window can be the distance to its k th nearest neighbor among the training windows [87], or as a prediction error that measures the discrepancy between the predicted and actual test window [131], or in a classification context where the anomaly score of a test window is 0 or 1 depending on whether it is classified as normal or anomalous [101].

Strengths and Weaknesses: The drawback of window-based techniques is that the expected pattern has to be defined and window size has to be chosen carefully in order explicitly capture the anomaly. The optimal size of the window depends on the length of the anomalous region in the anomalous time series. Another drawback of window-based techniques is that they are computationally expensive. Since every pair of test and training windows are compared, the complexity is $O((nm)^2)$, where n is the average length of the time series, and m is the number of test and training time series in the database. Most of the window-based techniques are proposed for detecting anomalous sub-sequence or discord detection problems. These drawbacks of window size and computational complexity are addressed by some of the discord detection techniques.

Window-based techniques can capture all different kinds of anomalies mentioned in 2.3 such as an anomalous observation in a time series, an anomalous sub-sequence

2.5 Review of time series Anomaly Detection Techniques

in a time series, and an anomalous time series as a whole. Since the entire time series is broken into smaller sub-sequences, we can easily identify if an observation or the sub-sequence is anomalous. If the entire time series is anomalous then all the sub-sequences are also anomalous, hence window-based techniques would capture it well.

2.5.1.2 Proximity Based

Proximity-based anomaly detection techniques are extensions of distance-based techniques. These techniques make use of the pairwise proximity between the test and training time series using an appropriate distance or similarity kernel to compute the anomaly score of the test time series [23, 77, 118]. The assumption behind these techniques is that the anomalous time series are “different” from the normal ones and this difference can be captured using a proximity measure. If the proximity of the region where the sample test time series is located is less than the predefined threshold, it is considered anomalous; otherwise, it is normal. Usually, density-based methods can obtain relatively higher detection accuracy than other methods. The anomaly score of a test time series with respect to the training database is calculated using the following methodologies:

1. **k-NN:** The distance of the test time series to its kth nearest neighbor in the training data set is its anomaly score.
2. **Clustering:** The training time series are clustered into a specified number of clusters and the cluster centroids are computed. The distance of the test time series to its closest cluster centroid is its anomaly score [5].

Strengths and Weaknesses: The drawbacks of proximity or similarity-based techniques are that they can determine if the entire time series is anomalous or not, but cannot exactly locate the anomalous sub-sequence. To localize the exact region(s) within the time series which causes the anomaly, one needs to do post-processing of the time series. Also, the phase misalignment and the non-linear alignments of different time series, which are some common challenges for the time series data, restrict the usage of different proximity measures for this class of techniques. Thus the performance of these techniques is highly dependent on the proximity measure, which is often not easy to choose. In detecting different types of anomalies, similarity-based techniques might fail when a single observation is anomalous in the time series as its effect might not be prominent when the entire time series are considered at once. These techniques would detect anomalies such as anomalous sub-sequences in a time series or an anomalous time series as a whole.

2.5.1.3 Transformation Based

Current digital era results in generating high dimensional time series with multiple attributes. In order to deal with such type of data, anomaly detection techniques that rely on the use of dimensionality reduction are required. The transformation-based anomaly detection techniques used transformation methods such as Haar Transform [50], Symbolic Aggregate Approximation (SAX) [86, 87], Principal Component Analysis (PCA) [81, 106], Partial Least Square (PLS) [57], and Wavelet transform [98, 170] to transform and extract useful features from the data in data pre-processing step.

2.5.1.4 Statistical Distance-based

Statistical distance-based handle anomaly detection problems as a distance measure between two probability distributions. Statistical distance metric such as Kullback-Leibler divergence (KLD) is used as a monitoring index to identify the dissimilarity between two probability distributions which allow automatic anomaly detection [70]. The referenced paper used partial least square for modeling and KLD as anomaly indicators that quantify the dissimilarity between current PLS-based residuals and reference probability distribution obtained using fault-free data. In another research, Wang et al proposed an anomaly detection method that constructs the probability density function of the different patterns using KLD theory [161]. The authors used PCA to deduce the measurement matrix and set up a statistical detection indicator using KLD under an identically independent Gaussian noise background.

2.5.2 Prediction Based Techniques

Prediction-based anomaly detection techniques work in the same scenario as standard learning algorithms, where a detection model receives a test time series as input and outputs a decision about its abnormalities. This means if the test time series conforms to the generation mechanism and fits the obtained model in the process of anomaly detection, it is regarded as normal; otherwise, it is anomalous. From the perspective of whether or not to use the label time series during the training of the model, the methods can be categorized as a supervised learning method, unsupervised learning method and semi-supervised learning method [29, 68]. A basic predictive model-based anomaly detection technique consists of the following steps:

1. Learn a predictive model on the given training time series, which uses values from l number of time steps (history) to predict the next $(l + 1)th$ time steps following them.
2. For a test time series, using the predictive model built in step 1, forecast the value at each time step using the values seen so far (previous l steps).

2.5 Review of time series Anomaly Detection Techniques

3. Compute the anomaly score of individual observation or sub-sequence of a given test time series using the prediction error. The prediction error corresponding to observation is a function of the difference between the forecasted value and the actual observation, and certain model parameters such as the variance of the model. Finally, the anomaly scores of individual observations or sub-sequences are aggregated to calculate the anomaly score of the given test time series which is used for identifying anomalies.

In this section, we provide an extensive review of prediction-based anomaly detection techniques. These techniques differ in the prediction models used and can be broadly classified into classical regression-based, machine learning-based, and deep learning based.

2.5.2.1 Classical Regression Based Models

Several statistical techniques have been proposed to detect anomalies within a time series using various time series modeling techniques such as Moving Average (MA) [31], AutoRegression (AR) [31, 52], ARMA [116], ARIMA [107], Kalman filters [88], and general regression [49, 127]. The input to these models is the entire time series and the length of the history (l), also denoted as the order of the model. These models differ in the kind of filters they use to generate the output.

Moving Average (MA): The MA models represent time series that are generated by passing the input through a linear filter which produces the output y_t at any time t using only the input values x_{t-i} , $0 \leq i \leq l$ as shown in equation 2.5. This is also called a non-recursive filter.

$$y_t = \sum_{i=0}^l b_i x_{t-i} + \epsilon_t \quad (2.5)$$

where, b_1, b_2, \dots, b_l are the coefficients of the non-recursive filter, ϵ_t is the noise at time t . If every instance t of a time series has a value equal to the mean of its previous l values then it can be represented by a moving average model, in which case $b_t = \frac{1}{l}$ and $\epsilon_t = 0$.

Autoregression (AR): The AR models represent time series that are generated by passing the input through a linear filter which produces the output y_t at any time t using the previous output values y_{t-i} , $0 \leq i \leq l$ as shown in equation 2.6. This is also called a recursive filter.

$$y_t = \sum_{i=0}^l a_i y_{t-i} + \epsilon_t \quad (2.6)$$

where, a_1, a_2, \dots, a_l are the autoregressive coefficients of the recursive filter, ϵ_t is the noise at time t .

2.5 Review of time series Anomaly Detection Techniques

Autoregressive Moving Average (ARMA): The ARMA models represent time series that are generated by passing the input through a recursive and through a non-recursive linear filter consecutively as shown in equation 2.7. In other words, the ARMA model is a combination of an autoregressive (AR) model and a moving average (MA) model. The orders of AR part of the model and MA part of the model can differ.

$$y_t = \sum_{i=0}^l a_i y_{t-i} + \sum_{i=0}^l b_i x_{t-i} + \epsilon_t \quad (2.7)$$

where, a_1, a_2, \dots, a_l are the autoregressive coefficients of the recursive filter, b_1, b_2, \dots, b_l are the coefficients of the non-recursive filter, ϵ_t is the noise at time t .

Autoregressive Integrated Moving Average (ARIMA): The ARIMA models which extend ARMA models, apply the ARMA model not immediately to the given time series, but after its preliminary differencing, which is the time series obtained by computing the differences between consecutive values of the original time series.

Kalman Filters: The basic idea of Kalman filter is described as follows - Consider, a time series with Markov property, described by the equation 2.8

$$x_{t+1} = Ax_t + \epsilon_t \quad (2.8)$$

where x_t represents a hidden state of the system and A is a matrix describing the causal link between current state x_t and next state x_{t+1} . The Kalman filter for time series $X = (x_1, x_2 \dots x_n)$ is described in equation 2.9

$$y_{t+1} = Ax_t + K(x_t y_t) \quad (2.9)$$

where K is called the Kalman gain.

General Regression: Linear regression [49], Gaussian process regression [121], Support vector regression [100], etc. The subsequences of length l (history length), extracted from the original time series are the input to these models. The training set will be a set of subsequences given by, $T = (X_t, y_t), t = l, \dots, n-1$, where $X_t = [x_{t+l+1} \dots x_t]$ and $y_t = x_{t+1}$. A linear regression function is constructed using a weight vector W and a mapping function $\phi(X_t)$, $y = W^T \phi(X_t) + b$. Different regression models differ in how they fit the function.

(a) Linear regression: For simple linear regression the above equation is solved by minimizing the sum of squared error of the residue ($y_i x_{i+1}$). The mapping function here is identity, $\phi(X_t) = X_t$.

(b) Support vector regression : These models use the ϵ insensitive loss function proposed by Vapnik [156]. They solve the above equation by solving the objective

function in equation 2.10

$$\begin{aligned}
 \text{minimize } P &= \frac{1}{2} \|W\|_2 + C \sum_{i=1}^l (\zeta_i + \zeta_i^*) \\
 \text{such that, } &y_i(W^T \phi(X_t) + b) \leq \epsilon + \zeta_i \\
 &= y_i(W_T \phi(X_t) + b) \leq \epsilon + \zeta_i \\
 &= \zeta_i, \zeta_i^* \geq 0
 \end{aligned} \tag{2.10}$$

This optimization criterion penalizes data points whose y values differ from $f(x)$ by more than ϵ . The slack variables ζ_i, ζ_i^* represent the amount of excess deviation. Different kernel functions [21] such as polynomial, RBF and sigmoid can be used.

Ma et al [100] use m -length training sub-sequences and use support vector regression on them for building an online novelty detection model which also uses statistical tests to determine the anomalies with some fixed confidence. Chandola et al [30] use AR model on the original time series while Lotze et al [98] use wavelet transformations of the training time series to produce multiple-resolution data and then use AR model, Neural Networks on them for prediction. In another context, a simple prediction model was built based on the lowest wavelet resolution (called trend) of the sub-sequences [170]. Let the original sub-sequence be $X_i = x_1, x_2 \dots x_{i-1}$, and its trend be $Y_i = y_1, y_2 \dots y_{i-1}$. The distribution of the difference of X_i and Y_i , called residual, is constructed. If the difference of the x_i and the trend y_{i-1} is statistically insignificant as per the residual distribution, then the observation x_i is termed anomalous.

2.5.2.2 Machine Learning Based

Machine learning-based anomaly detection techniques work in the same scenario as standard learning algorithms, where a detection algorithm receives new data instances as input and outputs a decision about its abnormalities. These techniques fall into three categories similar to classification algorithms: supervised, unsupervised, and semi-supervised [29].

The supervised methods involve training the detection algorithms with a preformed dataset with entries labeled normal and abnormal. Some statistical theory or machine learning methods, such as Gaussian distribution [121], Support Vector Machine (SVM) [51], Hidden Markov Model (HMM) [62, 93], and Artificial Neural Network (ANN) [103] are used for this purpose. In a word, the taxonomy of above-mentioned methods has some overlap to some extent and an obvious conclusion can be drawn that machine learning-based methods especially based on ANN are the dominating methods in the literature [68].

For Unsupervised methods, the goal of machine learning approaches for anomaly detection is to model the distribution of normal data. This allows for distinguishing

2.5 Review of time series Anomaly Detection Techniques

anomalous patterns from what is expected based on the available normal data. For multivariate time series data, this can be achieved e.g. by learning a multivariate Gaussian distribution $N(\cdot)$ that includes covariance statistics of the columns from the training dataset. The distribution of the normal data can be estimated by calculating μ as shown in equation 2.11 based on the available data xR^m per time step, where m is given by the number of columns.

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)} \quad (2.11)$$

The probability of a sample per time step belonging to the normal distribution is then given by $p(x)$ in equation 2.14. In practice, known anomalies as well as normal data which was not used for training can now be utilized to estimate a threshold parameter ϵ , so that $p(x) > \epsilon$ can be used to predict anomalies in new data. This parameter can be cross-validated in order to achieve sufficient accuracy for anomaly detection.

A semi-supervised approach involves the use of One-Class Support Vector Machines (OC-SVM), introduced by Schölkopf et al. [133]. OC-SVMs learn a hypercube from the distribution of the training data. This allows categorizing of novel samples according to their distance from the hypercube. OC-SVMs have been broadly used for anomaly or novelty detection tasks and can therefore be seen as a good baseline method for comparison. Even though, multivariate gaussian distributions and OC-SVMs anomaly detection methods can model covariance between values in different columns of multivariate time series. However, these approaches fail to model temporal dependencies between the column-wise values at different time steps. This can significantly reduce the potential to detect anomalies in a multivariate setting where these temporal dependencies would have to be encoded in features that introduce additional manual effort that requires good knowledge of the data domain. This is where recently published deep learning approaches show their full potential. In the next section, different deep-learning approaches for anomaly detection in time series data will be discussed.

2.5.3 Deep Learning Based

Deep learning describes a set of practices and algorithms for numerous architectures of deep neural networks, where the term deep refers to architectures consisting of multiple hidden layers. With these deep neural networks, the goal is often to derive hierarchical hidden representations of raw input data in order to solve a narrow task. As an example, in computer vision applications, deep convolutional networks are trained to detect different visual features from given images to categorize objects. This example shows the advantage of neural networks compared to traditional machine learning algorithms like support vector machines (SVM). Traditional machine learning algorithms have proven to be unsuccessful when it comes to solving tasks like object or

2.5 Review of time series Anomaly Detection Techniques

speech recognition, which are considered as central problems in artificial intelligence [61]. Especially on data with a high dimensional input space, simple algorithms cannot generalize sufficiently due to the sheer amount of possible different configurations of the input data, which is often much larger than the available training samples. In general, deep neural networks can learn latent features from raw data, whereas in the case of traditional learning algorithms, the input features have to be carefully engineered which often requires extensive domain knowledge. The performance of deep learning algorithms in unsupervised representation learning of time sequence applicable to text [35], video [154], and speech recognition [64, 65], show their ability to learn hierarchical discriminating features and handle the temporal nature of time series data [67].

This practical advantage of deep learning algorithms offers high potential in use cases, where relevant input features cannot be manually defined due to a lack of domain knowledge. Recent advancements made by deep learning methods in various machine learning problems, have also encouraged researchers to explore them for anomaly detection [28]. Current literature shows the use of deep learning algorithms such as Recurrent Neural Network (RNN) particularly based on Long Short-Term Memory (LSTM) [4, 97, 103] or Gated Recurrent Unit (GRU) [85, 131], Convolutional Neural Network (CNN) [110], and Autoencoder [5, 96, 99, 102, 126, 134, 173] for anomaly detection in time series data. Various architectures based on DNNs have been proposed with applications for anomaly detection in time series data. In this section, we will discuss the different architectures of these deep learning algorithms and their application for anomaly detection in time series data.

2.5.3.1 Recurrent Neural Network (RNN)

Recurrent Neural Networks are a branch of Artificial Neural Networks (ANNs) that used the backpropagation method to tune their parameters and optimize performance through Stochastic Gradient Descent (SGD). RNNs are called recurrent because of their ability to perform the same operations for all elements in a sequence of inputs. In literature, RNNs are now popularly used in analyzing time series data. Their performance in stock price forecasts [123, 135, 157] and an unsupervised representation learning applicable to text [35], video [154], and speech recognition [64, 65], shows their ability to handle temporal nature of time series data. In addition, they are applicable in autonomous driving systems, where they can anticipate car trajectories and help to avoid accidents [111], fault detection [144], and car or engine health monitoring [67]. The basic idea is to learn a temporal model of the system that captured the temporal as well as instantaneous dependencies between time series (or sensor readings). Figure 2.1 shows an example of RNN architecture where x_t is the input at time step t . For example, x_t is the input at time period one and a_t is the activation value for the hidden state at time step t . The activation value is calculated using an activation function

2.5 Review of time series Anomaly Detection Techniques

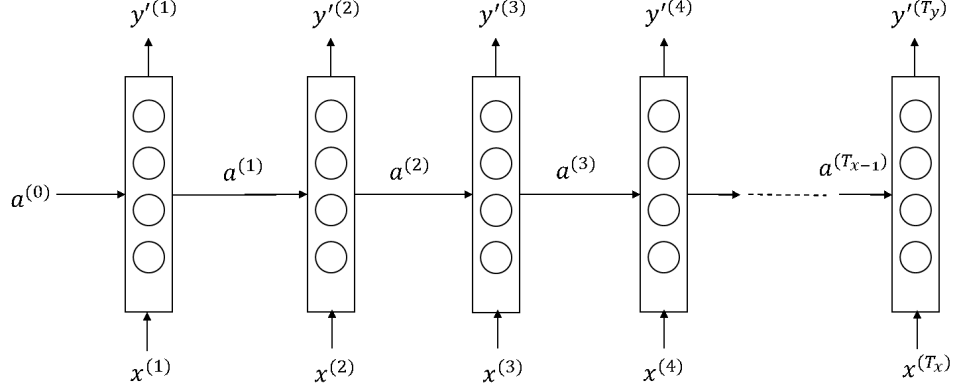


Figure 2.1: Recurrent Neural Network Architecture

using input from the previous hidden state and the current state. The activation value at time zero a_0 is usually initialized to zero (i.e. a vector of zeros). y'_t represent the predicted (output) value at time step t . It shows, for example, to predict the next value in a sequence; we will have a vector of probabilities across our time series. RNN cells are developed on the fact that one input is dependent on the previous input by having a hidden state or memory that captures what has been seen so far. It means the activation value of the hidden state at any point in time is a function of the activation value of the hidden state at the previous time step plus the input value at the current time step as shown in equation 2.12. Then, the output labels or predicted values are determined by the non-linear mapping of the hidden layers using the mathematical expression, as shown in equation 2.13.

$$a_t = g(W_{aa}a_{t-1} + W_{ax}x_t + b_a) \quad (2.12)$$

$$y'_t = f(W_{ya}a_t + b_y) \quad (2.13)$$

Where g and f are non-linear activation functions, W and b represent weight and biases. g can be chosen from sigmoid or tanh and f from sigmoid or softmax.

Similar to other artificial neural networks, an RNN used SGD to find and adapt to its set of internal model parameters (weights and biases) that perform well against some performance measures such as logarithmic loss or mean squared error on the training sets. Back propagation through time is normally used as the process of computing the gradient, which is obtained by calculating the partial derivatives of the cost function with respect to any weight or bias. The idea is to reduce the cost function C , thereby updating the network weights by SGD.

Deep RNNs consist of multiple hidden layers of recurrent units stacked one on top of the other with the output sequence of one layer forming the input sequence of another layer. This will helps in capturing the temporal dependencies at different time steps [84] as well as dependencies across dimensions for multivariate time series

[131]. The multiple hidden layers can be GRU or LSTM units to handle the vanishing gradient problem.

2.5.3.2 Gated Recurrent Units (GRU)

Gated Recurrent Unit is another RNN architecture that was proposed to make each recurrent unit adaptively capture dependencies of different time scales [37]. Similar to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit but without having a separate memory cell. GRU consists of two gates, namely *update gate* and *reset gate*, which control the flow of information by manipulating the hidden state of the unit. The reset gate is used to compute a proposed value for the hidden state at time t by using the hidden state at time $t - 1$ and the hidden state of the units in the lower hidden layer at time t . The update gate decides what part of the previous hidden state and proposed hidden state will be used to obtain the current hidden state at time t .

For a multilayered RNN with L hidden layers, the hidden state c'_t at time t is obtained from c_{t-1} . The time series goes through t the following transformations iteratively for $t = 1$ through T , where T is length of the time series as shown in equation 2.14 to equation 2.17:

$$\text{Proposed state: } c'_t = \tanh(W_c[c_{t-1}, x_t] + b_c) \quad (2.14)$$

$$\text{Update gate: } \gamma_u = \sigma(W_u[c_{t-1}, x_t] + b_u) \quad (2.15)$$

$$\text{Reset gate: } \gamma_r = \sigma(W_r[c_{t-1}, x_t] + b_r) \quad (2.16)$$

$$\text{Hidden state: } c_t = \gamma_u \times c'_t + (1 - \gamma_u) \times c_{t-1} \quad (2.17)$$

where W_c, W_r , and W_u are weight matrices of appropriate dimensions such that c'_t, γ_u, γ_r , and c_t are vectors in \mathbb{R}^c , where c is the number of units in the layer. The sigmoid (σ) and \tanh activation functions are applied element-wise.

From the above equations, the procedure of taking a linear sum between the existing state and the newly computed state is similar to the LSTM unit. The main difference between LSTM and GRU is that the content of the GRU cell is always exposed to the output and does not have any mechanism to control the degree to which its state is exposed. This feature makes GRU easier to implement since it requires fewer network parameters.

2.5.3.3 Long Short Term Memory (LSTM)

Long Short-Term Memory is an RNN architecture used for learning long-term dependencies in time series data. LSTM overcomes the vanishing gradient problem by replacing an ordinary neuron with a complex architecture called an LSTM unit or block.

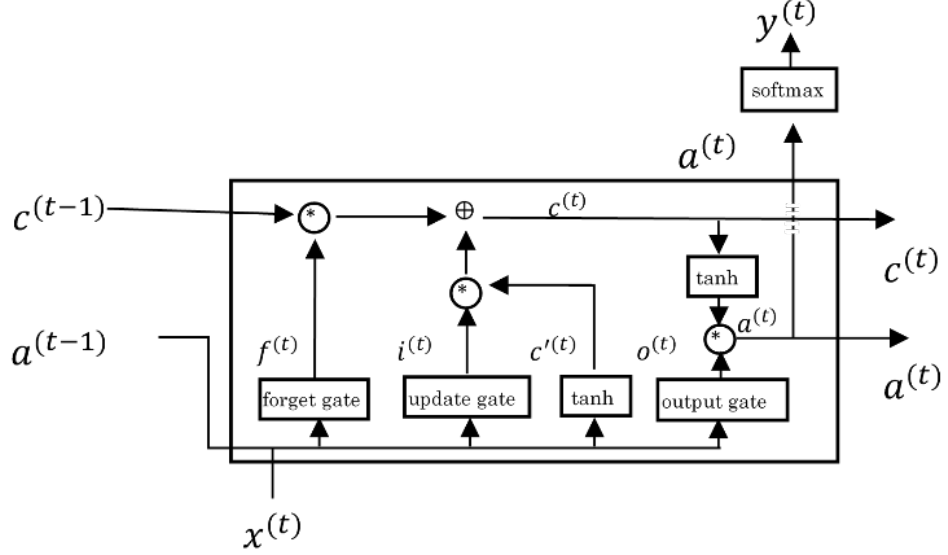


Figure 2.2: LSTM Architecture

LSTM units contain multiplicative gates that enforce constant error flow through the internal states of special units called "memory cells" as shown in figure 2.2 LSTM consists of three gates, namely update gate 2.18, forget gate 2.19, and output gate 2.20, which control the flow of information by manipulating hidden states of the units. As opposed to GRU, LSTMs used separate update and forget gates that are used to compute a proposed value for the hidden state at time t by using the hidden state at time $t - 1$ and the memory cell in the lower hidden layer at time t as shown in equation 2.21. The update and forget gates in LSTMs give the memory cells the option of keeping the old value for the hidden state at $t - 1$ and adding to it the new value for the hidden state at time t . LSTM is more potent than GRU because of its ability to learn long-term correlations in a sequence and is capable of accurately modeling complex multivariate sequences without the need for a pre-specified time window [75]. As such, it becomes more appropriate to consider LSTM units with multiplicative gates that enforce constant error flow through the internal states of individual units called "memory". LSTM also have internal memory that operates like a local variable, allowing them to accumulate state over the input sequence.

$$\text{Input: } c'_t = \tanh(W_{ca}a_{t-1} + W_{cx}x_t + b_c) \quad (2.18)$$

$$\text{Update Gate: } u_t = \sigma(W_{ua}a_{t-1} + W_{ux}x_t + b_u) \quad (2.19)$$

$$\text{Forget Gate: } f_t = \sigma(W_{fa}a_{t-1} + W_{fx}x_t + b_f) \quad (2.20)$$

$$\text{Output Gate: } o_t = \sigma(W_{oa}a_{t-1} + W_{ox}x_t + b_o) \quad (2.21)$$

2.5.3.4 Autoencoder

Autoencoder is another RNN architecture that is used for sequence-to-sequence predictions. As shown in figure 2.3, an autoencoder comprises of the input layer (encoder) that reads the input sequence and transformed it to a fixed-length representation (embedding), and an output layer (decoder) that interprets the internal representation and uses it to predict the output sequence. In this context, RNN is used as an encoder to obtain a fixed dimensional internal representation (embedding) of an input MTS, and then the decoder used this internal representation to generate an output sequence of MTS. The basic idea involves non-linear mapping of the input MTS to a fixed dimensional vector representation through an encoder function f_{enc} as shown in equation 2.22, which is followed by another non-linear mapping of the fixed-dimensional vector to an MTS using decoder function f_{dec} as shown in equation 2.23.

$$z_t = f_{enc}(x_i(t - w + 1, t)) \quad (2.22)$$

$$y'_t = f_{dec}(z_t) \quad (2.23)$$

LSTM networks can be used for both the encoder and decoder. It means one LSTM will be used to read the input sequence, one-time step at a time, to generate the fixed dimensional vector representation, and then use another LSTM to extract the output sequence from that vector. Autoencoders are used for dimensionality reduction that helps in classification and visualization tasks and are also used for learning the hidden representation of time series. As a result of its efficient data encoding in an unsupervised manner, it is also gaining popularity in anomaly and novelty detection problems.

2.5.3.5 Deep Learning-based Anomaly Detection Techniques

Malhotra et al. [103] proposed an anomaly detection method based on stack LSTM. The authors developed a predictive model that was trained using the normal training time series dataset, which is further evaluated to compute error vectors based on its performance on the anomalous test sequence. Anomaly is then defined by setting of an error threshold that is defined using the validation test sequence. A similar approach was also used to detect anomalies in ECG data [32]. They used RNN augmented with LSTM to detect 4 different types of anomalous behavior. In a similar context, Singh [139] explored the use of LSTM for anomaly detection in temporal data. The paper used the same approach in [103] where a prediction model is trained to learn the normal time series pattern and predict future values thereby modeling the prediction error as Gaussian distribution to estimate the anomalous likelihood of an observed future value. In addition, the paper also investigates different ways of maintaining LSTM states and their effect on prediction and detection performance.

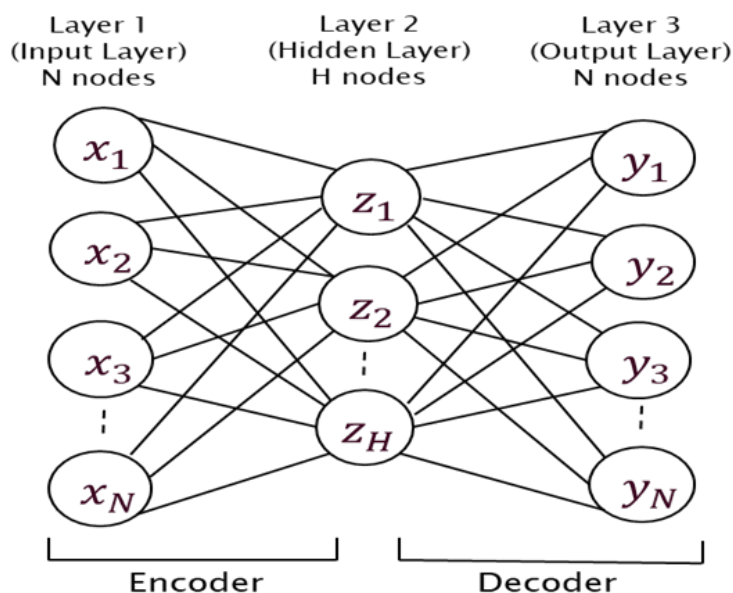


Figure 2.3: Autoencoder Architecture

In a slightly different approach, Bontemps et al, [22] combined LSTM prediction model with a circular array for the detection of collective anomalies in time series data. In contrast to the previous point anomaly detection methods, instead of considering each time step separately, they trained a prediction model that can predict multiple time steps thereby storing the prediction errors from those steps in a circular array. The circular array contains only the prediction errors from a certain number of recent time steps defined by minimum attack time. This detection method faces the problem of identifying collective anomalies using a relative error threshold that is defined using labeled anomalies from the validation test. Instead of using a circular array, Saurav et al [131], proposed another prediction-based anomaly detection method that used the sliding window to handle both the multidimensional and streaming nature of time series. The prediction model is trained incrementally as new data arrives which enable it to adapt to different types of changes (sudden, incremental, gradual, and continuous concept drifts) in the time series data. In addition, the authors used GRU units in the RNN architecture, which are a simplified version of LSTM units. Although, the method used a sliding window for multi-steps ahead prediction and related prediction error for updating model parameters which enable online anomaly detection, the anomaly score of the window is computed as an average of the square of individual points estimated error. This will lead to a false positive result and an increase in the misidentification of anomalous points in real-time where data instance arrives one after the other.

In order to solve this problem, Shipmon et al, [137] proposed another method that

2.5 Review of time series Anomaly Detection Techniques

combined LSTM based prediction model to predict the expected value in time series and Gaussian tail probability rule defined in [2] to estimate the anomaly likelihood of the current state based on the prediction history of the LSTM model. In their paper, they model the distribution of the error values as an indirect probabilistic metric and used it to measure the likelihood that the current state is anomalous. Although the distribution of prediction error is technically not Gaussian in some applications, they model it using a normal distribution which may still result in false positive results in very noisy, unpredictable streams.

In another context, Convolutional Neural Network (CNN) was used as a prediction model to propose an unsupervised anomaly detection method (DeepAnT) for time series data [110]. DeepAnT method consists of two modules. First, the predictor module is responsible for predicting the next timestamp using a window of previous time stamps. The predicted value is then passed to the anomaly detection module that is responsible for tagging a data instance as an anomaly or not. This method also used the same procedure as the previous methods where the anomaly score is computed by the assumption of a normal distribution on prediction error. However, all the above-mentioned methods face two main challenges when applied in most real-life scenarios, that involve complex systems. First, there are often external factors or variables, which are not captured by sensors that lead to unpredictable time series. Secondly, an exact parametric distribution is often not directly relevant in some applications and the assumption of any distribution will lead to false anomaly alerts due to high prediction variance.

To solve the unpredictable nature of the time series, an Autoencoder is used for learning hidden representation and extraction of relevant features from the time series data. Autoencoder involves two parts, Encoder, and Decoder. The Encoder learned how to encode or compress the input data while Decoder reconstructs the encoded data back to the representation that is close to the input data. Because of its efficient data encoding in an unsupervised manner, it is also gaining popularity in anomaly and novelty detection problems. Malhotra et al [102], proposed an LSTM-based Encoder-Decoder for anomaly detection in multivariate time series data. In the paper, an Encoder-Decoder model learns to capture the normal behavior of the machine by reconstructing the normal time series in an unsupervised manner. Since the model is trained only on time series corresponding to normal behavior, it is expected to perform well while reconstructing normal time series and poorly on abnormal time series. The reconstruction error is then used in computing an anomaly score that is used to identify anomalies in the time series data. Like the previous methods, a normal distribution is assumed on reconstruction error, which enhances the computation of the anomaly score. In a similar context, Schreyer et al, [134] also used deep autoencoders to detect anomalies in large-scale accounting data in the area of fraud detection.

A different approach is proposed in [5] that combined autoencoding with a clustering method for unsupervised novelty detection. The authors in their approach compute

2.5 Review of time series Anomaly Detection Techniques

an error threshold from the Autoencoder model and pass it to the density-based clustering method. Clustering is performed on the compressed data to detect novelty groups as points with low density. In a similar approach, Zong et al, [173] combine Autoencoder with Gaussian Mixture Model (GMM) for unsupervised anomaly detection. The Autoencoder is used for dimensionality reduction where the low-dimensional representation together with reconstruction error is passed to GMM for density estimation and identification of anomalies. Despite the performance of these methods and their unsupervised learning approach, they face a similar challenge of assuming a Gaussian distribution on the reconstruction error which is either ranked or threshold for anomaly detection. However, as mentioned earlier an exact parametric distribution is often not directly relevant in some applications, and the assumption of any distribution will lead to false anomaly alerts due to high reconstruction variance.

In an attempt to mitigate the false positive problem, Hundman et al [80] introduced a pruning procedure and learned from the history of labeled anomalies. The pruning procedure will limit evaluation to only maximum errors and helps in ensure the anomalous sequence are not the result of regular noise within the stream. Although, these procedures have played an important role in improving the precision of the detection method, limiting evaluation to only maximum errors thereby removing or reclassifying minimum error as nominal will lead to missed detections. To solve this problem, uncertainty is considered in the deep learning-based predictions for anomaly detection.

Various approaches have been developed that used uncertainty in deep neural networks for anomaly detection. They range from Bayesian approach [114] to evidential deep learning [6] and interpreting dropout as performing variational inference [54, 92, 172]. Bayesian NNs are used for estimating both epistemic and aleatoric uncertainties which place probabilistic priors over network weights and use sampling to approximate output variance [112]. However, Bayesian NNs face several limitations, including the intractability of directly inferring the posterior distribution of the weights given data, the requirement and computational expense of sampling during inference, and the question of how to choose a weight prior.

In contrast, evidential deep learning formulates learning as an evidence acquisition process to estimate a continuous target associated with evidence to learn both aleatoric and epistemic uncertainties [6]. Evidential deep learning is developed by incorporating the evidential theory into deep neural networks where every training example adds support to a learned higher-order, evidential distribution. Sampling from this distribution yields instances of lower-order likelihood functions from which the data was drawn. By training a neural network to output the hyper-parameters of the higher-order evidential distribution, a grounded representation of both epistemic and aleatoric uncertainty can then be learned without the need for sampling. Instead of placing priors on network weights, as is done in Bayesian NNs, evidential approaches place priors directly over the likelihood function. Recently, deep evidential learning has been targeted towards anomaly detection problems where an RNN-based evidential model was

2.5 Review of time series Anomaly Detection Techniques

designed to predict future trajectories associated with both data and model uncertainties that are used to detect anomalous trajectories in maritime domain [141]. In an alternative approach, deep evidential learning is used for human actions recognition where the Deep Evidential Action Recognition (DEAR) method is developed for open set action recognition task [11]. The referenced paper used the learned evidence to quantify the prediction uncertainty for diverse human actions which enable the model to identify unknown actions where unknown actions will have higher uncertainties.

On the other hand, some probabilistic deep learning models used other methods to compute Prediction Interval (PI) that quantifies the level of uncertainty associated with the point forecasts. David et al, [73] proposed an anomaly detection method that used autoregression model and its prediction interval to identify anomalies in environmental sensor streams. The authors used Student's t-distribution to calculate the prediction interval that accounts for uncertainty in both input data and model parameters. They classified a data point as anomalous when it falls outside a given prediction interval. In contrast, Lingxue and Nikolay [172] argued that the measurement of prediction uncertainty does not depend on only input data and model parameters but rather combines with model misspecification and inherent noise. They achieved this by using Monte Carlo dropout to estimate both data and model uncertainties. The estimated uncertainties are then used to construct a prediction interval using z th standard normal where values outside the constructed interval are labeled as anomalies. In a similar context, Legrand et al, [92] investigated how the inclusion of uncertainty in the computation of anomaly score improves the performance of anomaly detection using autoencoder. They do that by developing a new anomaly score function that is weighted with uncertainty as opposed to the Bayesian score function introduced in [172]. This approach is computationally demanding and imposes strong requirements.

In order to minimize the computational requirements, Reunanen et al [126] proposed another unsupervised anomaly detection method that combines Autoencoder and Logistic Regression for outlier detection and prediction in sensor data streams. The Autoencoder reconstructs the input data and produces a hidden representation of the input that can be used to create the required labels for Logistic Regression to classify anomalous points. The outlier is identified as extreme values that exceed a limit of three standard deviations defined using Chebyshev's inequality or any deviation in the correlation of data features. Although no assumption of the distribution of the data is required the method assumes the descriptive statistics (γ and μ) of the unknown normal values to be initially defined. It is also assumed that the feature value of the initial data point has non-zero variance often which the scaling limits for outlier detection cannot be defined. In contrast, a probabilistic forecasting model that returns a full conditional distribution was proposed in [129]. The authors introduced probabilistic forecasting using an autoencoder model that directly outputs parameters of Negative Binomial and Gaussian likelihoods for real-valued time series. This shows the probabilistic forecasts were also achieved by the assumption of a Gaussian distribution on the prediction er-

ror. Most of these methods used mean square error as the loss function which is always minimized to the normal distribution in their regression task. However, an exact parametric distribution is often not relevant in applications which will lead to false anomaly alerts due to high prediction variance.

In contrast, Quantile Regression models are trained to minimize quantile loss which is useful in areas of application where it is difficult to define a parametric distribution on the residuals [3]. Quantile regression aims at forecasting conditional quantiles that are given an input sequence and can produce a probabilistic forecast without making any distributional assumption. An attempt was made to use quantile regression for anomaly detection [104] but it is limited to the use of quantile interval to identify only uncertainties in the data. This limitation led to the research investigation of whether the estimation of uncertainty in prediction using quantile regression will increase the prediction performance and reduce false anomaly alerts. This results in the development of an improved quantile regression anomaly detection model (DQR-AD) in this thesis. Such a model is most welcome in applications requesting better-informed decisions and mitigating false anomaly alerts.

2.6 Anomaly Detection for Stream Data

The previous section reviewed anomaly detection techniques for time series. Although there has been extensive work on anomaly detection for time series, most of the techniques are offline which work on stationary time series. However, this is unsuitable for real-time streaming applications that continuously generate a large volume of data streams that may change over time. In such systems, storage of data that can be scanned multiple times for model predictions is difficult due to memory requirements and high processing demand. Even if it is possible, the stored data may not be a good representation of normal data in the future due to evolving nature of the data stream. As such, the models trained on only the previous data collected will suffer from any changes in the distribution of the data. These changes can be in two forms: 1) Concept evolution where the new normal concept will appear over time, which degrades the predictive performance of the model. 2) Concept Transformation where a normal concept may transform gradually or suddenly into an anomalous concept and vice versa. In this case, model-based anomaly detection must be able to adapt to these types of changes in order to cope with the dynamic nature of data streams. Most classical methods create linear functions on transformations of the actual time series in order to make future time step predictions. Unlike these methods, recent models have applied deep neural networks with inherently fewer restrictions on the input, while being able to explore underlying nonlinear relationships in the data. Some time series, like in the case of financial data, are inherently non-stationary, i.e. the statistics of the data changes over time. This makes it difficult to model and challenging to naively apply neural network

function approximation (a neural network typically learns a function that maps inputs or observations to target or predictions) which may not be suitable to cope with such non-stationarity and mismatch in data distributions.

Alternative to neural network-based models, stochastic process-based models like the Gaussian process (GP) may also be a good choice. They make use of prior knowledge and learn a distribution over functions rather than a single-function approximation. This makes GP models Bayesian as they involve constructing a prior distribution (here over functions directly rather than over parameters) and updating this distribution by conditioning on the actual data. This is particularly useful for financial data because of its volatile nature. The price of a financial asset is sensitive to many external or internal events like policy reforms, intrinsic turbulence in the market, news sentiments and natural disasters. Now with a distribution of functions, the risk or uncertainty can be embedded in the standard deviation of the model's predictions. This makes us better informed while making decisions and forming strategies based on these predictions. Due to the highly non-stationary nature of financial time series, the model we create ideally must also evolve with time as the relationship between the past and future is unlikely to remain stationary. This results in the need for online anomaly detection algorithms that can detect anomalies in real-time. Recently, researchers have proposed some new methods for online anomaly detection [43]. These methods generally can be categorized into two ways:

First is to build a new model using a single incremental learning algorithm [109, 117] and online ensemble learning theory [7, 43] that trains multiple individual models for different parts of the data streams. Although incremental learning methods that use a single model have advantages over ensemble classifiers in terms of speed and low computational cost, they suffer many challenges when the concept changes suddenly and rapidly. As a result ensemble classifiers are used due to their utmost predictive accuracy and stability-elasticity property. This property makes it easy for them to incorporate new data into the model, by training and adding new members to the ensemble, and naturally, forget irrelevant knowledge by removing the old members from the ensemble [44]. Research in the literature has proved the performance of online ensemble learning in handling concept drift in data streams [120]. Two main approaches are used in online ensemble learning methods which include the Chunk-based approach (that involves adding a new model which is trained on a recently arrived chunk of data) and the Online-based ensemble approach (that involves updating the base classifier in the ensemble). In an effort to verify the performance of online-based ensemble methods with their Chunk-base counterpart, in chapter 3, we carried out a preliminary study and experimentally compare their performance to concept drift and class imbalance ratio. The result of the experiments was published in a conference [147].

The second involves modification of the traditional method, i.e., training the initial detector based on all historical datasets and updating it based on the new coming instance of data. Recently, several anomaly detection techniques were proposed using

the first strategy where an online or incremental learning approach was used to fully label normal data [10, 128, 130]. Similar approaches are proposed that consider the non-availability of class labels in the data stream and applied unsupervised learning technique to detect anomalies in real-time [2, 16, 94]. In addition, a data-driven modeling approach was proposed in [73] which can detect anomalies in real-time without requiring any pre-classified input data. In this method, the data point is an anomaly based on whether or not it falls outside the prediction interval. A more recent approach that used Recurrent Neural Network to detect anomalies in real-time was proposed in [131]. This approach used instances of data that arrive continuously and trained the model incrementally thereby adapting to the changes that may occur in the data distribution. The authors used model prediction error to determine when to update the model based on the changes that occur in the data and whether those changes are anomalies or not. The authors used a window of time series sequence for the previous time steps to predict the estimated sequence of the next time step using the RNN model. The predicted result is compared with the actual observed sequence and prediction error will be used for anomaly score computation thereby updating next step RNN model using Back Propagation Through Time (BPTT). The prediction error of the model can be monitored only when the labels of data are available after classification. The aforementioned approach works under such assumption and will face a problem when labels are not available as the case may be in many real-world streaming applications. The challenge remains to find out whether the incoming data distribution matches the normal data used for training the model.

In order to solve the above problem, another approach is proposed in [122] that used the One Class Principal Component Classification (OC-PCC) method to develop a model that was trained based on the normal sensor data and used the model to detect anomalies in incoming data as any deviation from the normal concept learned by the model. This technique involves three phases: First, the training phase uses the normal data distribution to develop a normal reference model that will be used in real-time anomaly detection. The second phase serves as the detection phase where the model trained in the first phase is used to classify the new instances that arrive as anomalies or not using dissimilarity measures. The learning in this phase is done using an incremental or online fashion. The third phase is the adaptation phase where the approach adapts to the concept drift by retraining the classifier with the new concepts and producing a new reference model that can be used to update the old one based on certain criteria. Although this approach handles the problem of the unavailability of class labels and achieves real-time anomaly detection, it does this by retraining and updating the model to adapt to the dynamic changes in the environment. As such it suffers when the concept changes rapidly and suddenly.

Therefore a second strategy using an ensemble model can be adopted where a new model trains whenever changes are detected in data streams. An ensemble technique was proposed in the literature that used ensemble classifiers in combination with an

unsupervised anomaly detection method [171]. The authors used sequential ensemble learning where the classifiers are dependent on each other, and the execution of one classifier depends on the result of the execution of another classifier in the ensemble. However, such dependencies may degrade the performance of some classifiers and result in bias in the final prediction result. As such, other solutions are needed that used independent ensemble classifiers in the anomaly detection process. To achieve this, Tan et al. proposed an approach that used an ensemble of one-class decision trees called HS-Trees for real-time anomaly detection for evolving data streams [148]. The method is fast and efficient in terms of computational complexity because real data is not used in training the initial detector and it doesn't require model modification when processing streaming data. Despite these advantages, it suffers from continuous update whether changes are detected or not which reduce the real-time nature of the system.

A more recent and real-time anomaly detection approach that used one-class and ensemble approach for anomaly detection is given in [43] where a hyper-grid based ensemble classification method was used. This approach work by building a hyper-grid structure based on a window of training datasets and incoming instance to be characterized as anomalous or not through the region they are mapped into using a hyper-cube. A similar approach in [59] used one-class support vector machines (OCSVMs) to develop a window-based unsupervised adaptive anomaly detection method called Ensemble-based Self-Adaptive OCSVM (ESA-OCSVM). This approach is made of two phases which include; the change detection phase which detects both concept evolution and transformation using a sequential change detection approach and the learning phase where an ensemble of models was developed that can cope with both concept evolution and transformation in incremental or online passion. It does this by selecting a related set of instances when change is detected and a proper model from the ensemble or creating a new model for anomaly detection. Although the aforementioned solutions adapt to the changes in the data through the use of unsupervised ensemble models, because of the space complexity required by hyper-grid and support vector machines, they all face a serious challenge when dealing with high dimensional datasets. Similarly, most of the existing methods have challenges in defining the metric used in estimating when changes occur in the data. As such, in this thesis we consider the combination of feature extraction and an unsupervised prediction-based learning approach to develop an algorithm that detects anomalies in real-time and adapts to the dynamic changes that occur in time series.

2.7 Summary

This chapter introduced the problem concept for anomaly detection in time series. It starts with providing a theoretical background of important concepts that include time series, data streams, concept drift, prediction uncertainty, and quantile regression. This

is followed by a review of different concepts of formulating anomaly detection problems in time series. Finally, a review of anomaly detection techniques is given. The review techniques include both classical, deep learning-based, and online anomaly detection techniques. However, most of the deep learning-based anomaly detection methods compute anomaly scores by assuming a distribution usually of Gaussian type on the prediction error, which is either ranked or threshold to label data instances as anomalous or not. But, an exact parametric distribution is often not directly relevant in some applications, and the assumption of any distribution will lead to false anomaly alerts due to high prediction variance. It is also difficult to select an appropriate threshold that will differentiate anomalies with noise. As such, this thesis proposes a new anomaly detection method that computes uncertainty in estimates using quantile regression and used the quantile interval instead of a fixed threshold to identify anomalies. In addition, the volume and speed at which the time series data arrive demonstrate the need for adaptation of concept drift in the data. Although there are many online anomaly detection methods proposed in the literature, they faced challenges in defining the metric used in estimating when changes occur in the data. In this case, we consider the combination of feature extraction and an unsupervised prediction-based learning approach to develop an algorithm that detects anomalies in real-time and adapts to the dynamic changes that occur in time series. To the best of our knowledge, no such approach exists in the literature.

Comparison of Ensemble Classifiers on Drifting Data Streams

3.1 Introduction

As a result of preliminary study conducted to investigate the challenges faced by online learning algorithms when there is concept drift in the data. It was discovered that online ensemble methods are the most popular approach used in handling concept drift due to their stability-elasticity property that makes it easy for them to incorporate new data into the model, by training and adding new members to the ensemble, and naturally, forget irrelevant knowledge by removing the old members from the ensemble. As such this chapter focus on the experimental analysis of concept drift and its corresponding challenges on online ensemble classifiers. Literature shows two main ensemble approaches that are used for this purpose. They include a Chunk-based approach (which involves adding a new classifier that is trained on a recently arrived chunk of data) and an Online-based ensemble approach (which involves updating the base classifier in the ensemble). The experiment is conducted for comparison and performance evaluation of these ensemble classifiers using both real and synthetic data streams. The goal of this experiment is to compare and evaluate the performance of these algorithms toward a different type of concept drift in data streams. To carry out this task, we start with the literature review of existing methods and their corresponding challenges in adapting to concept drift. We then design an experiment to compare their performances using both real and synthetic datasets.

3.2 Literature Review

This section provides a review of online learning algorithms with respect to sensor streams or time series data. The section is structured to provide an understanding of the research area thereby giving references to the relevant information obtained from the literature. It starts with online learning techniques used in data stream mining and is then followed by concept drift detection.

3.2.1 Online Learning

Online learning also called Incremental learning is a classification technique where data arrives in sequential order and each piece of data is used to update the learning models once and forever. It involves processing incoming examples sequentially in such a way that the trained classifier is as accurate as the classifier that was trained on the whole dataset at once. Several online learning algorithms were proposed in the literature that either adapt to changes implicitly or explicitly using a change detector. The most popular algorithm that adapts to changes explicitly is Adaptive Windowing (ADWIN) [17] which keeps an adapting sliding window that grows when there is no change in the data using statistical mean. Whenever changes occur the window is split into old and new parts where the old part will be discarded and a new part that involves new changes keep growing. Online learning algorithms are developed either using a single classifier or an ensemble of based classifiers. Single classifier online learning algorithms such as Very Fast Decision Tree (VFDT) [45], Concept-adapting Very Fast Decision Tree (CVFDT) [79], adapt to the changes in the data by applying to adapt windowing to the decision tree learning process. These decision tree algorithms adopt an incremental learning approach by growing alternative branches and monitoring their performance using adapting windowing. When the alternative branches become more accurate than the main branches, they replace the main branches. However, VFDT and CVFDT use static windowing which faces a great challenge for learning in non-stationary data streams. An alternative to VFDT and CVFDT is Incremental On-Line Information Network (IOLIN) [38] that used dynamic windowing to generate a more accurate model. Similarly, an Extreme Learning Machine (ELM) [167] was used for system identification in the non-stationary environment. Online learning poses many challenges that include a large volume of data in real-time, concept drifts, temporal dependencies, limited class labels, class imbalance, noise, and limited resources (processing time and memory) requirements, etc. With an increase in real-world applications of stream learning algorithms, concept drift and class imbalance becomes the major challenges for this area and received major research attention.

3.2.2 Concept Drift Detection

In literature, three main approaches are used for handling concept drift in the data stream, which includes: window-based approaches, weight-based approaches, and ensemble classifiers [47, 56, 120]. These methods are generally categorized into active and passive approaches [44]. Active approaches such as drift detectors specifically aim at detecting concept drift and use mechanism that detects the presence of the change. There are many drift detection methods used for detecting concept drift which is intensively reviewed in [60]. One of the key challenges faced by active approaches is failing to detect the change or wrongly detecting non-existing change (false alarm). The passive approach manages the loopholes in active approaches through the use of an adaptation mechanism that allows the model to be updated every time new data arrives regardless of whether concept drift occurs or not. Passive approaches include models that are developed based on updating a single classifier and models that involve adding, removing, and modification of ensemble members.

Although passive approaches that involve updating a single classifier have advantages over ensemble classifiers in terms of speed and low computational cost, they suffer many challenges when the concept changes suddenly and rapidly. As a result ensemble classifiers are used due to their utmost predictive accuracy and stability-elasticity property. This property makes it easy for them to incorporate new data into the model, by training and adding new members to the ensemble, and naturally, forget irrelevant knowledge by removing the old members from the ensemble [44]. In today's literature, ensemble learners remain the most popular approach to handling concept drift in data stream mining. Two main approaches involve which include the Chunk-based approach (that involves adding a new classifier that is trained on a recently arrived chunk of data) and the Online-based ensemble approach (that involves updating the base classifier in the ensemble) [120]. In an effort to verify the performance of online-based ensemble methods with their Chunk-base counterpart, we design an experiment to compare their performance on concept drift adaptation and class imbalance ratio.

3.3 Datasets Description

In this experiment, we used both artificial and real datasets with different class imbalance ratios and were affected by both sudden and gradual concept drifts. Five real data streams are used which are commonly used as benchmark datasets. The real data streams include Airlines (Air) and Electricity (Elect) as moderately balanced datasets, and KDDCup and PAKDD as imbalance datasets [27]. Similarly, MOA framework [20] was used to simulate nine (9) artificial datasets with different types of concept drifts and class imbalance ratios. It is worth mentioning that 100,000 instances are

generated for all artificial datasets with 3 and 5 attributes. We introduce sudden and gradual changes appearing at every 25,000 instances and 10 *percent* of class noise to make the learning process more challenging. The details characteristics of the datasets are given in Table 3.1.

Datasets	Instances	Attributes	Class Ration	Noise	Drift Type
SEA _{ND}	100k	3	-	10%	None
SEA _{SD}	100k	3	-	10%	Sudden Drift
HYP _{SD}	100k	3	-	10%	Gradual Drift
Imb1	100k	5	1:1	10%	None
Imb2	100k	5	1:10	10%	None
Imb3	100k	5	1:20	10%	None
Imb4	100k	5	1:100	10%	None
SEA _{SC}	100k	3	1:1/1:100/1:10/1:1	10%	Sudden and Virtual
HYP _{GC}	100k	3	1:1 to 1:100	10%	Gradual Drift
MinMaj	100k	5	1:20/20:1	10%	Gradual Drift
Air	539k	7	24:30	-	Unknown
Elect	45k	8	19:26	-	Unknown
KDDCup	494k	41	1:4	-	Unknown
PAKDD	50k	30	1:4	-	Unknown

Table 3.1: Datasets and their characteristics

3.4 Experiments

In order to evaluate the performance of online-based ensemble methods with their Chunk-base counterpart, we design an experiment that compares their performance on concept drift adaptation and class imbalance ratio. The experiment compares and evaluates the performance of 10 ensemble classifiers for data stream classification as shown in Table 3.2.

3.4.1 Experimental Settings

To evaluate the performance of the ensemble approaches, a methodology similar to that described in [26, 27] was used where a prequential evaluation was carried out with an evaluation method called ImbalancePerformanceEvaluator. Prequential evaluation is the most commonly used evaluation method in stream learning, where instances are first used to test and then train a single model. The performance of the ensemble classifiers was measured using prequential AUC that is more robust to concept drift. The results were obtained using a sliding window of 1000 instances and performance

3.4 Experiments

	Approches	Description
Chunk-based	AWE[160]	Accuracy Weighted Ensemble
	Learn++.NSE [46]	Incremental Learning for Non-Stationary Environment
	AUE [24]	Accuracy Updated Ensemble
	RCD [60]	Recurrent Concept Drift
	EB [119]	Ensemble Building
Online Ensemble	DWM [89]	Dynamic Weighted Majority
	OAUE [25]	Online Accuracy Updated Ensemble
	LB [18]	Leveraging Bagging
	OzaBagASHT [19]	Bagging with Adaptive-Size Hoeffding Tree
	OzaBagADWIN [17]	Bagging with Adaptive Windowing

Table 3.2: Ensemble Classifiers for non-stationary data streams

sampling frequencies of 100 instances at a time. All the ensemble classifiers and evaluation methods used were implemented in Java as part of MOA. For ensemble classifiers with decision tree structure, we used 10 Very Fast Decision Trees (VFDT) [45] as base learners with Adaptive Naive Bayes leaf predictions with a grace period $n_{\min} = 100$, split confidence $\delta = 0.01$, and tie-threshold $\phi = 0.05$ [27]. Similarly, an error-based pruning strategy was used to Learn++.NSE while for RCD, we used Drift Detection Method (DDM) [55] as drift detection method. A chunk size of 500 was chosen for all chunk-based approaches and lastly, we maintain default settings for the rest of the parameters on all ensemble classifiers.

3.4.2 Experimental Results

Table 3.3 shows the average prequential AUC, individual rank, and average ranks of all ensemble classifiers under the used datasets. The result of predictive AUC in Table

Datasets	AWE	Learn++.NSE	AUE	RCD	EB	DMW	QAUE	LB	ASHT	ADWIN
SEAS _D	0.87(6)	0.86(9.5)	0.87(6)	0.87(6)	0.86(9.5)	0.87(6)	0.87(6)	0.88(2)	0.88(2)	0.88(2)
SEAS _D	0.74(7.5)	0.73(10)	0.74(7.5)	0.76(5)	0.74(7.5)	0.74(7.5)	0.78(4)	0.87(1)	0.81(3)	0.83(2)
HYP _{GD}	0.78(7.5)	0.74(10)	0.78(7.5)	0.80(5)	0.76(9)	0.84(2.5)	0.79(6)	0.88(1)	0.82(4)	0.84(2.5)
Imb1	0.99(9)	0.99(9)	0.99(9)	0.99(9)	0.99(9)	1.00(2.5)	0.99(9)	1.00(2.5)	1.00(2.5)	1.00(2.5)
Imb2	0.99(2)	0.69(10)	0.99(2)	0.96(6)	0.86(9)	0.90(8)	0.99(2)	0.94(7)	0.98(4.5)	0.98(4.5)
Imb3	0.99(4.5)	0.97(9)	0.99(4.5)	0.98(7.5)	0.96(10)	0.98(7.5)	0.99(4.5)	0.99(4.5)	1.00(1.5)	1.00(1.5)
Imb4	0.99(6)	0.98(9)	0.99(6)	0.99(6)	0.97(10)	0.99(6)	0.99(6)	1.00(1.5)	1.00(1.5)	1.00(1.5)
SEAS _C	0.78(4.5)	0.76(9)	0.78(4.5)	0.77(8)	0.75(10)	0.78(4.5)	0.78(4.5)	0.79(1)	0.78(4.5)	0.78(4.5)
HYP _{GC}	0.66(4)	0.62(8)	0.66(4)	0.66(4)	0.64(7)	0.66(4)	0.66(4)	0.67(1)	0.58(10)	0.59(9)
MinMaj	0.99(5)	0.85(10)	0.99(5)	0.99(5)	0.96(9)	0.98(8)	0.99(5)	0.99(5)	1.00(1.5)	1.00(1.5)
Air	0.67(2.5)	0.57(9)	0.67(2.5)	0.64(5)	0.56(10)	0.62(7)	0.63(6)	0.60(8)	0.68(1)	0.65(4)
Elect	0.84(7.5)	0.73(9)	0.84(7.5)	0.87(6)	0.56(10)	0.94(3)	0.94(3)	0.96(1)	0.93(5)	0.94(3)
KDDCup	0.96(6)	0.94(8)	0.92(10)	0.99(4.5)	0.95(7)	1.00(2)	0.93(9)	1.00(2)	1.00(2)	0.99(4.5)
PAKDD	0.61(5)	0.50(10)	0.61(5)	0.60(7)	0.56(8)	0.51(9)	0.62(3)	0.63(1.5)	0.61(5)	0.63(1.5)
Average rank	5.50	7.89	5.79	6.00	8.93	5.54	5.14	2.79	3.43	3.18

Table 3.3: Average Prequential AUC

3.3 shows the following:

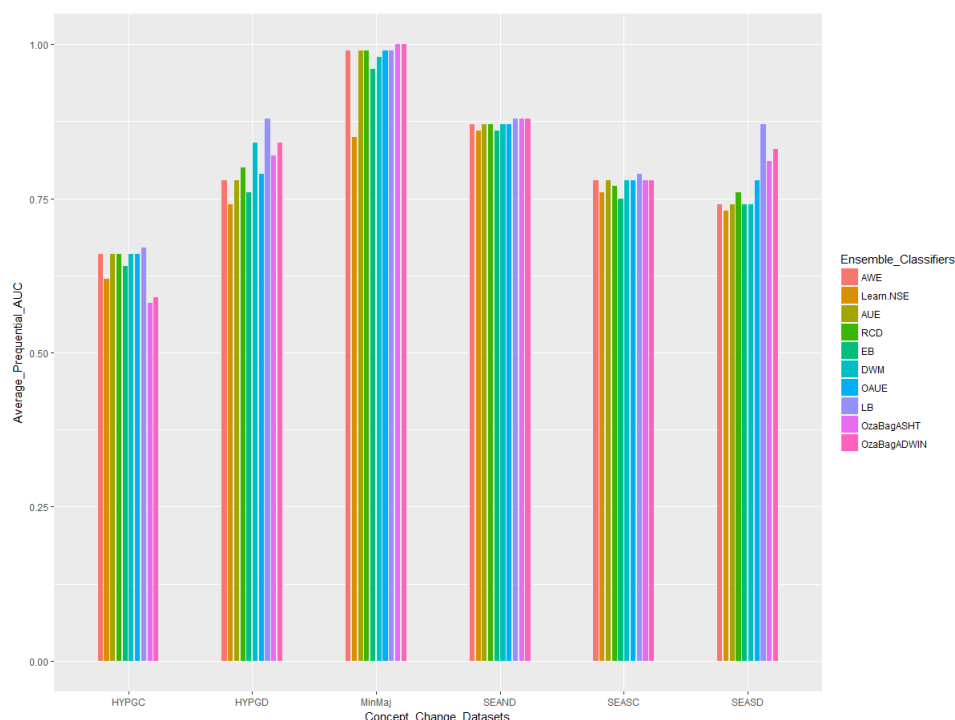


Figure 3.1: Prequential AUC for Concept change datasets

- LB shows the highest performance with the lowest average rank for datasets with artificial and real concept drift (both sudden and gradual). Learn⁺⁺.NSE and EB have the highest average rank that shows the worst performance in all cases as a result of their difficulty in detecting changes with instances arriving one at a time as shown in Fig. 6.
- For datasets with different class imbalance ratios, almost all ensemble classifiers have similar performance unless for the Imb2 dataset where there is variation in performance as shown in Fig. 3.3. This result indicates the power of AUC as an evaluation metric for imbalanced datasets and the ability of the classifiers to cope with changes in class ratio.
- Similarly, for real datasets, LB, OzaBagASHT, and OzaBagADWIN outperform the rest of the classifiers with Learn⁺⁺.NSE and EB have the lowest performance as shown in Fig. 8.

In order to identify whether the ensemble classifiers perform differently based on the null hypothesis, Friedman’s non-parametric statistical test and Nemenyi’s post hoc test were used [42]. The null hypothesis that the classifiers have similar performance with $X < 0.00001$ and Friedman test value = 87.45 at $\alpha = 0.05$ level of significance was

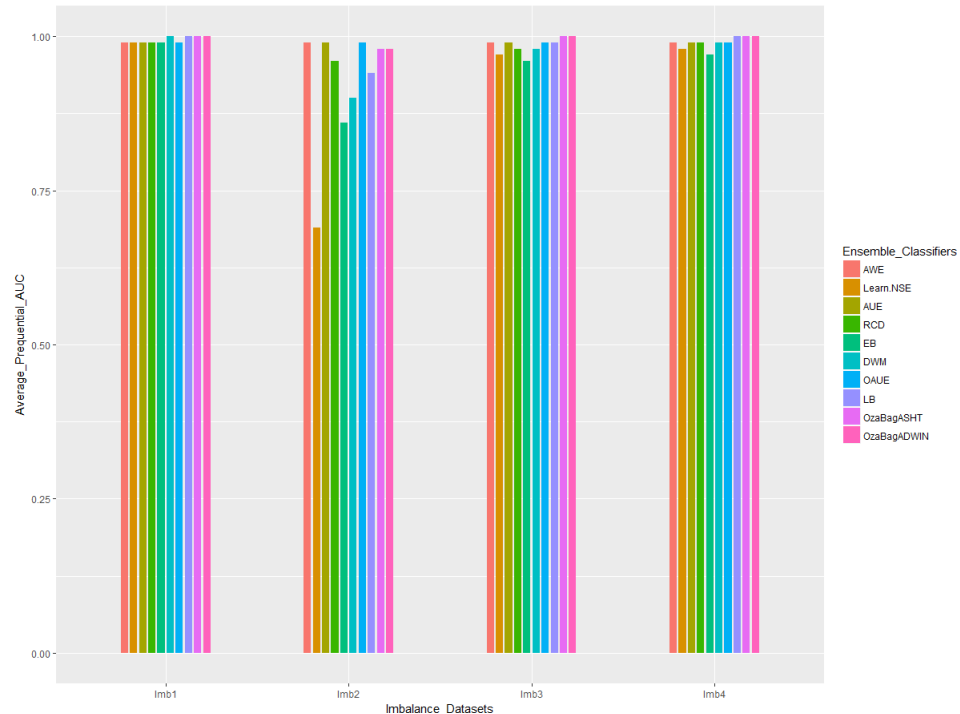


Figure 3.2: Prequential AUC for Imbalance datasets

rejected. In order to verify which ensemble classifiers perform better than the others, the critical difference was calculated as $CD = 3.62$. Fig. 3.4 represents the results of the Nemenyi test by grouping ensemble classifiers that are not significantly different. As can be observed from figure 9, the tree-based online ensemble classifiers (i.e. LB, OzaBagADWIN, and OzaBagASHT) show a significant statistical difference from two chunk-based approaches (Learn⁺⁺.NSE and EB) with RCD, AUE, DWM, OAUE, and AWE group at the middle with values below the critical difference but close to it when compared with other groups of the classifier. This shows online ensemble approaches particularly decision tree-based ensemble performs better than chunk-based approaches for data streams with concept drift and class imbalance problems.

3.5 Summary

This chapter demonstrates a preliminary study conducted to investigate the challenges faced by online learning algorithms when there is concept drift in the data. Its focus is on the experimental analysis of concept drift and its corresponding challenges on online ensemble classifiers. The experiment compared Ten (10) ensemble classifiers and studied their performance in the presence of concept drift and class imbalance, using

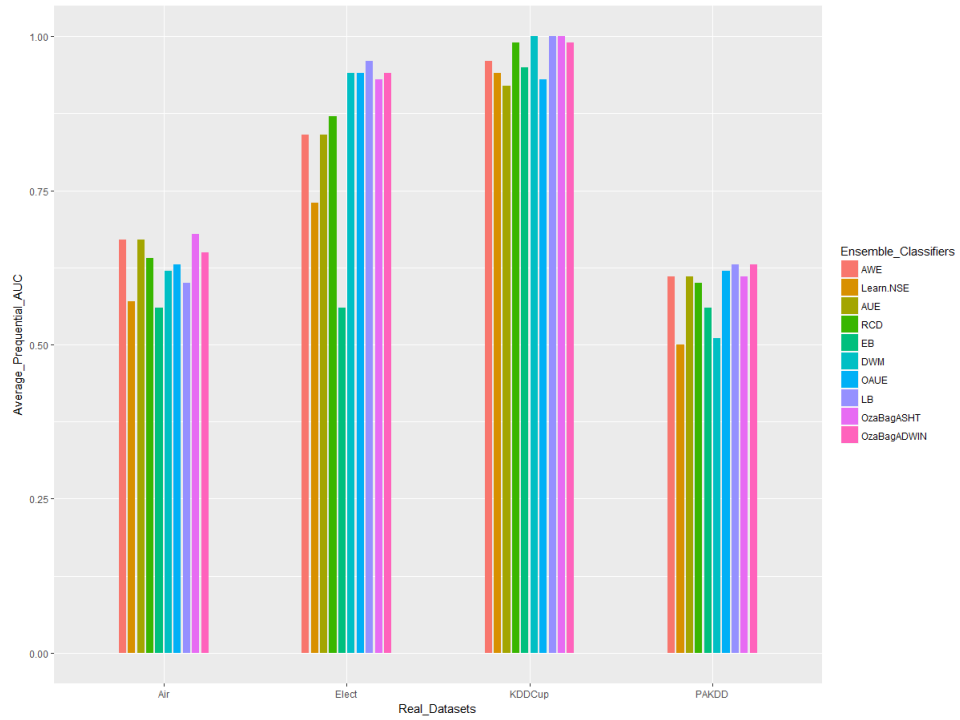


Figure 3.3: Prequential AUC for real datasets

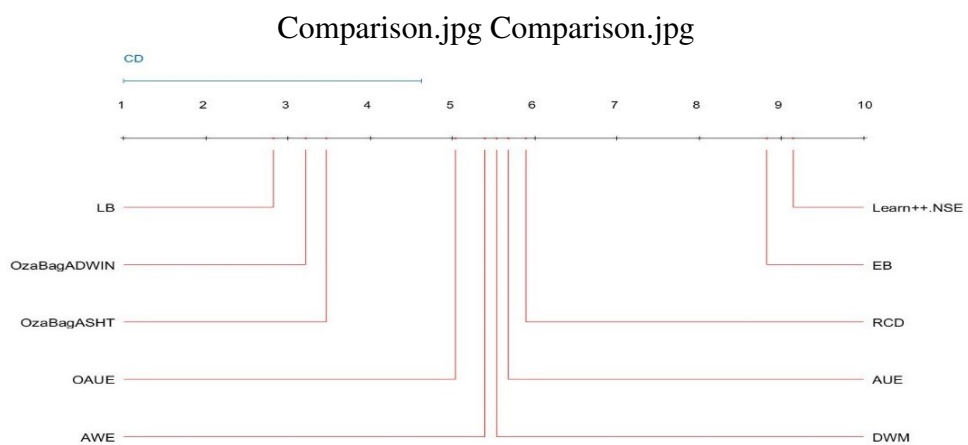


Figure 3.4: AUC comparison of ensemble classifiers with Nemenyi test

artificial and real datasets. In the overall consideration of datasets, LB is the best classifier in terms AUC having the lowest average rank, while EB and Learn⁺⁺.NSE, has the lowest performance. The result also shows online ensemble approaches perform better than chunk-based approaches and specifically, tree-based ensemble learners have the highest performance for learning in a non-stationary streaming environment with concept drift and class imbalance problems.

Unsupervised Anomaly Detection Method for Multivariate time series

4.1 Introduction

Recent advancements in deep learning methods applications to big data collections open also opportunities to study their applicability to anomaly detection [28]. These methods used sequential models, and their performance in sequence analysis reported in multimedia applications [35, 64, 65, 154] enable them to learn the hierarchical discriminatory features and time series temporal nature. In addition, for each time series point, an anomaly score is calculated which helps in handling the type of anomaly detection problem. Deep learning-based anomaly detection techniques using Long Short-Term Memory (LSTM) [4, 97, 103] and other forms of Recurrent Neural Network (RNN) [85, 131], Convolutional Neural Network (CNN) [110], and Autoencoder [5, 96, 99, 102, 126, 134, 173] demonstrate higher performance over the previously mentioned prediction-base anomaly detection techniques. Despite their performance and unsupervised learning approach, they compute the anomaly score by assuming a distribution usually of Gaussian type on the prediction error, which is either ranked or threshold to label data instances as anomalous or not. However, an exact parametric distribution is often not directly relevant in some applications, and the assumption of any distribution will lead to false anomaly alerts due to high prediction variance. When outliers exist in the data, they distort and skew means and variance toward them which affects the detection performance. This direction then leads to the use of uncertainty in deep learning for anomaly detection in time series data.

The goal of this chapter is to answer RQ1 defined in section 1.1 by exploring the use of uncertainty and prediction interval to improve prediction performance and reduce false positive rates in anomaly detection. To carry out this task, we start by reviewing existing research on anomaly detection methods that used prediction intervals

to address uncertainty in time series prediction and reduce false positive alerts. We then design an unsupervised model called Deep Quantile Regression Anomaly Detection (DQR-AD) for anomaly detection in multivariate time series. We developed three algorithms for each component of the model that includes time series segmentation, prediction, and anomaly detection. The algorithms are implemented and experiments are conducted using both real and synthetic datasets to evaluate the algorithms.

4.2 Literature Review

In current literature, various approaches have been developed that used uncertainty in deep neural networks for anomaly detection. They range from Bayesian approach [114] to evidential deep learning [6] and interpreting dropout as performing variational inference [54, 92, 172]. Bayesian NNs are used for estimating both epistemic and aleatoric uncertainties which place probabilistic priors over network weights and use sampling to approximate output variance [112]. However, Bayesian NNs face several limitations, including the intractability of directly inferring the posterior distribution of the weights given data, the requirement and computational expense of sampling during inference, and the question of how to choose a weight prior.

In contrast, evidential deep learning formulates learning as an evidence acquisition process to estimate a continuous target associated with evidence to learn both aleatoric and epistemic uncertainties [6]. Evidential deep learning is developed by incorporating the evidential theory into deep neural networks where every training example adds support to a learned higher-order, evidential distribution. Sampling from this distribution yields instances of lower-order likelihood functions from which the data was drawn. By training a neural network to output the hyperparameters of the higher-order evidential distribution, a grounded representation of both epistemic and aleatoric uncertainty can then be learned without the need for sampling. Instead of placing priors on network weights, as is done in Bayesian NNs, evidential approaches place priors directly over the likelihood function. Recently, deep evidential learning has been targeted towards anomaly detection problems where an RNN-based evidential model was designed to predict future trajectories associated with both data and model uncertainties that are used to detect anomalous trajectories in maritime domain [141]. In an alternative approach, deep evidential learning is used for human actions recognition where the Deep Evidential Action Recognition (DEAR) method is developed for open set action recognition task [11]. The referenced paper used the learned evidence to quantify the prediction uncertainty for diverse human actions which enable the model to identify unknown actions where unknown actions will have higher uncertainties.

On the other hand, some probabilistic deep learning models used other methods to compute Prediction Interval (PI) that quantifies the level of uncertainty associated with the point forecasts. David et al, [73] proposed an anomaly detection method that

used an autoregression model and its prediction interval to identify anomalies in environmental sensor streams. The authors used Student's t-distribution to calculate the prediction interval that accounts for uncertainty in both input data and model parameters. They classified a data point as anomalous when it falls outside a given prediction interval. In contrast, Lingxue and Nikolay [172] argued that the measurement of prediction uncertainty does not depend on only input data and model parameters but rather combines with model misspecification and inherent noise. They achieved this by using Monte Carlo dropout to estimate both data and model uncertainties. The estimated uncertainties are then used to construct a prediction interval using z th standard normal where values outside the constructed interval are labeled as anomalies. In a similar context, Legrand et al, [92] investigated how the inclusion of uncertainty in the computation of anomaly score improves the performance of anomaly detection using autoencoder. They do that by developing a new anomaly score function that is weighted with uncertainty as opposed to the Bayesian score function introduced in [172]. This approach is computationally demanding and imposes strong requirements.

In order to minimize the computational requirements, Reunanen et al [126] proposed another unsupervised anomaly detection method that combines Autoencoder and Logistic Regression for outlier detection and prediction in sensor data streams. The Autoencoder reconstructs the input data and produces a hidden representation of the input that can be used to create the required labels for Logistic Regression to classify anomalous points. The outlier is identified as extreme values that exceed a limit of three standard deviations defined using Chebyshev's inequality or any deviation in the correlation of data features. Although no assumption of the distribution of the data is required the method assumes the descriptive statistics (γ and μ) of the unknown normal values to be initially defined. It is also assumed that the feature value of the initial data point has non-zero variance often which the scaling limits for outlier detection cannot be defined. In contrast, a probabilistic forecasting model that returns a full conditional distribution was proposed in [129]. The authors introduced probabilistic forecasting using an autoencoder model that directly outputs parameters of Negative Binomial and Gaussian likelihoods for real-valued time series. This shows the probabilistic forecasts were also achieved by the assumption of a Gaussian distribution on the prediction error. Most of these methods used mean square error as the loss function which is always minimized to the normal distribution in their regression task. However, an exact parametric distribution is often not relevant in applications which will lead to false anomaly alerts due to high prediction variance.

In contrast, Quantile Regression models are trained to minimize quantile loss which is useful in areas of application where it is difficult to define a parametric distribution on the residuals [3]. Quantile regression aims at forecasting conditional quantiles that are given an input sequence and can produce a probabilistic forecast without making any distributional assumption. An attempt was made to use quantile regression for anomaly detection [104] but it is limited to the use of quantile interval to identify only

uncertainties in the data. This limitation led to the research investigation of whether the estimation of uncertainty in prediction using quantile regression will increase the prediction performance and reduce false anomaly alerts. This results in the development of an improved quantile regression anomaly detection model (DQR-AD) in this chapter. Such a model is most welcome in applications requesting better-informed decisions and mitigating false anomaly alerts.

4.3 The Proposed DQR-AD Model

This section presents a detailed description of the proposed (DQR-AD) model. DQR-AD takes the advantage of quantile regression which allows us to get the forecasts at different quantile levels, hence drawing a more comprehensive of the forecasted moment of the time series. The use of quantile regression not only makes it easy to get multiple quantile forecasts but also allows the computation of Prediction Interval (PI) which can be used for anomaly detection. PI quantifies the level of uncertainty associated with the point forecasts, thereby offering an interval of confidence for a prediction of lower and upper bounds. The model consists of three modules, which include: time series Segmentation, time series Prediction, and Anomaly Detection. A detailed concept of these modules is depicted in Fig. 5.1 and described in the following subsections.

4.3.1 time series Segmentation

This section presents the time series Segmentation module which is used to segment time series into overlapping windows which works as follows: Consider a multivariate time series $x = x_1, x_2, \dots, x_t$, where t is the length of the time series and each point $x_i R^m$ (for $i = 1 \dots t$) in the time series is an m -dimensional vector corresponding to the m features or sensor channels read at time t . A sliding window method is used to segment the time series into two sequences of overlapping windows of size l . First, is a history window (h_w) = $x_{t-l}, \dots, x_{t-1}, x_t$ of length l which defines the number of previous time stamps in history that will be used as input to the model. Second, is the 1-steps predicted window (p_w), which represents the number of time steps to be predicted where the number of dimensions d being predicted is $1 < d < m$. For example, given a history window of five previous time steps $h_w = 5$ and one step ahead prediction window $p_w = 1$ will result in equation (4.1) below:

$$x_{t-4}, x_{t-3}, x_{t-2}, x_{t-1}, x_t \rightarrow x_{t+1} \quad (4.1)$$

In a regression problem, the left-hand side of equation (??) serves as input data to the model and the right-hand side as the output which is treated as a label to the input data. In this research, the aim is to predict the next time step value for a single channel (i.e.

4.3 The Proposed DQR-AD Model

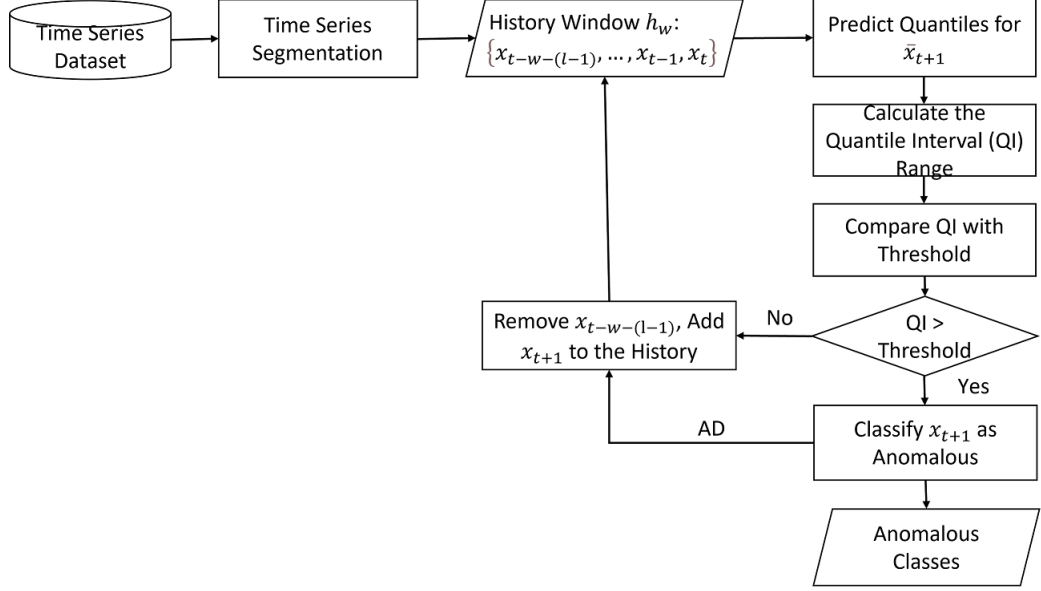


Figure 4.1: Schematic representation of DQR-AD method that involves time series segmentation, prediction, and anomaly detection

$d = 1$ and $p_w = 1$) which resulted in a single scalar prediction \bar{y}_t to be generated for the actual value at each step t . Detailed steps involved in time series segmentation are given in Algorithm 1. The Algorithm receives as input a sequence of time series, the number of time steps which represent the size of the history window, and a number of features. It then loops through the sequence segmenting it into a subsequence of history and prediction windows which are stored in two different lists. The Algorithm returns these two lists as output. To reduce the dynamic range of the signal and enhance the performance of the regression model, the min-max normalization method is applied to the output from Algorithm 1 before passing it to the next module. Normalization is important for the numerical stability of training deep neural networks, particularly because time series prone to anomalies can have unbounded records. It also provides more emphasis on the temporal pattern of time series and has been shown to be useful for sequence mining [95]. The sequence of both history and prediction windows are scaled between 0 and 1 ($x_{ij} \in [0, 1]$) where $j = 1$ to m , as shown in equation (4.2).

$$x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (4.2)$$

where x_{max} and x_{min} are vectors that contain the minimum and maximum values of the features. The scaling is done for each point per feature.

Algorithm 1 time series Segmentation

```

1: Input: Sequence  $S$ ,  $S_{t=0}^T$ , n_steps, n_features.
2: Output:  $x$  (list history window ( $h_w$ )) and  $y$  (list of prediction window ( $p_w$ )).
3: for  $i \leftarrow 0$  to  $len(S)$  do
4:    $end\_ix \leftarrow i + n\_steps$ .
5:   if  $end\_ix > Len(S) - 1$ 
6:     break.
7:   else do.
8:      $x \leftarrow S[i : end\_ix]$ 
9:      $y \leftarrow S[end\_ix]$ 
10:  end if
11: end for
12: return  $x, y$ 

```

4.3.2 time series Prediction

This section describes a time series Prediction module that involves Deep Quantile regression (DQR) that uses the history window (h_w) to forecast the quantiles of the next time step (p_w). The section starts with a description of DQR process which works as follows: The DQR model is built using multilayered LSTM-based RNNs. The choice of LSTM is motivated by its performance in forecasting time series points and its ability to extract long-term patterns in the time series [58]. However, this research work focuses on developing a prediction model, which can forecast time series points considering uncertainty. This is achieved via quantile regression that generates quantiles as opposed to pointing estimates in ordinary LSTM model [58]. Quantile regression is used because it predicts conditional quantiles of the target distribution which produces an accurate probabilistic forecast without making any distributional assumptions [164]. To perform quantile regression on LSTM, the custom Quantile Loss (QL) function is created which penalizes errors based on the quantile and error's sign. In Quantile Regression, models are trained to minimize the QL and by minimizing this loss, the model can learn to predict the normal behavior of the time series. To estimate uncertainty, the DQR model focuses on predicting extreme values (lower (10th), upper (90th), and classical (50th) quantiles).

However, one of the complexities of estimating uncertainties in LSTM using QR is uncertainty due to internal weights initialization, which results in quantiles overlapping. To handle this problem, bootstrapping is employed. This allows the regression model to be iterated n times, thereby storing the predicted values in an array which is finally used to compute the desired quantiles. By computing upper and lower quantiles, the model is considered to have covered the range of possible values. The size of this range can be small when the model is sure about the future and big otherwise.

This behavior is used to enable the model to detect abnormal values from the test set as described in the next section. Detailed steps of time series prediction based on the developed model are given in Algorithm 2. The Algorithm receives as input the list of history and prediction windows which is used to train and fit the model for quantile regression. The model predicts the quantile values for each next time step based on the history. To achieve this, a quantile loss function is created which is integrated with the LSTM model. To have an optimal result, bootstrapping is employed where the model is iterated 100 times predicting and storing the quantile values in an array. Thus, the output of this algorithm is an array of predicted quantiles.

4.3.2.1 LSTM Network Architecture

An extensive experiment is carried out to finalize the model architecture and its parameters. Grid [124] and random [15] optimization search strategies are commonly used for hyper-parameters selection in DNN [14]. Although the random search strategy has a lower computational cost, it involves uncertainties during the search process [34]. As such, we used a grid search strategy due to its simplicity and being deterministic. Similarly, we adopted the strategy to avoid the possibility of having an improved anomaly detection performance which was induced by the optimization algorithm. The detailed procedure adopted for the model parameter selection is described below:

1. Training and validation data were prepared using normal time series data
2. Search candidate sets were configured for each hidden layer with sequence from 16 to 512 with a step size equal to 16
3. A loop is executed to train LSTM using training data, and then apply trained LSTM on validation data to obtain VAMSE under all possible combinations of the parameters candidates.
4. We then choose the model parameters 16 and 64 which have the minimal VAMSE.

Therefore, the proposed model consists of four hidden layers that include input, output, and two hidden layers for the prediction of quantile values as shown in Fig. 4.2. The input layer has l input nodes corresponding to segmented time series into l window vectors which represent the h_w . The first and second hidden layers are composed of 64 and 16 nodes respectively, which is followed by an element-wise activation function ReLU. These fully connected layers learn the pattern of the time series and pass it to the next layer for generating the forecasts. At last, the output layer is a fully connected dense layer with its nodes connected to all nodes in the previous layers. This last layer produces the predicted quantiles of the next timestamp with a number of nodes equal to the size of p_w . For the model predicting only the next time stamp, the number of output nodes is 1.

Algorithm 2 time series Prediction using DQR Model

```

1: Input: list  $x$  and list  $y$ .
2: Output: array of lowerQuantile, array of classQuantile, array of upperQuantile.
3: function q_loss( $q, y, f$ )
4:    $e \leftarrow (y - f)$ 
5:    $loss \leftarrow mean(maximum(q \times e, (q - 1) \times e), axis = -1)$ 
6:   return loss
7: end function
8: losses  $\leftarrow$  [lambda  $y, f$ : q_loss(0.1, $y,f$ ), lambda  $y, f$ : q_loss(0.5, $y,f$ ), lambda  $y, f$ : q_loss(0.9, $y,f$ )].
9: inputLayer  $\leftarrow$  ( $x.shape[1], x.shape[2]$ )
10: lstmLayer1  $\leftarrow$  LSTM(64, return_sequence=True, dropout=0.3)(inputLayer, training=True)
11: lstmLayer2  $\leftarrow$  LSTM(16, return_sequence=True, dropout=0.3)(lstmLayer1, training=True)
12: lstmLayer3  $\leftarrow$  Dense(50)(lstmLayer2)
13: output_10  $\leftarrow$  Dense(1)(lstmLayer3)
14: output_50  $\leftarrow$  Dense(1)(lstmLayer3)
15: output_90  $\leftarrow$  Dense(1)(lstmLayer3)
16: lstmModel  $\leftarrow$  Model(inputLayer, [output_10, output_50, output_90])
17: lstmModel.compile(loss=losses, optimizer=adam, loss_weights=[0.3, 0.3, 0.3])
18: lstmModel.fit( $x, [y, y, y]$ )
19: forecastModel  $\leftarrow$  K.function([lstmModel.layers[0].input, K.learning_phase()], [lstmModel.layers[-3].output, lstmModel.layers[-2].output, lstmModel.layers[-1].output])
20: lowerQuantile  $\leftarrow$  [ ]
21: classQuantile  $\leftarrow$  [ ]
22: upperQuantile  $\leftarrow$  [ ]
23: for  $i \leftarrow 0$  to 100 do
24:    $\bar{y} \leftarrow$  forecastModel( $[x, 0.5]$ )
25:   lowerQuantile  $\leftarrow$   $\bar{y}[0]$ 
26:   classQuantile  $\leftarrow$   $\bar{y}[1]$ 
27:   upperQuantile  $\leftarrow$   $\bar{y}[2]$ 
28: end for
29: return lowerQuantile, classQuantile, upperQuantile

```

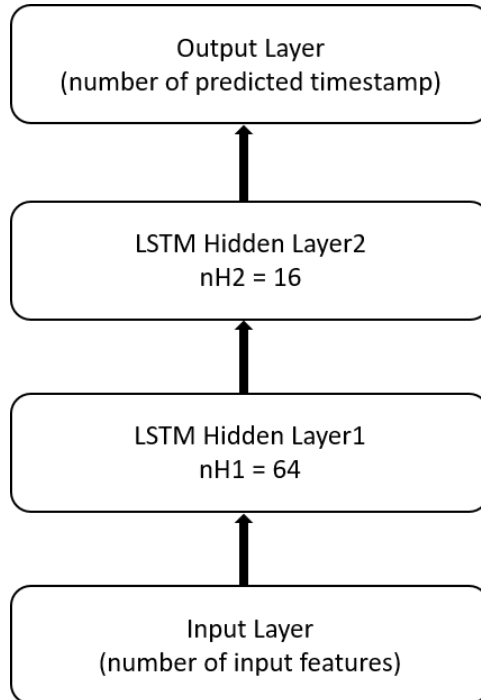


Figure 4.2: DQR model network architecture for time series prediction

In comparison with other proposed model parameters, [103] used stack LSTM architecture where one unit is used in the input for each feature. Similarly, the number of predicted points multiplied by the number of features is used as the units in the output layer. In [126] the number of hidden neurons is varied in an autoencoder from 2 to 10 to provide an extensive evaluation of their model. While two convolution layers each followed by a max-pooling layer are used in [110]. The number of nodes in the input layer is equal to the size of the window and the output layer has only one node because the model is predicting the next time stamp. Similar to our model, each convolution layer is composed of 32 filters followed by an element-wise activation function Relu. Although these models used stacking recurrent hidden layers that capture the structure of time series, none of them combined it with quantile regression for predicting time series and using it for anomaly detection.

4.3.2.2 Quantile Loss

The proposed LSTM structure has the ability to generate point or probabilistic forecasts depending on the loss function. In regression problems, the most commonly used loss function is mean square error. When an LSTM prediction model is created that minimizes this loss, then it is predicting the mean value of the output which may have been noisy in the training set. This model the average behavior of the sequence, which

is useful but gives less information about the forecasts. Quantile regression provides forecasts at different quantile levels which draw a more comprehensive picture of the forecasted moment. Quantile regression can be achieved on LSTM using a quantile loss which is defined in equation (4.3).

$$L(\xi_i|\alpha) = \begin{cases} \alpha\xi_i & \text{if } \xi_i \geq 0 \\ (\alpha-1)\xi_i & \text{if } \xi_i < 0 \end{cases} \quad (4.3)$$

where α is the required quantile with values between 0 and 1 and $\xi_i = y_i - f(x)_i$ with $f(x)_i$ as the predicted quantile and y_i as the observed value for the corresponding input x . To create a quantile loss for the entire dataset, the average of the quantile loss for an individual point is taken, as shown in equation (4.4).

$$L(y, f|\alpha) = \frac{1}{N} \sum_{i=0}^N L(y_i - f(x_i)|\alpha) \quad (4.4)$$

4.3.3 Anomaly Detection and Classification

This section presents Anomaly Detection and Classification module which classify anomalous points in time series data. The anomaly detection process is described as follows: Given the model predicted quantiles from the previous section, a time series point can then be identified as anomalous using the QI. The QI is computed as a vector of difference between upper and lower quantiles (90 -10 quantiles range). It is expected to have a small interval when the model knows about the future and on the contrary, to have uncertainties when there is a bigger interval. This is because the model is not trained to handle this type of scenario and is unable to detect such changes in the data that can result in anomalies. The QI is used as an anomaly score: large values flag-up similarly relevant anomalies for the given timestamp. However, this module is expected to clarify the threshold based on the time series type: a limitation and constraint for most anomaly detection techniques. When the QI related to a point in time is greater than the threshold, that point is classified and flagged up as an anomaly. According to the literature, [73, 114], two strategies are used for flagging abnormal data which includes: Anomaly Detection (AD) and Anomaly Detection and Mitigation (ADAM). In order to allow continuous detection of an anomaly, AD is chosen in this method as against ADAM, which faces the problem of false alarms for dynamic data [114]. Detailed steps of the anomaly detection process are given in Algorithm 3. The Algorithm receives input arrays of lower and upper quantiles from the previous module and a fixed threshold value. Quantile interval is then computed as a difference between upper and lower intervals. The QI is compared with the threshold to classify anomalies that are given as output. The output is given as an array of anomalous points.

Algorithm 3 Anomaly Detection and Classification

```
1: Input: threshold, lowerQuantile, upperQuatile.
2: Output: array of anomalous class.
3: anomaly  $\leftarrow$  []
4: QI  $\leftarrow$  (upperQuantile - lowerQuantile)
5: if QI > threshold
6:   anomaly  $\leftarrow$  1.
7: else do.
8:   anomaly  $\leftarrow$  0
9: end if
10: return anomaly
```

4.4 Dataset Description

4.4.1 NAB (Numenta Anomaly Benchmark) Dataset

NAB dataset [90] is a publicly available streaming benchmark dataset released by Numenta and available in their repository ¹. The benchmark dataset has been widely used in literature for evaluating anomaly detection techniques [2, 131]. This dataset contains 58 data streams, each with 1000 - 22,000 records. The dataset contains real data from different application domains: road traffic, network utilization, online advertisement, sensors on industrial machines, and social media, and some artificially generated data files that demonstrated some anomalous behaviors that are not present in the real data. Each dataset file records have time stamps and data values with anomaly labels stored in a separate set of files. An additional column is created in each dataset that holds an anomalous label for each data point. The labels are created either based on the known root cause or as a result of labeling procedures defined in [2]. Although NAB contains labeled streaming anomaly detection datasets, it is faced with some challenges, which may affect point anomaly detection [140]. First, each label of an unusual point depends on the defined anomalous window. This means when one data point within the window is an anomaly, all other data points in that window are also abnormal. Secondly, most of the data files have different data distributions where the distribution of a few timestamps is quite different from the distribution of the remaining time series. The second challenge affects the choice of training and test sets which may come from different distributions and will have a negative impact on the training and evaluation of time series models. However, this characteristic becomes the main reason why we selected this dataset to evaluate the performance of our model in handling uncertainties due to model misspecification, as discussed in section 3. Figures 4.3 and 4.4. shows normal time series samples with anomalous points from machine temperature and ambient

¹<https://github.com/numenta/NAB>

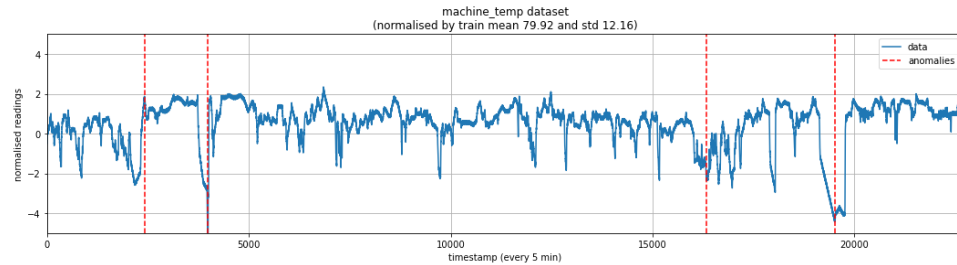


Figure 4.3: A Sample of Normal time series (blue line) with anomalous point (vertical red dashed line) from Machine Temperature Dataset

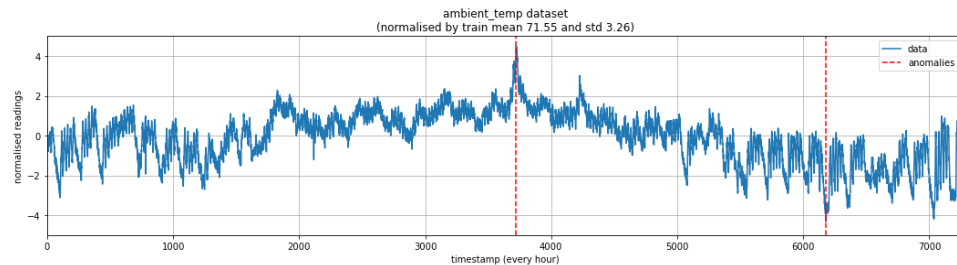


Figure 4.4: A Sample of Normal time series (blue line) with anomalous point (vertical red dashed line) from Ambient Temperature Dataset

temperature datasets respectively.

4.4.2 ECG Dataset

Since NAB datasets have fewer anomalies, we used ECG data with a relatively large number of anomalous points. ECG data is an anomaly benchmark dataset from the MIT- BIH Arrhythmia Database ¹. The dataset consists of 47 ECG records which are slightly over 30 minutes long from 47 patients studied by the BIH Arrhythmia Laboratory. Literature [36, 108] shows 23 of the recordings were chosen at random from a set of 4000 24-hour ambulatory ECG recordings collected from a mixed population of inpatients (about 60%) and outpatients (about 40%) at Boston’s Beth Israel Hospital; the other 25 recordings were selected from the same set to include less common but clinically significant arrhythmia’s that would not be well-represented in a small random sample. According to [108], the records were digitized at 360 samples per second per channel with 11-bit resolution over a 10 mV range. Two or more cardiologists independently annotated each record; disagreements were resolved to obtain the computer-readable reference annotations for each beat (approximately 110,000 annotations in all) included with the database.

¹<https://www.physionet.org/content/mitdb/1.0.0/>

4.4.3 sMAP Dataset

sMAP is a real sensor data that consist of the reading of electrical sensors from an elevator in Cory Hall at the University of California, Berkeley. The dataset contains five unlabeled time series (SMAP1 to SMAP5) each one having 20,000 data points and four features. The dataset is retrieved using a python public front end ¹. To label the time series and ease the computation of evaluation metric for our algorithm, we used the same procedure as in [126] where the value of the random features of a data point is swapped. This approach preserves the order of the data points and helps in providing an access to the anomalies during evaluation.

4.5 Experiments

This section covers the extensive experiments conducted to test and compare the performance of DQR-AD with eight anomaly detection methods that model prediction errors using Gaussian distribution to identify anomalies. These methods includes LSTM-AD [103], Deep LSTM-AD [32], DeepAnT [110], NumentaTM [2], ContextOSE [71], EXPoSE [132], AE [126], VAE-LSTM [96]. The experiment is conducted using real and synthetic datasets from the different application domains. We divided the experiment into three parts which are based on NAB, ECG, and sMAP benchmark datasets. Each experiment starts with experimental setups and ends with results and discussion. All experiments are carried out on the same computer with an Intel Pentium core i7 processor, Windows OS, and deep learning libraries on python and anaconda 3.7.

4.5.1 Part I Experiment: NAB Dataset

4.5.1.1 Experimental Setups

In this subsection, we provide a detailed description of how the experiment is conducted. Two levels of the same experiment are conducted in this section. On the first level, DQR-AD is evaluated and compared with four time series anomaly detection methods based on precision and recall using 20 NAB time series from different domains. In this experimental work, the same time series reported in [110] is used. On the second level of the experiment, a detailed analysis of five state-of-the-art methods and compare them with DQR-AD on the whole NAB benchmark datasets. For the anomaly detection part, AE and VAE-LSTM are evaluated on the same settings and parameters with DQR-AD; while for the remaining methods, we used the same settings and parameters as stated in [110]. For this detailed evaluation, we used F-score (4.5) to report the overall performance of the anomaly detection methods. F-score is the

¹https://pythonhosted.org/Smmap/en/2.0/python_access.html

most commonly used metric to evaluate the performance of anomaly detection methods [110]. Since NAB records have multiple time series related to different application domains, the average F score is reported for each method in relation to each domain:

$$F_score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4.5)$$

4.5.1.2 Training and Test Data Construction

To train the model for time series prediction, we first split each time series into 80% as the training set and the rest of the 20% data as a test set. We further split the training set and used 20% of it for validation. A sliding window technique is used to segment both the training, test and validation sets into history (h_w) and prediction window (p_w). Although, literature shows the NAB anomaly scoring technique to be based on anomaly windows which is chosen to be 10% of the number of instances in the datasets, divided by the number of anomalies in a given dataset [2]. But, Singh and Olinsky [140] have argued that the window size cannot be chosen in this way in many real-world settings which gives room for proposing a different way of selecting window size depending on the problem scenario. In this paper, we choose the size of the window through computing auto-correlation which shows the presence of either daily, hourly, or weekly patterns in all the time series. For example, in nyc_taxi and time series, the taxi demand seems to be driven by a weekly trend which was proved by computing auto-correlation as shown in Fig. 4.5. As such, a daily history window of size 24 timestamps (one observation every hour) and a prediction window of size 1 were chosen for predicting only one timestamp based on previous daily timestamp points. We also used the same 20% test split and the same window size for all the anomaly detection methods we used in our comparisons. Finally, the segmented time series are normalized and passed as input for model training, validation, and testing.

4.5.1.3 Model Training and Validation

The training and validation set from the previous step are used for model training and validation. Since this is an unsupervised approach, we do not use any label information in the training process. Instead, for each window of previous timestamps, only the next timestamp is predicted and serves as the target. Both the training and validation sets are also reshaped into a 3-dimensional format as required by the model. We train the model using mini-batch gradient descent where a batch size of 128 is used and 50 epochs. At each iteration of the epoch, the training and validation loss is captured and stored which is later plotted as shown in Fig. 4.6.

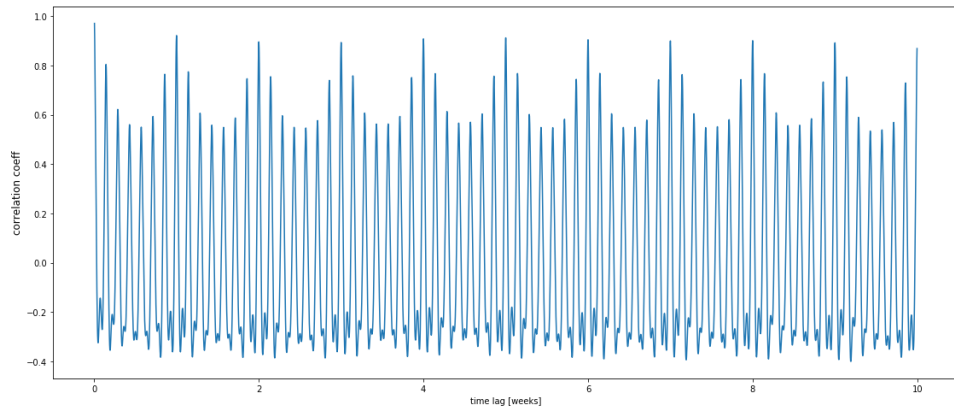


Figure 4.5: Autocorrelation of 10 weeks depth for nyc_taxi time series data

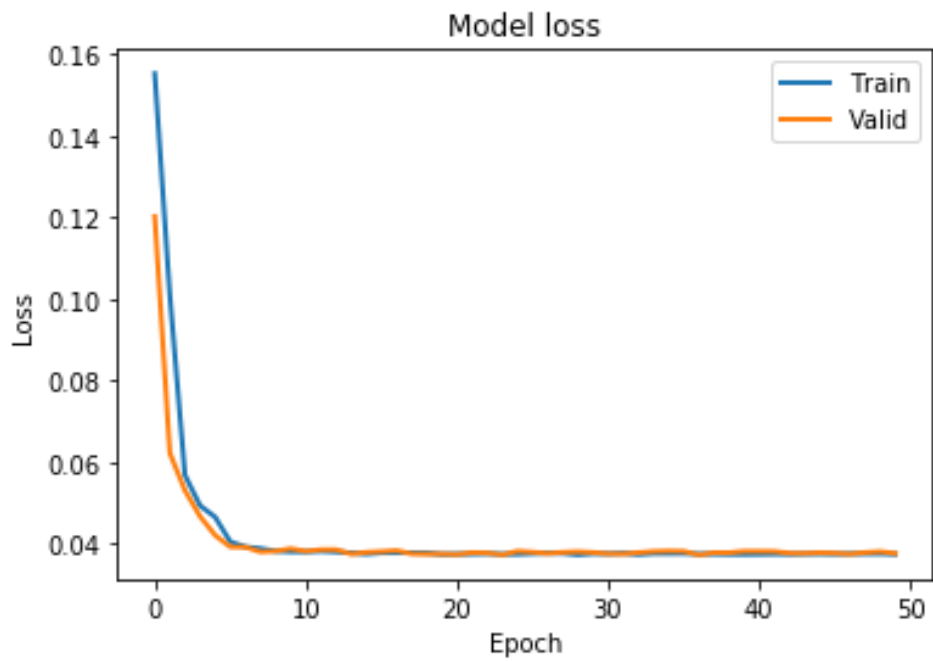


Figure 4.6: DQR Model training and validation loss for nyc_taxi time series data

4.5.1.4 Model Testing and Prediction

To produce the predicted quantiles for an entire time series test set, we make use of bootstrapping in the prediction phase by reactivating the dropout in our network and iterating the prediction 100 times, thereby storing the predicted values in an array which is finally used to compute the desired quantiles. By computing upper and lower quantiles, the model is considered to have covered the range of possible values. This behavior is used to enable the model to detect abnormal values from the test set as described in the next section.

4.5.1.5 Anomaly Detection and Classification

This part of the experiment shows how QI can be used for anomaly detection and classification. The quantile values predicted in the previous step are used to compute QI which is used in computing an anomaly score. The value of QI is low when the model understands the pattern of the time series and is higher in a period of uncertainty. This behavior is used to classify anomalies by setting up a threshold value based on the time series pattern. Finding the best threshold is very important for evaluation. Each time series in the NAB dataset has its own characteristics and finding a generic threshold that works for all the time series is not a straightforward task. As such, we used the validation data to set a threshold for each time series. This is achieved by choosing a value that provides a reasonable trade-off between precision and recall of the predicted results on validation data as shown in Fig. 4.7. The threshold value is then used to classify anomalous points in the test data. Detailed experimental results and performance of the method in comparison with other anomaly detection methods are given in the next section.

4.5.1.6 Experimental Results and Discussions

Figure 4.8 shows DQR-AD anomaly detection result from the test set of nyctaxi dataset: The actual time series is depicted in red lines and QI in blue dots. It can be seen in this example; the QI goes high in the period of uncertainties (circled in green). We used this behavior to set up a threshold value of 17000 which is the maximum trade-off value between precision and recall of the validation set as indicated in the previous section. The threshold value is then used for classification in the test data. The test data contains 3 anomalous points with classification results shown in Fig. 4.9. where the actual normal points are depicted in blue and abnormal points in orange with a red line indicating the threshold used by DQR-AD for classification of anomalies. True Positive and False Positive values are depicted with points above the threshold line in orange and blue, respectively. To have a better look at the classification result, a confusion matrix is shown in Fig. 4.10. The confusion matrix shows that DQR-AD method was able to identify one anomalous point with only 4 false positives. When

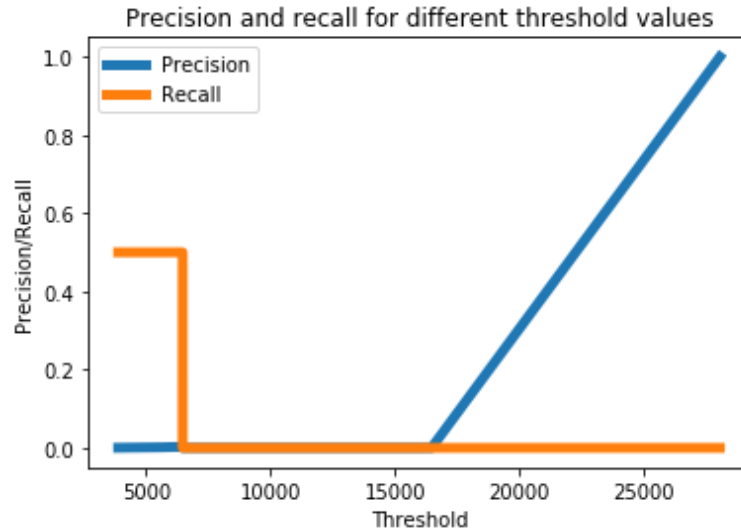


Figure 4.7: A trade-off between precision and recall used for setting up a threshold value for nyc_taxi time series data

compared with the LSTM-AD method on the same time series and under the same settings, it shows the same performance in terms number of anomalies identified (True Positive) but with a higher rate of false positive value equal to 21 as shown in the second confusion matrix in Fig. 4.11.

As indicated in the above figures, the classification result of DQR-AD is significantly better than that of LSTM-AD in terms of false positive values which is the main problem that this chapter addresses. On a more detailed level, Table 4.1 shows the results of the first level of the experiment. It can be observed from this table that DQR-AD obtained relatively better precision and recall than other methods in almost all the time series with only one case where precision and recall is 0. This demonstrates its ability in detecting a higher number of anomalies with low false positive rates. On the other hand, Table 4.2 shows the result of the second level experiment where the mean F_score for the three algorithms is reported on the whole NAB dataset. As indicated in the table, DQR-AD outperforms other methods in all domains except one (i.e. Real AWS Cloud Watch) where DeepAnT has better performance with a relatively small margin. DQR-AD is approximately 2 – 3 times better than the DeepAnT which performs better than LSTM-AD for different domains in the NAB dataset. In Table 4.2 the results are described by the mean F_score for each domain of the NAB benchmark dataset. As indicated in the table, DQR-AD outperforms other methods in all domains except in two domains (i.e., real known cause and real AWS cloud watch) where VAE-LSTM and DeepAnT, respectively, have better performance with a relatively small margin. It is therefore clear that DQR-AD outperforms both other methods on 4 out of

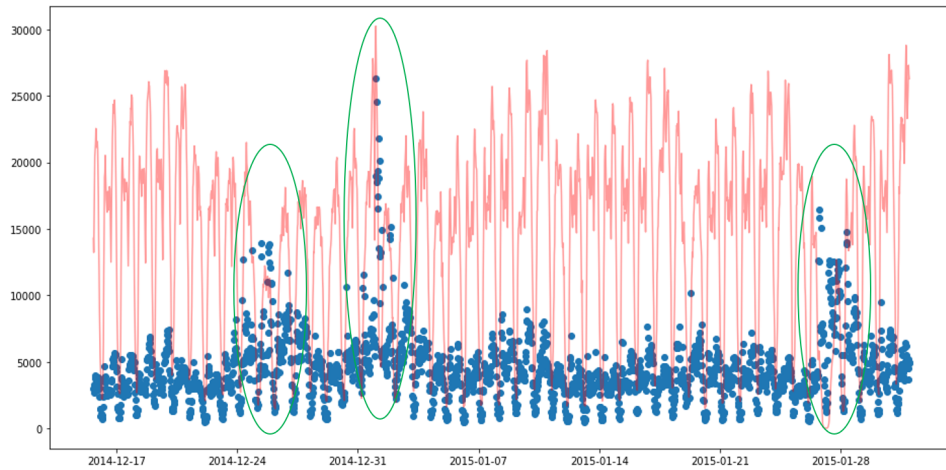


Figure 4.8: For nyc_taxi dataset, actual time series values (red) plotted against the IQ (blue) to show periods of uncertainties (circled in green) in the test set

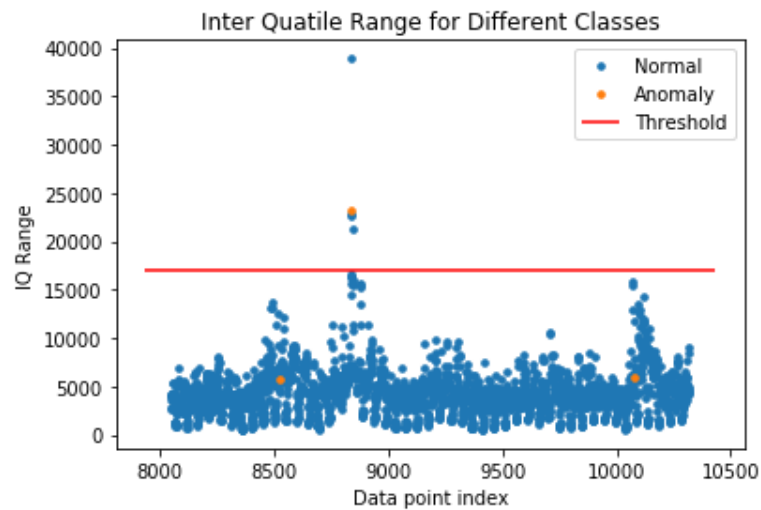


Figure 4.9: Using a threshold value of 17000 for anomaly detection in nyc_taxi time series data

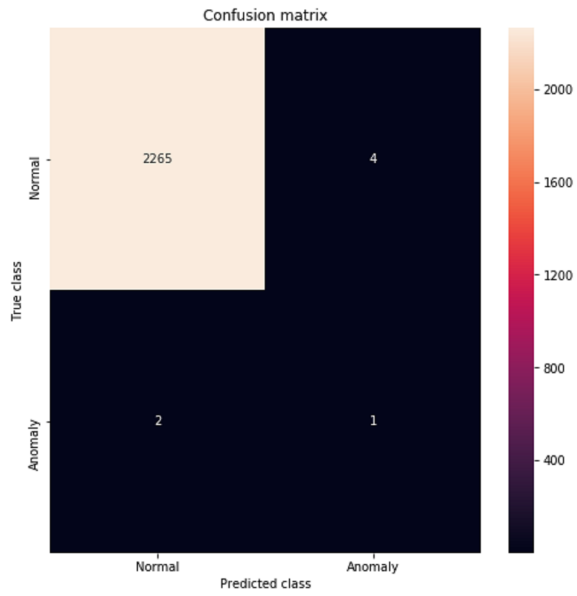


Figure 4.10: Confusion Matrix for DQR-AD Method

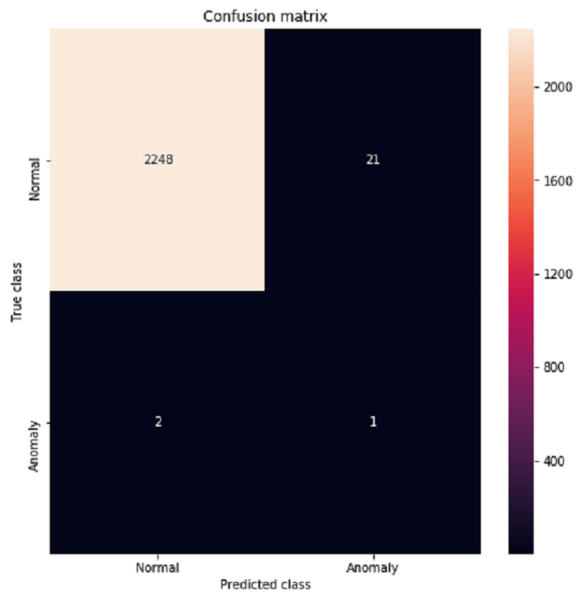


Figure 4.11: Confusion Matrix for LSTM-AD Method

6 domains from the table.

NAB Dataset		NumentaTM		ContextOSE		DeepAnT		VAE-LSTM		DQR-AD	
		Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Real Known Cause	nyc_taxi	0.85	0.006	1	0.002	1	0.002	0.961	1	1	0.33
	ambient_temperature	0.05	0.006	0.33	0.001	0.26	0.06	0.806	1	1	0.06
	cpu_utilization	0.52	0.01	0.12	0.001	0.63	0.36	0.694	1	1	0.5
	ec2_request_latency	1	0.009	1	0.009	1	0.04	0.993	1	1	0.7
	machine_temperature	0.27	0.004	1	0.001	0.8	0.001	0.559	1	0.5	0.5
	rogue_agent_key_hold	0.5	0.005	0.33	0.005	0.34	0.05	0.02	0.1	1	0.05
	rogue_agent_key_updown	0	0	0	0	0.11	0.001	0.11	0	0.5	0.1
Real Ad Exchange	exchange_2_cpc_results	0	0	0.5	0.006	0.03	0.33	0	0	0.7	0.11
	exchange_3_cpc_results	1	0.007	0.75	0.02	0.71	0.03	0	0	1	0.33
Real Tweets	Twitter_volume_GOOG	0.38	0.005	0.75	0.002	0.75	0.01	0.03	1	0.83	1
	Twitter_volume_IBM	0.22	0.005	0.37	0.002	0.5	0.005	0	0	1	0.1
Real Traffic	occupancy_6005	0.2	0.004	0.5	0.004	0.5	0.004	0.01	1	1	0.1
	occupancy_t4013	0.66	0.008	1	0.008	1	0.036	0.11	0.5	1	0.48
	speed_6005	0.25	0.008	0.5	0.004	1	0.008	0	0	1	0.5
	speed_7578	0.6	0.02	0.57	0.03	1	0.07	0.01	0.33	1	0.25
	speed_t4013	0.8	0.01	1	0.008	1	0.08	0	0	1	0.20
	TravelTime_387	0.33	0.004	0.6	0.01	1	0.004	0	0	0	0
	TravelTime_451	0	0	1	0.005	1	0.009	0.10	1	1	0.25
Real AWS Cloud Watch	ec2_cpu_utilization_5f5533	1	0.01	1	0.005	1	0.01	0.06	1	1	0.23
	rds_cpu_utilization_cc0c531	1	0.002	1	0.005	1	0.03	0	0	1	0.33

Table 4.1: Comparative evaluation of DQR-AD with four anomaly detection methods (NumentaTM, ContextOSE, DeepAnT, and VAE-LSTM) on 20 NAB time series from different domains. Precision and Recall is reported in this table

NAB Dataset	ContextOSE	EXPoSE	NumentaTM	VAE-LSTM	DeepAnT	DQR-AD
Real Known Cause	0.005	0.005	0.012	0.629	0.200	0.408
Real Ad Exchange	0.022	0.005	0.035	0.006	0.132	0.301
Real Tweets	0.003	0.003	0.010	0.011	0.075	0.280
Real Traffic	0.02	0.011	0.036	0.060	0.223	0.376
Real AWS Cloud Watch	0.007	0.015	0.018	0.002	0.146	0.031
Artificial With Anomaly	0.022	0.004	0.017	0.012	0.156	0.276

Table 4.2: Comparative evaluation of DQR-AD with five other anomaly detection methods (ContextOSE, EXPoSE, NumentaTM, DeepAnT, and VAE-LSTM) applied to entire NAB dataset using mean F_score for each domain

4.5.2 Part II Experiment: ECG Dataset

In order to evaluate the proposed model with a dataset that has a larger number of anomalies, ECG data were used in this part of the experiment. We compare the performance of the proposed model with LSTM-AD [103] and Deep LSTM-AD [32] that used the same ECG dataset in their evaluation.

4.5.2.1 Experimental Setup

We used the same setting in the part 1 experiment where our proposed model is trained using an 80-20 splitting pattern. Similarly, an additional 20% from the training set

is used for validation. Both the training, test and validation sets are segmented into history (hw) and prediction window (pw). The size of the history window is set-up daily window of size 24 timestamps (one observation every hour). A prediction window of size 1 was chosen for predicting only one timestamp whereas for each window of previous timestamps only the next timestamp is predicted. We maintained the unsupervised learning in the training process of both algorithms by removing the class label and considering for each window of previous timestamps, only the next timestamp is predicted and serves as the target. We adopted the parameter settings in [103] for LSTM-AD, while on the other hand, we used the same parameter settings in the previous part of the experiment for DQR-AD. The performance of the two models is reported using F_Score, Precision, and Recall.

4.5.2.2 Experimental Results and Discussion

In order to compare the performance of our propose method with LSTM-AD in terms of the number of anomalies detected, a confusion matrix is shown for both methods that indicate the classification results from the test set of MBA_ECG14046_data_8 datasets. The confusion matrix for DQR-AD is shown in Fig. 4.12 which shows that the DQR-AD method was able to identify 26 anomalous points with only 33 false positives. When compared with the confusion matrix for the LSTM-AD method in Fig. 4.13 on the same dataset, it shows a better performance in terms number of anomalies identified (176 True Positive) but with a higher rate of false positive value equal to 283 as shown in the second confusion matrix

As indicated in the two figures, the classification result of DQR-AD is significantly better than that of LSTM-AD in terms of false positive values which is the main problem that this chapter addresses. On a more detailed level, Table 4.4 shows a higher level performance result of the methods in terms of F_Score, Precision, and Recall. It can be observed from this table that, although LSTM-AD method has better performance in almost all the time series, DQR-AD obtained relatively better than Deep LSTM-AD method. This demonstrates its ability in detecting a higher number of anomalies with low false positive rates compared to LSTM-AD.

4.5.3 Part III Experiment: sMAP Dataset

In this part of the experiment, we tested and evaluate our method using sMAP dataset which is used in [126].

4.5.3.1 Experimental Setup

We maintained the same setting as previous parts of experiments where our proposed model is trained using 80-20 splitting pattern. Similarly, an additional 20% from the

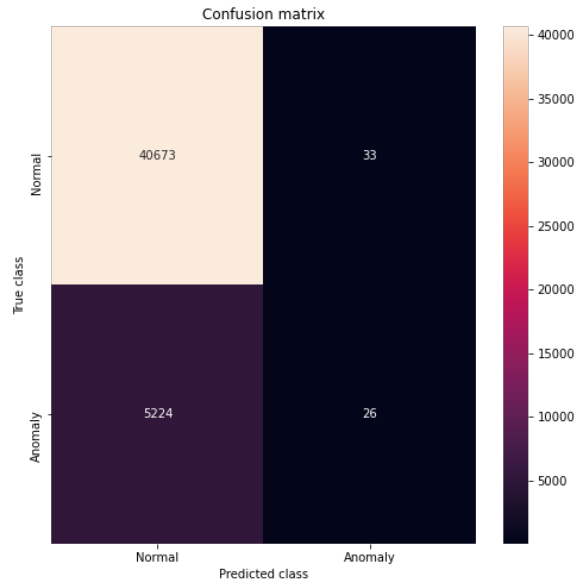


Figure 4.12: Confusion Matrix for DQR-AD Method on MBA_ECG14046_data_8

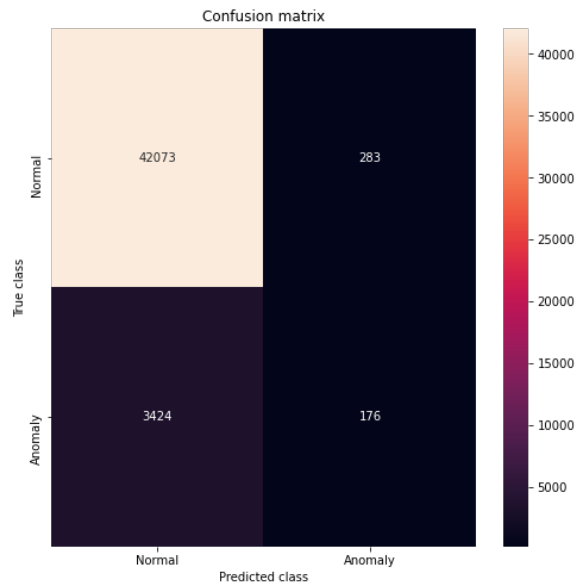


Figure 4.13: Confusion Matrix for LSTM-AD Method on MBA_ECG14046_data_8

training set is used for validation. Both the training, test and validation sets are segmented into history (hw) and prediction window (pw). The size of the history window is set up daily window of size 24 timestamps (one observation every hour). A prediction window of size 1 was chosen for predicting only one timestamp whereas for each window of previous timestamps only the next timestamp is predicted. We maintained the unsupervised learning in the training process of both algorithms by removing the class label and considering for each window of previous timestamps, only the next timestamp is predicted and serves as the target. For AE, we used 2 and 10 hidden neurons to provide an extensive evaluation of the outlier detection algorithm as indicated in [126]. While on the other hand, we used the same parameter settings in the previous part of the experiment for DQR-AD. In this part of the experiment, AUROC is used as a metric to measure the performance of our model. Similar to [126], the performance evaluation is repeated 10 times which enables the evaluation of our learning algorithm for the data stream. An average AUROC for the ten repetitions is reported for each time series in Table 4.4.

4.5.3.2 Experimental Results and Discussion

Table 4.4 reports the outlier detection performance of DQR-AD compared with AE on the sMAP dataset (SMAP1 to SMAP5). Each cell of the table contains an average AUROC value of ten repetitions. It can be observed from the table, both algorithms achieved AUROC values that exceed 0.8 which shows they all succeeded in detecting anomalies in sensor streams. However, the result in Table 4.4 demonstrates good performance of DQR-AD where out of the five-time series (SMAP1 to SMAP5), DQR-AD is 10% better in performance than AE on three datasets (SMAP1, SMAP3, and SMAP5) and relatively equal performance on the remaining two datasets which have a higher level of noise. This demonstrates the ability of DQR-AD in consistently provide accurate results on different datasets and experimental settings. The use of sMAP time series with 4-dimensional features demonstrates the applicability of our proposed approach in multivariate time series data.

4.6 Summary

This chapter presents a deep learning-based anomaly detection method for the detection and classification of anomalies in time series data. Deep Quantile Regression Anomaly Detection (DQR-AD) is unsupervised and does not require any assumption of the regular data distribution to identify anomalous data points. Instead, the method used Quantile Interval which quantifies the level of uncertainties associated with the LSTM point forecasts and helps mitigates false anomaly alerts. The proposed method can detect sudden spikes in time series: this particular challenge is generally missed

in reports listed in the literature review section that propose other distance and density anomaly detection methods.

In the first part of the experiment, DQR-AD is evaluated on the NAB benchmark dataset containing 58 real and synthetic time series and compared with 6 other prediction-based anomaly detection methods that assume normal distribution on prediction or reconstruction error for identification of anomalies. The experimental results as shown in Table 1 indicates that DQR-AD obtained relatively better precision than all other methods. This demonstrates DQR-AD is capable of detecting a higher number of anomalous points with low false positive rates. Similarly, the results in Table 2 show DQR-AD to be approximately 2 – 3 times better than the DeepAnT which performs better than all the remaining methods on all domains in the NAB dataset. This demonstrates our approach can be practically applied to the time series with large amounts of unlabeled data. In the second part of the experiment, DQR-AD has 10% better performance than AE on three datasets (SMAP1, SMAP3, and SMAP5) and equal performance on the remaining two datasets (SMAP2 and SMAP4) with relatively higher levels of noise. The use of sMAP time series with 4-dimensional features demonstrates the applicability of DQR-AD on multivariate time series data.

4.6 Summary

ECG Dataset	LSTM-AD			Deep LSTM-AD			DQR-AD		
	F_Score	Precision	Recall	F_Score	Precision	Recall	F_Score	Precision	Recall
MBA_ECG14046_data_1	0.085	0.314	0.049	0.002	0.088	0.001	0.001	0.125	0.001
MBA_ECG14046_data_2	0.072	0.458	0.039	0.002	0.278	0.001	0.002	0.316	0.001
MBA_ECG14046_data_3	0.097	0.603	0.053	0.011	0.444	0.005	0.01	0.441	0.005
MBA_ECG14046_data_4	0.087	0.383	0.049	0.002	0.333	0.001	0.004	0.186	0.002
MBA_ECG14046_data_5	0.119	0.680	0.065	0.009	0.434	0.005	0.005	0.351	0.003
MBA_ECG14046_data_6	0.126	0.207	0.090	0	0	0	0	0	0
MBA_ECG14046_data_7	0.086	0.120	0.067	0.004	0.020	0.002	0.002	0.010	0.001
MBA_ECG14046_data_8	0.109	0.370	0.064	0.003	0.148	0.001	0.004	0.128	0.002
MBA_ECG14046_data_9	0.033	0.035	0.030	0	0	0	0	0	0
MBA_ECG14046_data_10	0.164	0.351	0.107	0.001	0.048	0.001	0.001	0.045	0.001
MBA_ECG14046_data_11	0.132	0.336	0.082	0.002	0.118	0.001	0.004	0.148	0.002
MBA_ECG14046_data_12	0.095	0.163	0.067	0	0	0	0	0	0
MBA_ECG14046_data_13	0.176	0.682	0.101	0.005	0.381	0.003	0.006	0.357	0.003
MBA_ECG14046_data_14	0.171	0.392	0.109	0.002	0.133	0.001	0.005	0.182	0.002
MBA_ECG14046_data_15	0.062	0.192	0.037	0.002	0.188	0.001	0.001	0.081	0.002
MBA_ECG14046_data_16	0.086	0.190	0.056	0.005	0.078	0.003	0.005	0.077	0.003
MBA_ECG14046_data_17	0.115	0.301	0.071	0.005	0.2	0.003	0.001	0.056	0.001
MBA_ECG14046_data_18	0.122	0.090	0.190	0	0	0	0.002	0.143	0.001
MBA_ECG14046_data_19	0.128	0.553	0.072	0.002	0.114	0.001	0.004	0.189	0.002
MBA_ECG14046_data_20	0.123	0.573	0.069	0.007	0.325	0.003	0.005	0.333	0.002
MBA_ECG14046_data_21	0.10	0.190	0.068	0	0	0	0	0	0
MBA_ECG14046_data_22	0.136	0.179	0.109	0	0	0	0	0	0
MBA_ECG14046_data_23	0.143	0.399	0.087	0.008	0.333	0.004	0.006	0.250	0.003
MBA_ECG14046_data_24	0.131	0.194	0.099	0.002	0.062	0.001	0.002	0.091	0.001
MBA_ECG14046_data_25	0.105	0.163	0.077	0.006	0.158	0.003	0.006	0.103	0.003
MBA_ECG14046_data_26	0.062	0.092	0.047	0.003	0.007	0.002	0.012	0.027	0.008
MBA_ECG14046_data_27	0.118	0.214	0.082	0.003	0.167	0.002	0.005	0.214	0.003
MBA_ECG14046_data_28	0.085	0.314	0.049	0.002	0.088	0.001	0.001	0.125	0.001
MBA_ECG14046_data_29	0.088	0.131	0.067	0	0	0	0.006	0.061	0.003
MBA_ECG14046_data_30	0.104	0.120	0.092	0	0	0	0	0	0
MBA_ECG14046_data_31	0.097	0.244	0.060	0.007	0.156	0.004	0.007	0.119	0.004
MBA_ECG14046_data_32	0.105	0.163	0.077	0.004	0.167	0.002	0.006	0.214	0.003
MBA_ECG14046_data_33	0.114	0.290	0.071	0.001	0.029	0.001	0	0	0
MBA_ECG14046_data_34	0.142	0.327	0.091	0.016	0.219	0.016	0.025	0.265	0.013
MBA_ECG14046_data_35	0.049	0.172	0.029	0.011	0.076	0.006	0.011	0.096	0.006
MBA_ECG14046_data_36	0.044	0.131	0.026	0.008	0.108	0.004	0.007	0.079	0.004
MBA_ECG14046_data_37	0.125	0.479	0.072	0.003	0.333	0.002	0.009	0.412	0.005
MBA_ECG14046_data_38	0.059	0.362	0.032	0.006	0.133	0.003	0.008	0.145	0.004
MBA_ECG14046_data_39	0.055	0.447	0.029	0.003	0.138	0.002	0.004	0.103	0.002
MBA_ECG14046_data_40	0.060	0.329	0.033	0.008	0.244	0.004	0.004	0.078	0.002
MBA_ECG14046_data_41	0.086	0.466	0.047	0.006	0.280	0.003	0.010	0.198	0.005
MBA_ECG14046_data_42	0.076	0.575	0.040	0.003	0.385	0.002	0.003	0.212	0.002
MBA_ECG14046_data_43	0.054	0.512	0.028	0	0.133	0	0.003	0.131	0.001
MBA_ECG14046_data_44	0.032	0.024	0.049	0	0	0	0	0	0
MBA_ECG14046_data_45	0.139	0.218	0.103	0.010	0.068	0.005	0.01	0.065	0.005
MBA_ECG14046_data_46	0.047	0.135	0.029	0.010	0.032	0.006	0.011	0.049	0.006
MBA_ECG14046_data_47	0.095	0.166	0.067	0	0	0	0.001	0.091	0.001

Table 4.3: Comparative evaluation of DQR-AD with two anomaly detection methods (LSTM-AD and Deep LSTM-AD) on 47 ECG time series. F_Score, Precision and Recall is reported in this table

	Datasets	AE ($h = 2$)	AE ($h = 10$)	DQR-AD
	SMAP1	0.88	0.89	0.98
	SMAP2	0.89	0.89	0.86
	SMAP3	0.88	0.88	0.95
	SMAP4	0.88	0.88	0.87
	SMAP5	0.88	0.88	0.95

Table 4.4: Comparative evaluation of DQR-AD and AE ($h = 2$, $h = 10$) on Five sMAP Datasets (SMAP1 to SMAP5). An average AUROC is reported on each dataset.

Online Anomaly Detection Method for Multivariate Sensor Streams

5.1 Introduction

Although deep learning-based anomaly detection methods used sequential models that enable to handle temporal nature of time series data, they are widely offline and used in stationary data. However, the dynamic nature of time series will affect the performance of deep learning-based anomaly detection methods due to changes in the distribution of data that result in concept drift. Concept drift means changes in the characteristics of data over time where the characteristic of the new data is different from the previous data [2, 56, 153]. For example, when the computer's software is updated or its configuration is changed, data such as CPU utilization and the speed of reading or writing data in the disk will change. In anomaly detection system, the definition of abnormal behavior often changes with the change in data characteristic. As such, anomaly detection methods should be able to adapt to the new data and redefine the meanings of abnormal behaviors to accurately detects anomalies in the new data. These challenges results in the need for online anomaly detection methods which are able to adapt to concept drift [2, 66, 131]. The goal of this chapter is to answer research question 2 that investigate how concept drift adaptation improve the performance of anomaly detection in time series. This is done by developing an online anomaly detection method that can detect anomalies in real time. To carryout this task, we start by reviewing the existing research works to identify a research problem in this area. We then analyse the problem and develop an online model that can identify anomalies in real time. An experiment is conducted using both real and synthetic data streams to evaluate the performance of the model. Finally, the result is analyze to compare the model performance with other state of the earth methods.

5.2 Literature Review

Researchers have proposed some new methods for online anomaly detection [43]. These methods are generally categorized into two based on the approach they used: First is to build a new online model using a single incremental learning algorithm that will be train and test in real time [109, 117] or a stationary model that is initially train with historical dataset which is later updated to capture the features of new incoming data instance [131]. While the second involves an ensemble learning theory [7, 43] that trains multiple individual model for different part of the data streams. Research in the literature have proved the performance of online ensemble learning in handling concept drift in data streams [120, 147]. The performance is due to their utmost predictive accuracy and stability-elasticity property. This property makes it easy for them to incorporate new data into the model, by training and adding new members to the ensemble, and naturally, forget irrelevant knowledge by removing the old members from the ensemble [44]. Despite this performance, they are slow and have higher computational cost. As such, the focus of this review will be on the first strategy that build online model for anomaly detection in data streams using the single incremental algorithm. The review is structured into paragraphs each highlighting our review on the methodology used for identifying anomalies in each technique.

A multivariate Gaussian model is combined with a probabilistic ratio test (Page's test) in [10] to estimate the likelihood of an incoming data to be anomalous by exceeding a fixed threshold. This is achieved by initially training the model offline to learn the data concept and determine model parameters corresponding to normal and abnormal concept. The model settings are then used in real-time to raise an alarm for any incoming abnormal data instance while periodically updating the model settings whenever there is change in data concept. The proposed method serves as first line of defence against unauthorize users in a network system but is faced with a challenge of using PCA which increase computation burden. The study in [94] define anomaly detection as binary classification problem and went ahead to developed an improved real-time learning model that classify a current data point as anomalous or not based on the previous time stamps. This is achieved using Multivariate Normal Distribution, K-Nearest Neighbour, and One Class Support Vector Machine algorithms. Although, this method used real-time learning algorithm, a fixed threshold is used for identification of anomaly. It is often difficult to define a precise threshold on real-time data where there is no sufficient amount of previous data. Similarly, the machine learning algorithms used especially KNN and SVM are highly computational when dealing with big data.

OCPC combined with similarity measure is used in [122] to detect anomalies in incoming data as any deviation from the normal data concept. The proposed technique involves three phases which includes training phase, detection phase, and updating phase for model training, detection and updating respectively. The method also used principal component analysis which is computationally attractable when deal-

ing with big data. Also, mean and standard deviation of both current and previous measurements are required for anomaly identification and updating the model parameters. Computations of mean and standard deviation is difficult in real time where there is no full data content. In a similar context, the study in [91] used time series similarity measure to identify causal event in DBMS that are similar in pattern with anomaly stat metrics. The proposed method used autoencoder to identify anomalies in non-stationary time series from DBMS metrics. In addition to anomaly detection, the method also identifies time periods containing anomalies and causal events related with anomalies within those periods. Apart from computational burden of using similarity measures, DBMS consultants are required to identify stat metrics they consider when finding anomaly periods. This often delay the process and become difficult when dealing with big data.

In order to handle sequential and temporal nature of time series, a Hierarchical Temporal Memory (HTM) is used for an online anomaly detection in data streams [2]. This method models the temporal nature of the data stream at a given time and make predictions for the next time step. At each step, the actual instance is compared with the predicted instance to compute anomaly score which is thresholded to determine whether the point is anomalous or not. The likelihood of a point to be anomaly is determined by assumption of Gaussian distribution on prediction error. In a similar context, a study in [78] proposed a decision support mechanism for outlier detection in the concept drifting environment. This is achieved by implementing resistance learning concept with envelop module using single layer feed-forward neural network. A sequence-based moving window is also used to demonstrate incremental learning process where incoming data streams are added into learning process while older ones are discarded. This will allow the model to adapt with the dynamic changes in the time series and differentiate them with outliers.

In a different approach, a deep learning-based anomaly detection method is proposed where RBF classifier is used to identify engine fault using real data analysis [130]. This is done by training an offline model whose parameters are updated in real-time to adapt with model uncertainties and dynamic changes caused by environment or mechanical wear of engine part. Fault detection is done using a Gaussian basic activation function that model the prediction error as normal distribution to identify faulty data when the prediction error exceeds a specified threshold. A more recent approach that used Recurrent Neural Network to detect anomalies in real time was proposed in [131]. This approach used instances of data that arrives continuously and trained the model incrementally thereby adapting to the changes that may occur in the data distribution. The authors used model prediction error to determine when to update the model based on the changes that occur in the data and whether those changes are anomalies or not. The predicted result is compared with the actual observed sequence and prediction error will be used for anomaly score computation thereby updating next step RNN model using BPTT. The likelihood of the predicted window to be anomaly is

determined by assumption of Gaussian distribution on prediction error. The challenge remains as to find out whether the incoming data distribution matches the normal distribution. However, data do not necessarily follow a clear distribution and defining or assuming a distribution in modelling step is often difficult or inappropriate.

Extreme Value Theory solve these problems by deducing the distribution of extreme values detected without making assumption on the original data distribution. As such, Siffer et al, proposed an approach that used Extreme Value Theory for outlier detection in streaming univariate time series [138]. This approach does not require manual setting of threshold or assumption of any distribution. In order to cope with data streams, a moving average is used which model the local behaviour is an efficient way to adapt to concept drift.

However, many of the above mention techniques does not take into account uncertainty in their predictions which may lead to over-confident predictions especially when there is limited training data [3]. Quantifying uncertainty is particularly important in critical applications such as clinical diagnosis [125], where a realistic assessment of uncertainty is important in determining disease status and appropriate treatment. Also, such approach is most welcome in anomaly detection applications requesting better-informed decisions and mitigate against false anomaly alerts.

An alternative approach is the use of prediction interval which is computed by taking into accounts the uncertainty in both the data and data driven model. Hill and Minsker proposed an anomaly detection method that combine model prediction and its corresponding prediction interval to detect anomalies in data streams [73]. This method provides a principle framework for selection of threshold by simply classifying a data point as anomalous or not by based on whether or not they fall outside a given PI. The type of prediction interval used is t-interval which relies on Student's t-distribution where the prediction levels guide the selection of the interval width. This demonstrates the benefit of using PI over an arbitrary threshold value. The method also used AD strategy for processing future data points after flagging an anomaly. AD was used because its long-term performance is unaffected by previous miss-classifications.

Similarly, Chebyshev's inequality which proves that the majority of the distribution values are clustered around the distribution mean, can also be used to define an interval for identification of anomalies. The study in [16] used Chebyshev inequality condition to define a prediction interval that provides the same anomaly detection result without any assumption of the distribution of the data. The proposed method combines the application of TEDA for fault detection in industrial process. It is density-based anomaly detection approach that analyses the density of each data sample by computing its distance from all other samples read so far. Similarly, Ferdowsi et al [48] avoids the use of fixed threshold for identification of anomalies and proposed an online outlier detection system that defined an interval using Chebyshev's inequality to declare outliers. As a result of dynamic changes in an online system which lead to change in the distribution of the data, the authors used a fixed time window and assumed the measurement in

each time window to have a fixed distribution. With this a data point at given time can be identified as outlier when it exceeds the interval defined using Chebyshev's inequality. In another approach, Reunanen et al, also utilized Chebyshev's inequality to identify anomalies in data streams [126]. This method combines Autoencoder and Logistic Regression for outlier detection and prediction in sensor data streams. The Autoencoder reconstruct the input data and produce hidden representation of the input that can be used to create the required labels for Logistic Regression to classify anomalous points. A data point is classified as anomalous when its reconstruction cost exceeds the expected reconstruction cost with three standard deviation which represents an upper bound of the inequality. Although no assumption of the distribution of the data is required but the method assume the descriptive statistics (μ and σ) of the unknown normal values to be initially defined.

An alternative approach is to use a boxplot and compute lower and upper quantiles that are used to calculate mean and standard deviation of the given window of data. Salem et al, [128] proposed an online anomaly detection method that used lower and upper quantiles in healthcare system to differentiate between faulty measurements from clinical deterioration. This approach combines DWT, NSHW, the Hampel filter, and box-plot. The DWT, NSHW, and the Hampel filter are used to detect spatial deviations, and the boxplot is used for temporal analysis in order to pinpoint suspicious underlying attributes, which are responsible for the detected deviation. In order to classify an incoming data point as anomaly, upper and lower bound are computed which are used directly to threshold the new data points for anomalies. Although this method handles the issue of using a fixed threshold, it still suffers from computational complexity of DWT which has to be combined with Hampel Filter to support sequential analysis of the data.

As such, deep learning algorithms combine with quantile regression are required to predict target value accompanied with its quantile values. In an attempt to demonstrate the performance of regression-based anomaly detection methods, van de Wiel et al [155], carried out experimental comparison real-time quantile regression-based anomaly detection methods for both univariate and multivariate time series data. All the anomaly detection methods used in the paper are quantile regression-based where the prediction of a target variable is accompanied with the quantile values. This reduces the burden of defining a fixed threshold and instead used a quantile interval for identifying anomalous data points. Their results shows that multivariate anomaly detection methods perform better out of which quantile regression-based methods was overall the best approach for time series anomaly detection. This impression motivates the research work in chapter 3 where we proposed a new method called DQR-AD that used quantile interval compared with a fixed threshold to identify anomalies in time series data [146]. Despite the performance of the proposed method compared with other anomaly detection methods, fixed threshold was used for identification of anomalies. However, selecting an appropriate threshold that differentiate anomalies with noise is

a difficult task. As such, using a single global threshold across the entire time series will lead to poor trade-off between true and false positive rate.

In summary, all the deep learning-based anomaly detection methods reviewed above faces either the challenge of assumption of data distribution, non-consideration of uncertainty in their predictions or use of fixed threshold for anomaly identification. In order to handle these challenges and dynamic nature of time series, we updated our proposed method in chapter 4 by incorporating feature extraction through combination of Autoencoder with LSTM prediction model. Prior to time series prediction, we first employ autoencoder that can extract meaningful representations from the time series. In this context, the representation learned will provide useful features for prediction and capture unusual input that is further propagated to the prediction model. Specifically, this chapter will proposed a new online anomaly detection method that incorporate feature extraction, uncertainty estimation, and concept drift adaptation to improve the performance of anomaly detection in time series data.

5.3 Proposed Approach

Let's consider a multivariate time series $x = x_1, x_2, \dots, x_t$, where t is the length of the time series and each point $x_i \in \mathbb{R}^m$ (for $i = 1 \dots t$) in the time series is an m -dimensional vector corresponding to the m features. In order to formulate the anomaly detection problem in an online setting, we used a sliding window to segment the time series with the aim to predict the next time step given a window of previous time steps. As such, a window of w previous time series points $h_w = x_{t-w+1}, \dots, x_{t-1}, x_t$ is used to forecast the next sequential time series point x_{t+1} . An observe point is then classified as anomaly if it deviates from its forecasted value. In summary, the proposed approach involves the following steps beginning at time t : (1) An autoencoder is use which takes h_w and reconstruct its representations as $r_w = e_{t-w+1}, \dots, e_{t-1}, e_t$ which is then pass as input to the regression model. (2) DQR model then takes the representations r_w as input to forecast upper and lower quantiles that corresponds to the expected value at time $t + 1$. (3) the upper and lower quantiles are used to calculate the quantile interval (QI) within which the actual point should lie. (4) when the current data point at time $t + 1$ arrives, it is compared with the QI and when it is outside the QI range, it is classified as anomalous otherwise classified as normal data point. (5) the anomaly likelihood is determine using Q-function. (6) when the likelihood is less than or equal to a threshold, the predicted mean \hat{x}_{t+1} is added to h_w using ADAM strategy, otherwise, the current data point x_{t+1} is added to the h_w using AD strategy to update DQR model. This will allow the model to learn new characteristics of the data and redefine the meanings of abnormal behaviors. (7) repeat steps 1 to 5. These steps are illustrated using a flowchart in fig. 5.1 The detail description of the process is given in Algorithm 1 which is explain in the following subsections.

5.3 Proposed Approach

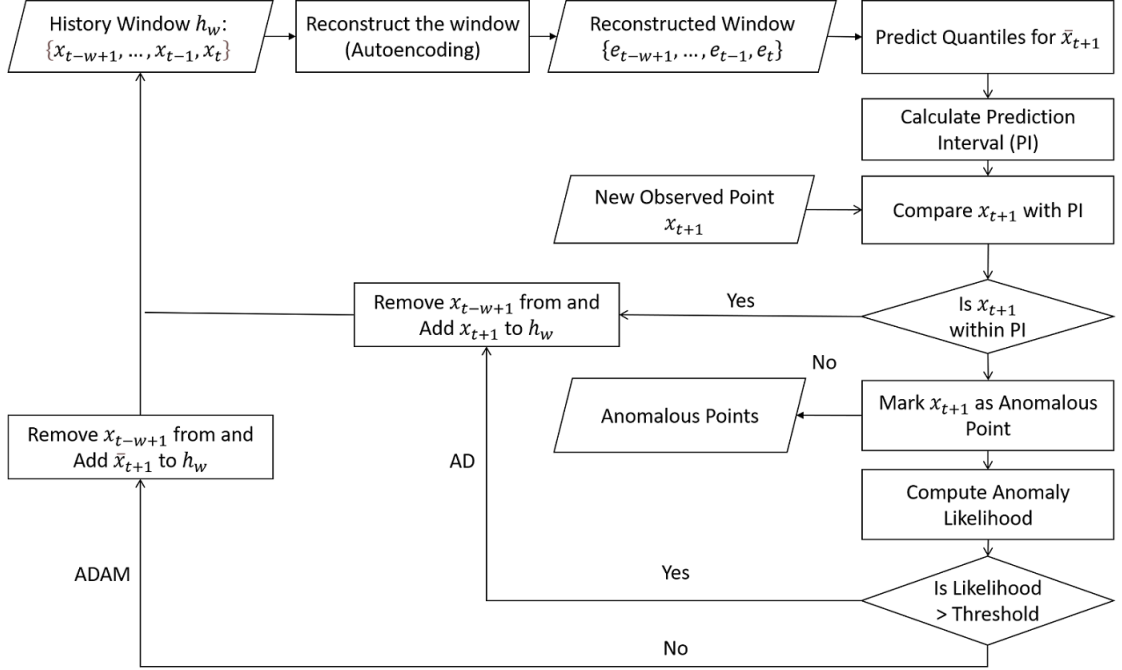


Figure 5.1: Schematic representation of proposed online DQR-AD method

5.3.1 Autoencoding

Prior to time series prediction, we first employ autoencoder that can extract meaningful representations from the time series. The Autoencoder model consists of an encoder and a decoder with two layers of LSTM each. The basic idea involves non-linear mapping of the input sequence to a fixed dimensional vector representation through an encoder which is then followed by another non-linear mapping of the vector representation back to the time series sequence using decoder. Specifically, the encoder read the history window h_w of t time steps and constructs a fixed-dimensional state from which the decoder constructs the following sequence of representations $e_{t-w+1}, \dots, e_{t-1}, e_t$ as illustrated in fig. 5.2. After the autoencoder perform the feature extraction, the DQR model then works on the reconstructed sequence received from the decoder to forecast the next time series point

5.3.2 Deep Quantile Regression (DQR)

We consider a regression model that can forecast next time step based on previous time steps to be suitable method for learning temporal characteristics of time series. However, the focus here is to develop a regression model which can forecast time series points considering uncertainty. This is achieve via quantile regression that estimate conditional quantiles as oppose to classical regression which estimate the conditional

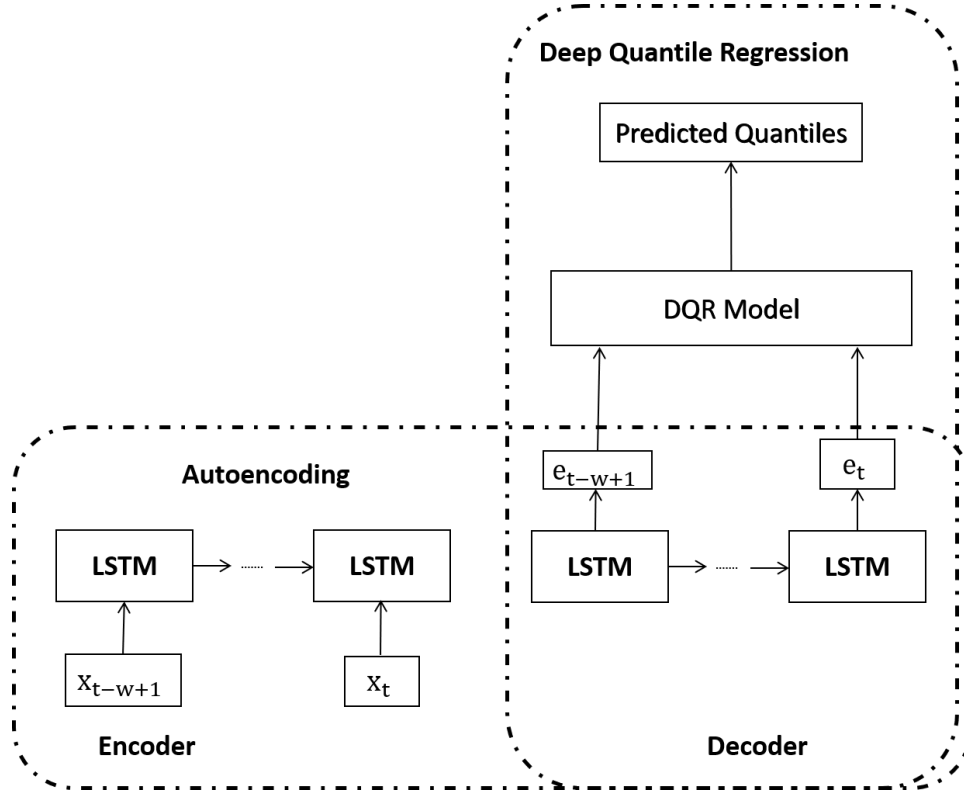


Figure 5.2: Neural network Architecture, with autoencoding phase using LSTM encoder-decoder, followed by DQR model whose input is reconstructed sequence from the autoencoding

mean of the respond variable [3]. The goal of quantile regression is to estimate conditional quantiles of interest which is applied in cases where parametric likelihood cannot be specified [168]. From the literature reviewed, it has been shown that quantile regression is able to captured aleatoric uncertainty [3]. As such we quantify uncertainty in our model estimates using conditional quantile regression which estimate multiple quantiles of interest. To perform quantile regression on our LSTM model, we replace the Gaussian likelihood term in the LSTM loss function with the quantile loss which penalizes errors based on the quantile values generated. In this context, we specifically estimate lower quantile (LQ) and upper quantile (UQ) which corresponds to the median and one standard deviation from the mean respectively. These LQ and UQ are then used to calculate the mean μ and variance σ that can be used to obtain the corrected p-values for anomaly detection. We also reduce the chance of quantile overlapping via bootstrapping which allow the regression model to be iterated n times, thereby storing the predicted values in an array that is finally used to compute the desired quantiles.

5.3.3 Online Anomaly Detection

In order to obtain a probabilistic threshold, we train the DQR model to estimate LQ and UQ. We adapted the approach in [3] where we assume the output of DQR to be Gaussian. As such, we can use the two estimated quantiles to fully characterize the Gaussian distribution. Specifically, DQR estimate 0.15-th and 0.5-th quantiles which corresponds to the median and one standard deviation from the mean respectively. These LQ and UQ are then used to calculate the mean (μ) and variance (σ) that can be used to obtain the corrected p-values. The p-values are used for anomaly detection task where a probabilistic threshold is set at an $\alpha = 0.05$ significance level. This means, we perform the anomaly detection task by estimating a 95% confidence interval where an approximate α level prediction interval is computed by $[\mu - z_{\alpha/2}\sigma, \mu + z_{\alpha/2}\sigma]$, where $z_{\alpha/2}$ is the upper $\alpha/2$ quantile of standard normal. The prediction interval will serve as the threshold for identifying whether the data point is anomalous or not. The new data point is classified as normal when it is within the prediction interval; otherwise, it is classified as anomaly. Details of these steps is given in algorithm 24. In order to reduce the number of false positives, the threshold is selected based on the corrected p-values calculated to control the False Discovery Rate (FDR) [3]. Specifically, we chose the threshold corresponding to an FDR corrected p-value of 0.05.

5.3.4 Concept Drift Adaptation

In order to adapt to concept drift, the model needs to be updated in incrementally as new data arrives. To achieve this, this chapter used anomaly likelihood which is computed using Q-function to define the abnormal degree of the current data point based on the previous data points. The likelihood of the data point at time t+1 is define as shown in equation 5.1:

$$L_{t+1} = \frac{Q(x_{t+1} - \mu_{t+1})}{\sigma_{t+1}} \quad (5.1)$$

Where x_{t+1} is the actual observe point at $t + 1$. The Q-function measures the abnormal degree of the data point relative to the previous data points. The smaller the value of the Q-function, the higher the abnormal degree of the current data point, and vice versa. As such, when the current data point at time t+1 is classified as an anomaly, L_{t+1} is computed which determines when the model will be updated to adapt to the new changes. This is achieved by comparing L_{t+1} with user-defined threshold ϵ . When $L_{t+1} < \epsilon$, the predicted mean μ_{t+1} is added to h_w using ADAM strategy, otherwise, the current data point x_{t+1} is added to the h_w using AD strategy. When x_{t+1} is added to the h_w , the earliest data point x_{t-w+1} is removed from h_w . Specifically, when concept drift occurs, our proposed method will mark the current data point as anomalous. However, when the abnormal behavior continues for a longer period of time, the abnormal degree of the current data point will be low compared to the previous data points using L_{t+1} .

Algorithm 4 Online Anomaly Detection

- 1: **Input:** Current time t , $h_w = x_{t-w+1}, \dots, x_{t-1}, x_t$, Window size w , threshold ϵ
 - 2: **Output:** list of anomalies
 - 3: anomaly $\leftarrow []$
 - 4: **Repeat**
 - 5: train the Autoencoder to produce reconstructed window sequence $r_w = e_{t-w+1}, \dots, e_{t-1}, e_t$ using history window h_w
 - 6: train the DQR model using the reconstructed sequence to predict mean μ_{t+1} as UQ and variance σ_{t+1} as LQ at time $t + 1$
 - 7: compute prediction interval $[\mu_{t+1} - z_{\alpha/2}\sigma_{t+1}, \mu_{t+1} + z_{\alpha/2}\sigma_{t+1}]$
 - 8: if data point x_{t+1} is within the prediction interval
 - 9: x_{t+1} is normal data point
 - 10: add x_{t+1} to h_w
 - 11: else
 - 12: x_{t+1} is anomalous data point
 - 13: add 1 to anomaly
 - 14: compute the anomaly likelihood L_{t+1} using equation (1)
 - 15: if $L_{t+1} > \text{threshold } \epsilon$
 - 16: add x_{t+1} to h_w
 - 17: else
 - 18: add \bar{x}_{t+1} to h_w
 - 19: end if
 - 20: end if
 - 21: remove x_{t-w+1} from h_w
 - 22: $t = t + 1$
 - 23: **until all test data points are processed**
 - 24: return anomaly
-

As such, the current data point is added to the h_w to retrain the DQR model. This will allow the DQR model to learn the new characteristics of the data and hence adapt to the concept changes thereby redefining the abnormal behavior.

5.4 Dataset Description

The proposed method is evaluated on two commonly used anomaly benchmark datasets in the literature. These datasets include; Yahoo Webscope dataset and Numenta Anomaly Benchmark (NAB) dataset.

5.4.1 Yahoo Dataset:

Yahoo Webscope dataset, is publically available dataset release by Yahoo Labs. The benchmark datasets have been widely used in literature to validate anomaly detection algorithms and determine the accuracy of various types of anomaly including outliers and change-points [110, 131]. The choice of this datasets is because of the availability of the anomaly labels that we can used to validate our model. The dataset consists of 367 real and synthetic time series with label anomaly points. The real datasets consist of time series with representing the metrics of various yahoo services. The synthetic datasets consists of time series that demonstrate trend, noise, and seasonality changes with anomalies presents at random positions. Each time series contains 1,420 to 1,680 instances. The benchmark dataset is further divided into four sub-benchmarks that include A1Benchmark, A2Benchmark, A3Benchmark, and A4Benchmark. In each dataset file, there is a Boolean attribute named “is_anomaly” with values 1 and 0 that indicate if the value at a particular time stamp is anomaly or not. Because we are using unsupervised learning approach in our evaluation, we drop the label attribute from each datasets. However, A3Benchmark and A4Benchmark contain additional attributes such as change-point, trend, noise, and seasonality that indicate the rate of changes in the data. In order to enhance the performance of our forecasting model, we used these additional attributes as additional external inputs to the model. The sample time series are depicted in Fig. 5.3 to 5.6

5.4.2 NAB Dataset:

NAB (Numenta Anomaly Benchmark) [90] is another publically available streaming anomaly detection benchmark dataset release by Numenta and available in their repository¹. We specifically used seven time series from the NAB benchmark dataset which includes ambient temperature in an office setting, CPU utilization from Amazon Web

¹<https://github.com/numenta/NAB>

5.4 Dataset Description

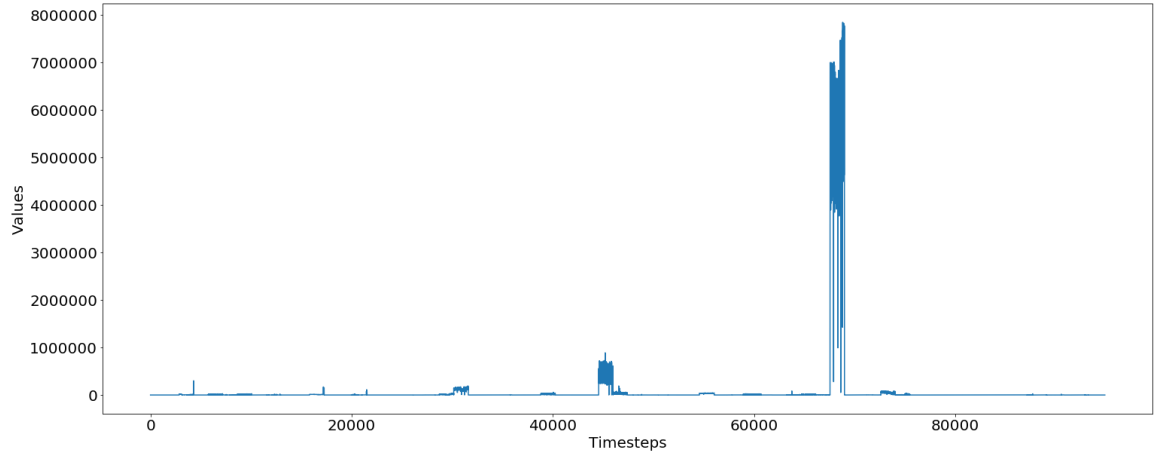


Figure 5.3: A1Benchmark time series Sample

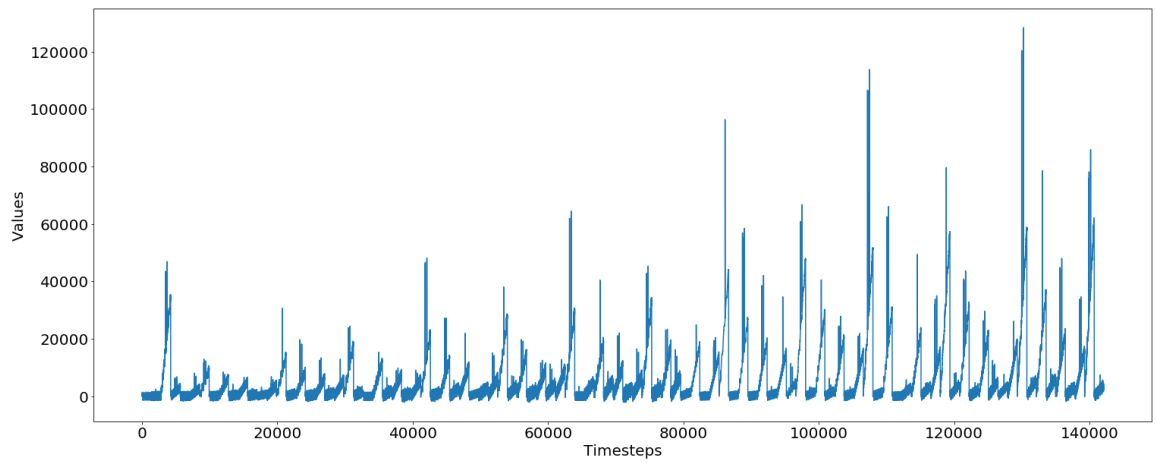


Figure 5.4: A2Benchmark time series Sample

5.4 Dataset Description

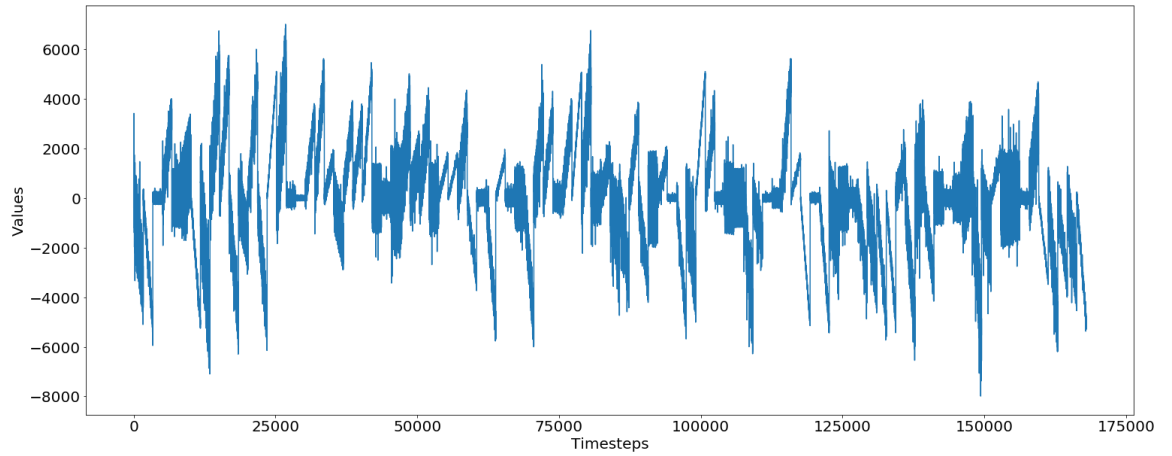


Figure 5.5: A3Benchmark time series Sample

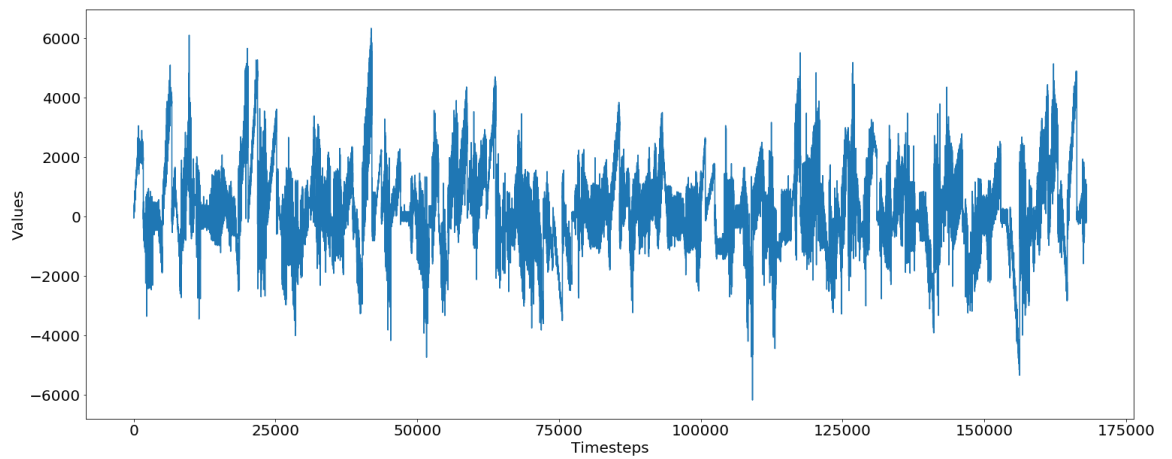


Figure 5.6: A4Benchmark time series Sample

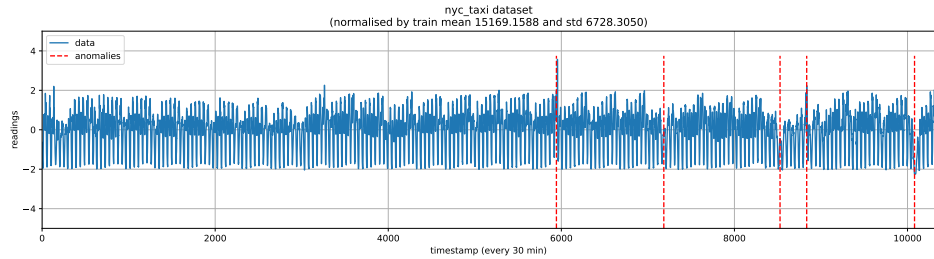


Figure 5.7: NYC_Taxi Sample time series with Anomalies

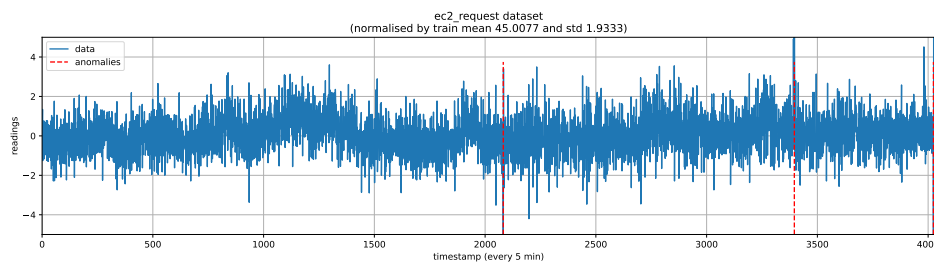


Figure 5.8: EC2 Request Sample time series with Anomalies

Services (AWS), CPU usage data from a server in Amazon’s East Coast datacentre, internal temperature of large industrial machine, number of NYC taxi demands in New York City, timing of key holds and key strokes of several computer users. Each time series file contains two columns of time stamps and data values. We created an additional column that hold anomalous label for each data point. The labels are created either based on the known root cause or as a result of labelling procedures defined in [2]. Some of the time series samples with anomalous points are depicted in Fig. 5.7 and 5.8.

5.5 Experiments

This section describes experiments conducted to evaluate the performance of the proposed online DQR-AD method. We evaluate the performance of the method via comparison with other state-of-the-art anomaly detection methods. Specifically, we compare our proposed method with VAE-LSTM [96], NumentaTM [2], online RNN-AD [131], data-driven-AD [73], AE-AD [126], SGP-Q [66], and DQR-AD [146]. The section starts with model training, followed by anomaly detection and ends with result and discussion.

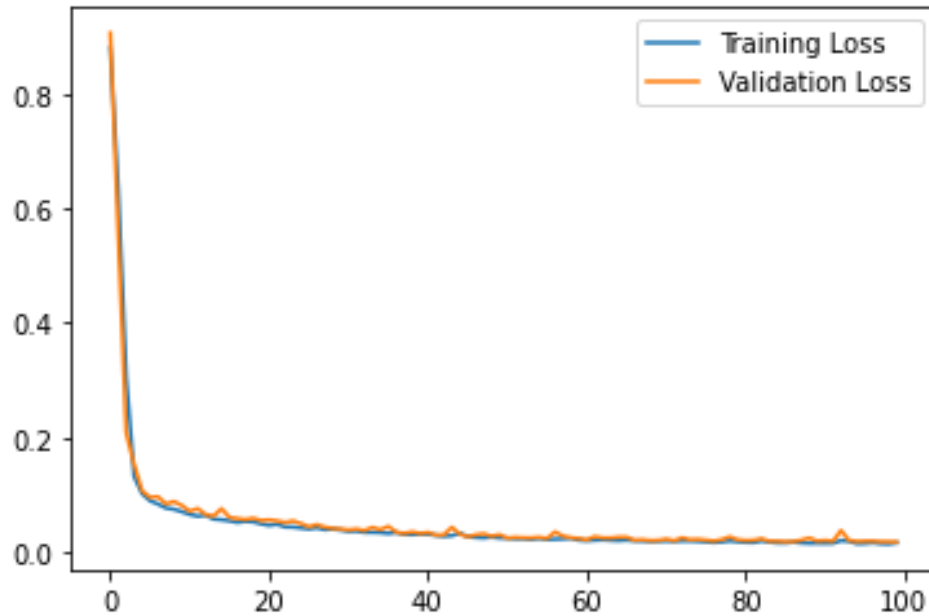


Figure 5.9: Training and validation loss

5.5.1 Model Training

To train all the models for time series anomaly detection, records of each time series are organized in 80% training set and 20% test set. From the training set, 20% of it will be used for validation. Fig.5.9. illustrate the training and validation loss behaviour on the nyc_taxi dataset. A sliding window technique is used to segment both training, testing and validation sets into 24 hours samples of history h_w and prediction h_w windows. The size of the sliding window is set up by looking at the data and notice the presence of hourly pattern in all the time series. Each input to the model is scaled between 0 and 1 for numerical stability during model training. The h_w is pass as input to the autoencoder which produced reconstructed sequence r_w . Figures 5.10 and 5.11 shows the sample of actual and reconstructed sequences for nyc_taxi dataset respectively. The DQR model will then takes the reconstructed sequence as input and predicts the quantile values. An autoencoder is constructed with two layers of LSTM cells that comprises 128 and 32 hidden states for both encoder and decoder. Deep quantile regression model is constructed using LSTM with two fully connected layers of 64 and 16 hidden units with adam activation function. We also train each models using mini-batch gradient descent where a batch size of 128 is used and 100 number of epochs. For DQR-AD and online DQR-AD, we make use of bootstrapping by reactivating the dropout with value 0.5 and iterate for 100 times, thereby storing the predicted values in an array which is finally used to compute the desired quantiles.

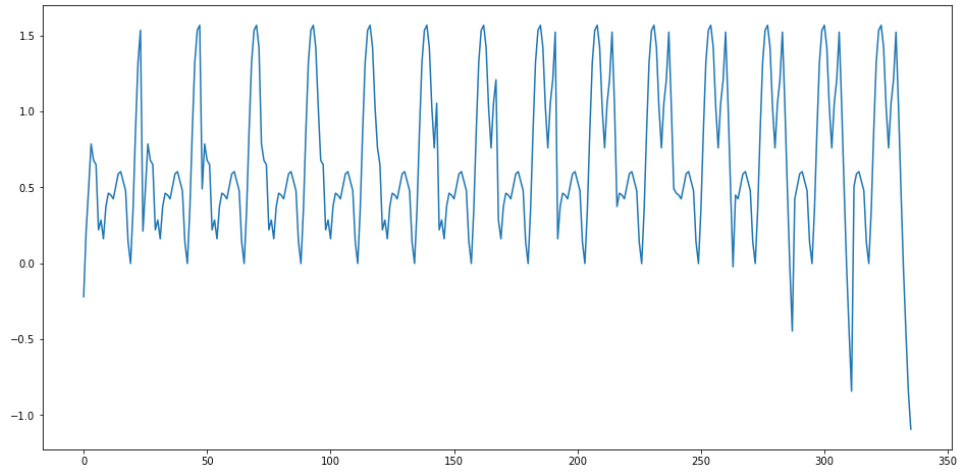


Figure 5.10: Sample of actual sequence from nyc_taxi dataset

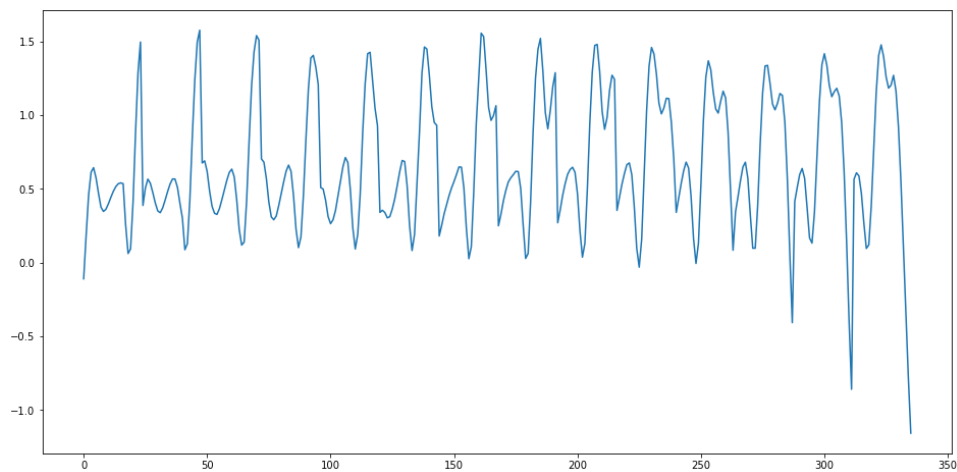


Figure 5.11: Sample of reconstructed sequence from nyc_taxi dataset

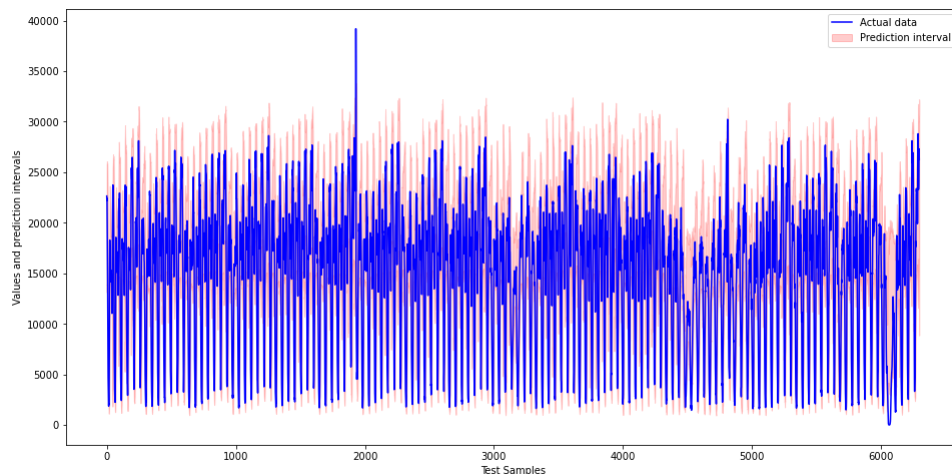


Figure 5.12: Predicted test set sample from nyc_taxi data. Actual values are shown in blue with prediction interval shown in orange

5.5.2 Anomaly Detection

The trained DQR model estimates 0.15-th and 0.5-th quantiles which corresponds to the median (μ) and one standard deviation (σ) from the mean respectively. These mean (μ) and variance (σ) can be used to obtain the corrected p-values. The p-values are used for anomaly detection task where a probabilistic threshold is set at an $\alpha = 0.05$ significance level. This means, we perform the anomaly detection task by estimating a 95% confidence interval where an approximate α level prediction interval is computed by $[\mu - z_{\alpha/2}\sigma, \mu + z_{\alpha/2}\sigma]$, where $z_{\alpha/2}$ is the upper $\alpha/2$ quantile of standard normal. Fig. 5.12 shows the actual data plot against the prediction interval for nyc_taxi test set. The prediction interval will serve as the threshold for identifying whether the data point is anomalous or not. The new data point is classified as normal when it is within the prediction interval; otherwise, it is classified as anomaly.

5.5.3 Experimental Results and Discussions

In order to evaluate the prediction performance of the proposed model on the basis of uncertainty estimation, we compare the performance of our proposed model that combine both LSTM and Autoencoder (Model1) with a model that have a single LSTM Forecaster (Model2). The same procedure was applied for evaluation and comparison of both models. The models were evaluated by iterating each of them 100 times thereby computing and storing the prediction score at each iteration using Mean Absolute Error (MAE). With the scores stored, the mean, standard deviation, and relative uncertainty of MAE is the computed. Table 5.1 reports the MAE and related confidence of both models using validation set. In addition, the result shown in Table 5.1 are also depicted

in Fig. 5.13 to 5.16 for the respective datasets. As it can be observed from the table

Datasets	Models	MAE	MAE Confidence
A1Benchmark	Model1	0.0385	0.0007
	Model2	0.0696	0.0011
A2Benchmark	Model1	0.0296	0.0006
	Model2	0.0363	0.0007
A3Benchmark	Model1	0.0268	0.0002
	Model2	0.0294	0.0006
A4Benchmark	Model1	0.0313	0.0002
	Model2	0.0336	0.0006

Table 5.1: MAE and its Relative Confidence

and figures, our model have overall improvement of at least 3% in prediction accuracy with a similar degree of confidence in all the datasets. With this result, we can assert that LSTM Autoencoder in the proposed model improve the prediction performance by extracting important unseen features from time series.

On the other hand, to evaluate the performance of the proposed online anomaly detection method in terms of false-positive rate and concept drift adaptation, we conducted two levels of the same experiment. On the first level, we compare online DQR-AD with VAE-LSTM [96] and DQR-AD [146] using precision, recall, and F-score 5.2 which are the most commonly used metrics to evaluate the performance of anomaly detection methods in terms of false-positive rate [146].

$$F_{score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (5.2)$$

Table 5.2 shows the metrics scores of each of the three methods on the seven time series data used. It can be observed from this table that online DQR-AD has good performance in terms of f-score where out of seven datasets, online DQR-AD is minimally 18% better than both DQR-AD and VAE-LSTM on five datasets as indicated in bold. Although, DQR-AD and VAE-LSTM have better precision and recall respective in most of the datasets, online DQR-AD obtained relatively good score in both precision and recall. This demonstrates that online DQR-AD is capable of identifying large number of true anomalies with a smaller number of false positives. On the second level, the performance of online DQR-AD is evaluated in terms concept drift adaptation. The proposed method is compared with other online anomaly detection methods that includes NumentaTM [2], online RNN-AD [131], data-driven-AD [73], AE-AD

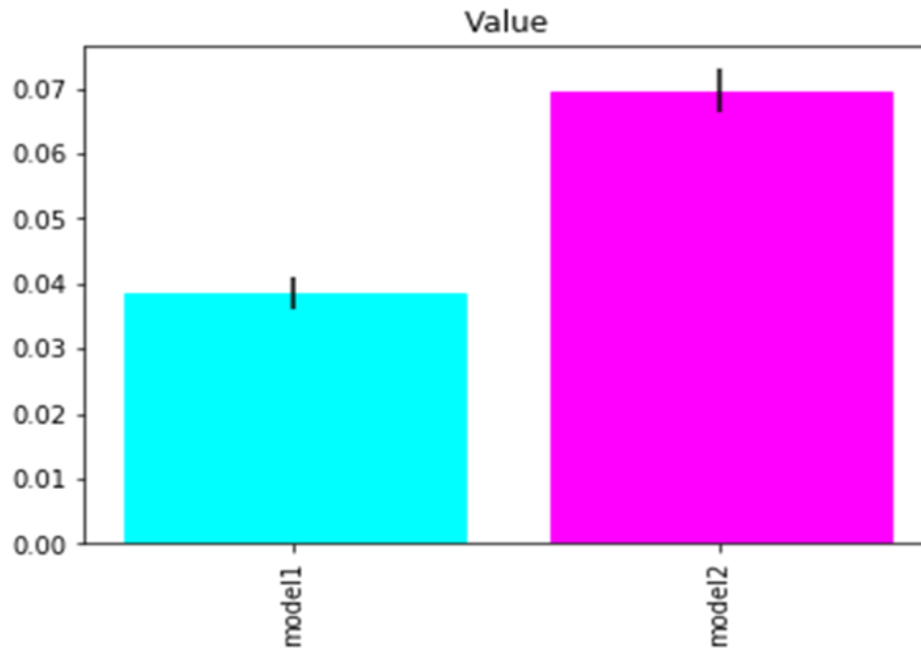


Figure 5.13: MAE and Relative Uncertainty for A1Benchmark

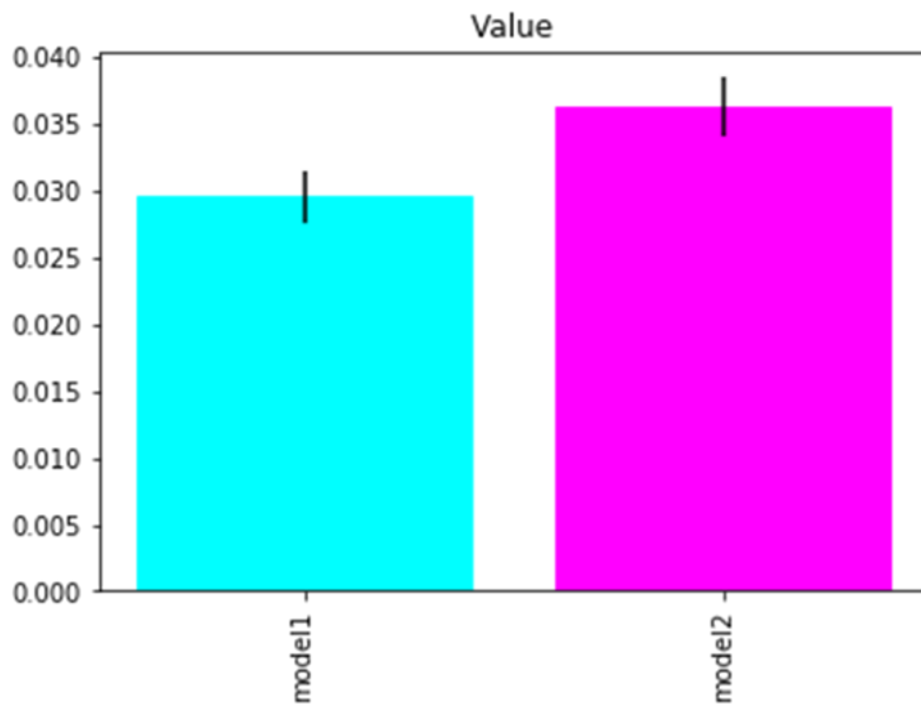


Figure 5.14: MAE and Relative Uncertainty for A2Benchmark

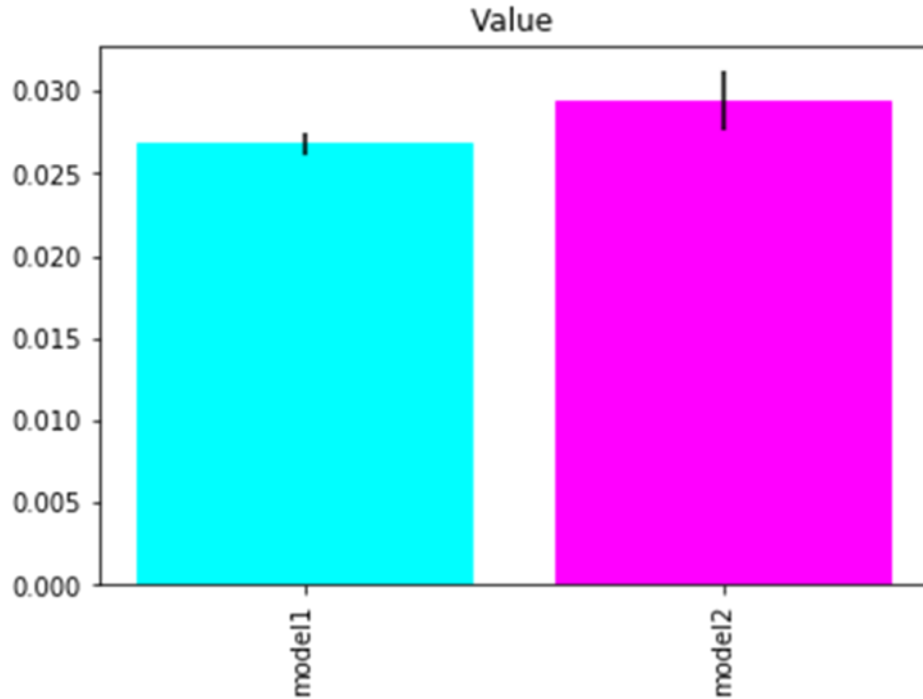


Figure 5.15: MAE and Relative Uncertainty for A3Benchmark

[126], and SGP-Q [66]. In this level, the performance of these methods is reported using area under the receiver operating characteristic curve (AUC) as shown in Table 5.3 with best performance result indicated in bold.

NAB Dataset	DQR-AD			VAE-LSTM			Online DQR-AD		
	F-Score	Precision	Recall	F-Score	Precision	Recall	F-Score	Precision	Recall
nyc_taxi	0.49	1	0.33	1	1	1	0.87	0.98	0.78
ambient_temperature	0.11	1	0.06	0.89	0.81	1	0.99	0.98	1
cpu_utilization	0.67	1	0.50	0.81	0.69	1	0.98	0.97	0.99
ec2_request_latency	0.82	1	0.70	0.99	0.99	1	0.99	0.99	1
machine_temperature	0.50	0.50	0.50	0.72	0.56	1	0.99	0.99	1
rogue_agent_key_hold	0.09	1	0.05	0.03	0.02	0.10	0.85	1.00	0.74
rogue_agent_key_updown	0.17	0.50	0.10	0	0.11	0	0.96	0.99	0.93

Table 5.2: Comparative evaluation of Online DQR-AD with two other anomaly detection methods (DQR-AD and VAE-LSTM) on 7 Real Known Cause Anomaly time series. F-Score, Precision and Recall is reported in this table

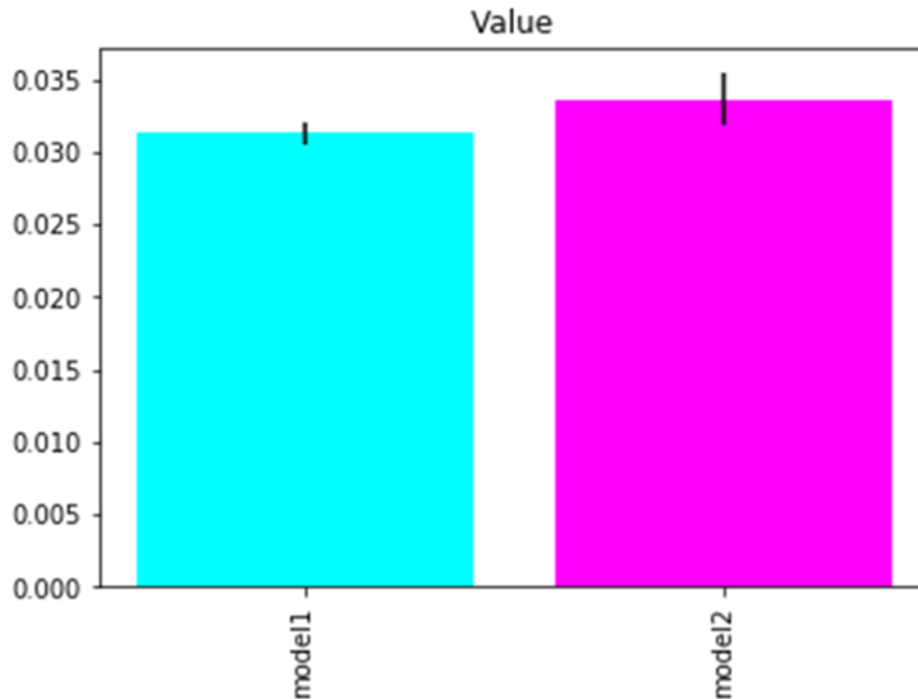


Figure 5.16: MAE and Relative Uncertainty for A4Benchmark

5.6 Summary

This chapter proposed a new online anomaly detection method that incorporate feature extraction, uncertainty estimation, and concept drift adaptation to improve the performance of anomaly detection in time series data. This is achieved by combining autoencoder with quantile regression model to estimates lower and upper quantiles which are used to threshold the input sequence for anomalies. In addition, an anomaly likelihood is computed using Q-function that is used to update the model parameters for concept drift adaptation. Initial experiment was conducted to evaluate the model prediction performance on the basis of uncertainty estimation. The proposed model that combine both LSTM and Autoencoder (Model1) is compared with a model that have a single LSTM Forecaster (Model2). Experimental results shows the proposed model have overall improvement of at least 3% in prediction accuracy with a similar degree of confidence in all the datasets. This demonstrates an increase in the prediction performance as a result of feature extraction. On the other hand, two level of experiments were conducted to evaluate the performance of online DQR-AD on the basis of false-positive rate and concept drift adaptation. In the first level of the experiment, online DQR-AD is evaluated and compared with DQR-AD and VAE-LSTM anomaly detection methods using seven time series from the NAB datasets. Results in Table 5.2

5.6 Summary

NAB Dataset	NumentaTM	Online RNN-AD	Data-driven-AD	AE-AD	SGP-Q	Online DQR-AD
nyc_taxi	0.74	0.89	0.48	0.61	0.42	0.97
ambient_temperature	0.83	0.83	0.56	0.66	0.88	0.98
cpu_utilization	0.54	0.71	0.80	0.73	0.55	0.72
ec2_request_latency	0.70	0.89	0.72	0.71	0.90	0.99
machine_temperature	0.33	0.55	0.50	0.60	0.54	0.71
rogue_agent_key_hold	0.43	0.33	0.22	0.50	0.48	0.64
rogue_agent_key_updown	0.40	0.24	0.23	0.42	0.38	0.59

Table 5.3: Comparative evaluation of Online DQR-AD with five other online anomaly detection methods (NumentaTM, Online RNN-AD, Data-driven-AD, AE-AD, and SGP-Q) applied to Real Known Cause anomaly dataset using AUC

shows that online DQR-AD has good performance in terms of f-score where online DQR-AD is 18% better than both DQR-AD and VAE-LSTM on five datasets. In the second level of the experiment, we demonstrate the performance of online DQR-AD on concept drift adaptation where it is compared with five other online anomaly detection method. Results in Table 5.3 shows that online DQR-AD method has better performance than its counterpart methods with relatively 10% margin on six out of the seven datasets. This result demonstrates how feature extraction and concept drift adaptation strategies adopted in the proposed online DQR-AD improve the performance of anomaly detection in time series.

Conclusions

At initial stage of this research work, a preliminary study was conducted to investigate concept drift and its corresponding challenges on online learning algorithms. As a result of the preliminary study, it was discovered that online ensemble methods are the most popular approach used in handling concept drift due to their stability-elasticity property that makes it easy for them to incorporate new data into the model, by training and adding new members to the ensemble, and naturally, forget irrelevant knowledge by removing the old members from the ensemble. In chapter 3 an experimental comparison and performance evaluation was conducted for online ensemble classifiers using both real and synthetic data streams. We explore ensemble methods because they are the most popular approach used in handling concept drift due to their stability-elasticity property that makes it easy for them to incorporate new data into the model, by training and adding new members to the ensemble, and naturally, forget irrelevant knowledge by removing the old members from the ensemble. Literature shows two main ensemble approaches that are used for this purpose. They include Chunk-based approach (that involves adding a new classifier that is trained on a recently arrived chunk of data) and Online-based ensemble approach (that involves updating the base classifier in the ensemble). The experiment is conducted for comparison and performance evaluation of these two ensemble classifiers using both real and synthetic data streams. The goal of this experiment is to compare and evaluate the performance of these algorithms towards a different type of concept drift in data streams. Experimental results obtained shows that online ensemble approaches perform better than chunk-based approaches and specifically, tree-based ensemble learners having the highest performance for learning in a non-stationary streaming environment with concept drift and class imbalance problems. The work was published in the 37th SGAI international conference on artificial intelligence Cambridge UK in December 2017 [147].

Later on it was discovered that time series anomaly detection receives increasing

research interest given the growing number of data-rich application domains delivered by recent technologies (e.g. Sensors and IoT) in industry 4.0. Recent additions to anomaly detection methods in research literature include deep learning algorithms. The nature and performance of these algorithms in sequence analysis enables them to learn hierarchical discriminates features and time series temporal nature. However, their performance is affected by usually assuming a Gaussian distribution on the prediction error, which is either ranked, or threshold to label data instances as anomalous or not. An exact parametric distribution is often not directly relevant in many applications though. This will potentially produce faulty decisions from false anomaly predictions due to high variations in data interpretation. The expectations are to produce outputs characterized by a level of confidence. Thus, implementations need the prediction interval (PI) that quantify the level of uncertainty associated with the model forecasts, which helps in making better-informed decision and mitigates against false anomaly alerts. Similarly, in big data context, the speed and dynamic nature of time series will affect the performance of deep learning-based anomaly detection methods due to changes in the distribution of data that result in concept drift. Concept drift means changes in the characteristics of data over time where the characteristic of the new data is different from the previous data. In anomaly detection system, the definition of abnormal behaviour often changes with the change in data characteristic. As such, anomaly detection methods should be able to adapt to the new data and re-define the meanings of abnormal behaviours to accurately detects anomalies in the new data. This underscores the importance and application of current deep learning-based anomaly detection methods in many application domains.

In an attempt to contribute to the mitigating challenges of uncertainty estimation and concept drift adaptation in deep learning-based anomaly detection methods, the following three novel research questions RQ1, RQ2, and RQ3 were respectively proposed as; Will estimation of uncertainty in prediction using quantile regression increase the prediction performance and reduced false anomaly alerts? Will obtaining probabilistic threshold using confidence interval reduces the trade-off between true and false positive rate? Does feature extraction and concept drift adaptation improve the performance of anomaly detection in sensor data streams thereby reducing the rate of false positive alerts? This study applied current state-of-the-art machine learning techniques and developed novel methods and algorithms for anomaly detection in time series. Answers to these questions are important to some critical applications such as clinical diagnosis, weather forecast, and automotive system.

RQ1 was addressed in chapter 4 by exploring the use of uncertainty and prediction interval to improve prediction performance of deep learning model and reduce false positive rate in anomaly detection. More precisely, an unsupervised anomaly detection method called DQR-AD was proposed that used quantile regression for anomaly detection in time series. Three algorithms were developed for each component of the proposed method that includes time series segmentation, prediction, and anomaly de-

tection. The algorithms are implemented and experiments were conducted in two different parts using both real and synthetic datasets. In the first part of the experiment, DQR-AD is evaluated on the NAB benchmark dataset containing 58 real and synthetic time series and compared with 6 other prediction-based anomaly detection methods that assumes normal distribution on prediction or reconstruction error for identification of anomalies. The experimental results obtained indicates that DQR-AD obtained relatively better precision than all other methods. This demonstrate DQR-AD is capable of detecting higher number of anomalous points with low false positive rates. In terms of f-score, the results obtained shows DQR-AD to be approximately 2 – 3 times better than the DeepAnT which perform better than all the remaining methods on all domains in the NAB dataset. This demonstrates the proposed approach can be practically applied on the time series with large amount of unlabeled data. In the second part of the experiment, DQR-AD have 10% better performance than AE on three datasets (SMAP1, SMAP3, and SMAP5) and equal performance on the remaining two datasets (SMAP2 and SMAP4) with relatively higher level of noise. The used of sMAP time series with 4 dimensional features demonstrate the applicability of DQR-AD on multivariate time series data. The initial work in chapter 4 was presented in the 3rd Annual Innovative Engineering Research Conference (AIERC 2019) which is a local conference organized by the faculty for postgraduate students. The extended work was also published in computing conference 2020 [145] which was extended to a journal paper that was publish in September 2021 with springer nature computer science [146].

RQ2 was actually a consequence of RQ1, the proposed method in chapter 4 used fixed threshold to identify abnormal data points. However, selecting an appropriate threshold that differentiate anomalies with noise is a difficult task. As such, using a single global threshold across the entire time series will lead to poor trade-off between true and false positive rate. RQ2 was address in chapter 5 where a new anomaly detection approach was proposed that directly used the estimated lower and upper quantiles to threshold the input sequence for anomalies. Specifically, the anomaly detection is done by directly estimating a 95% confidence interval. Experimental results obtained using NAB benchmark dataset demonstrates better performance of the proposed approach with good trade-off between true and false positive rate.

RQ3 was also addressed in chapter 5 where a new online anomaly detection method was proposed that incorporate feature extraction, uncertainty estimation, and concept drift adaptation to improve the performance of anomaly detection in time series data. In order to perform feature extraction and uncertainty estimation, autoencoder was combined with quantile regression model to estimate quantile values. Concept drift adaptation was achieved by computation of anomaly likelihood using Q-function which define the abnormal degree of the current data point based on the previous data points. The likelihood was used to update the model parameters to adapt to the changes observed aver a significant period of time. An algorithm was developed that carried out these tasks which is implemented and evaluated using both real and synthetic datasets.

Two level of experiments were conducted to evaluate the performance of the proposed online DQR-AD on the basis of uncertainty estimation and concept drift adaptation. In the first level of the experiment, online DQR-AD is evaluated and compared with DQR-AD and VAE-LSTM anomaly detection methods using seven time series from the NAB datasets. Results obtained shows that online DQR-AD has good performance in terms of f-score where online DQR-AD is 18% better than both DQR-AD and VAE-LSTM on five datasets. In the second level of the experiment, we demonstrate the performance of online DQR-AD on concept drift adaptation where it is compared with five other online anomaly detection method. Results obtained shows that online DQR-AD method has better performance than its counterpart methods with relatively 10% margin on six out of the seven datasets. The results at this level demonstrates how feature extraction and concept drift adaptation strategies adopted in the proposed online DQR-AD improve the performance of anomaly detection in time series. This research work has been submitted to journal of Applied Intelligence (APIN) for consideration.

Our journal paper that was published with Springer Nature has been recognize by researchers in the area of time series analysis. We received a congratulatory email and had a series of communications with Professor Eamonn Keogh ¹ who is the founder of UCR time series archive ². In his emails, he notify us about their arguments that "Current time series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress" [165]. As such, the authors suggested that the existing benchmark datasets to be abandoned and back-up their arguments, by creating a new anomaly detection benchmark dataset in their archive ³. When creating the benchmark dataset, they consider each dataset to have only one anomaly which is range of values that occur over a period of time. In addition, the threshold issue is remove from the dataset and anomalies can appear any where within the dataset. In another research, Tatbul et al [149] argued that real-world anomalies are range-based that occur over a period of time as opposed to point-based anomalies. As such, the classical precision and recall that is used for evaluating the performance of point-based anomaly detection methods will not be sufficient for range-based anomalies. As a result of that, the authors proposed range-based precision and recall to evaluate the accuracy of time series anomaly detection methods. These arguments motivated us to carryout some experiments that will evaluate the performance of our proposed method using the new benchmark dataset. The experimental results shows how our method was able to identified anomalies existing in the dataset. Appendix A shows figures that visualizes the detected anomalies against the actual anomalies.

Base on the two arguments and the outcomes of the experiments we conducted using UCR benchmark dataset, this study recommends the following future research

¹<https://www.cs.ucr.edu/~eamonn/>

²<https://www.cs.ucr.edu/~eamonn/time-series-data-2018/>

³<https://www.cs.ucr.edu/~eamonn/time-series-data-2018/UCR-TimeSeriesAnomalyDatasets2021.zip>

works: i) Identify the suitable metrics for performance evaluation of time series anomaly detection methods. ii) An extensive performance evaluation of the proposed approach in this study and other state-of-the-art anomaly detection methods on the new UCR anomaly benchmark datasets.

Regarding model adaptation to sudden changes in the incoming data (faulty sensor), Our proposed online online anomaly detection method is limited to adapting any incoming fault data that continue for a longer period as a new normal. As such, this study recommends a future work that will detect a continuous faulty data from faulty sensor as an anomaly. In addition, computational complexity of the algorithms will be considered in the future as one of the metrics to be use for performance evaluation.

References

- [1] N. Abe, B. Zadrozny, and J. Langford, “Outlier detection by active learning,” in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 504–509.
- [2] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, “Unsupervised real-time anomaly detection for streaming data,” *Neurocomputing*, vol. 262, pp. 134–147, 2017.
- [3] H. Akrami, A. Joshi, S. Aydore, and R. Leahy, “Quantile regression for uncertainty estimation in vaes with applications to brain lesion detection,” in *International Conference on Information Processing in Medical Imaging*. Springer, 2021, pp. 689–700.
- [4] M. S. alDosari, “Unsupervised anomaly detection in sequences using long short term memory recurrent neural networks,” Ph.D. dissertation, 2016.
- [5] T. Amarbayasgalan, B. Jargalsaikhan, and K. Ryu, “Unsupervised novelty detection using deep autoencoders with density based clustering,” *Applied Sciences*, vol. 8, no. 9, p. 1468, 2018.
- [6] A. Amini, W. Schwarting, A. Soleimany, and D. Rus, “Deep evidential regression,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 14 927–14 937, 2020.
- [7] S. Ando, T. Thanomphongphan, Y. Seki, and E. Suzuki, “Ensemble anomaly detection from multi-resolution trajectory features,” *Data mining and knowledge discovery*, vol. 29, no. 1, pp. 39–83, 2015.
- [8] F. Angiulli and C. Pizzuti, “Fast outlier detection in high dimensional spaces,” in *European conference on principles of data mining and knowledge discovery*. Springer, 2002, pp. 15–27.

-
- [9] D. B. Araya, K. Grolinger, H. F. ElYamany, M. A. Capretz, and G. Bitsuamlak, "Collective contextual anomaly detection framework for smart buildings," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 511–518.
- [10] B. Balasingam, P. Mannaru, D. Sidoti, K. Pattipati, and P. Willett, "Online anomaly detection in big data: The first line of defense against intruders," in *Data Science and Big Data: An Environment of Computational Intelligence*. Springer, 2017, pp. 83–107.
- [11] W. Bao, Q. Yu, and Y. Kong, "Evidential deep learning for open set action recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 349–13 358.
- [12] J. Basak, M. Keller, L. S. Mignet, and S. Roy, "Determining a window size for outlier detection," Mar. 29 2011, uS Patent 7,917,338.
- [13] S. R. Beeram and S. Kuchibhotla, "Time series analysis on univariate and multivariate variables: a comprehensive survey," *Communication Software and Networks*, pp. 119–126, 2021.
- [14] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.
- [15] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.
- [16] C. G. Bezerra, B. S. J. Costa, L. A. Guedes, and P. P. Angelov, "An evolving approach to unsupervised and real-time fault detection in industrial processes," *Expert systems with applications*, vol. 63, pp. 134–144, 2016.
- [17] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.
- [18] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2010, pp. 135–150.
- [19] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 139–148.

REFERENCES

- [20] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, “Moa: Massive online analysis, a framework for stream classification and clustering,” in *Proceedings of the First Workshop on Applications of Pattern Analysis*. PMLR, 2010, pp. 44–50.
- [21] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
- [22] L. Bontemps, J. McDermott, N.-A. Le-Khac *et al.*, “Collective anomaly detection based on long short-term memory recurrent neural networks,” in *International Conference on Future Data and Security Engineering*. Springer, 2016, pp. 141–152.
- [23] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “Lof: identifying density-based local outliers,” in *ACM sigmod record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.
- [24] D. Brzezinski and J. Stefanowski, “Reacting to different types of concept drift: The accuracy updated ensemble algorithm,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 81–94, 2013.
- [25] D. Brzezinski and J. Stefanowski, “Combining block-based and online methods in learning ensembles from concept drifting data streams,” *Information Sciences*, vol. 265, pp. 50–67, 2014.
- [26] D. Brzezinski and J. Stefanowski, “Prequential auc for classifier evaluation and drift detection in evolving data streams,” in *International Workshop on New Frontiers in Mining Complex Patterns*. Springer, 2014, pp. 87–101.
- [27] D. Brzezinski and J. Stefanowski, “Prequential auc: properties of the area under the roc curve for data streams with concept drift,” *Knowledge and Information Systems*, vol. 52, no. 2, pp. 531–562, 2017.
- [28] R. Chalapathy and S. Chawla, “Deep learning for anomaly detection: A survey,” *arXiv preprint arXiv:1901.03407*, 2019.
- [29] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.
- [30] V. Chandola, D. Cheboli, and V. Kumar, “Detecting anomalies in a time series database,” ... *Department, University of Minnesota, Tech. Rep*, p. 12, 2009.
- [31] C. Chatfield, *The analysis of time series: an introduction*. Chapman and Hall/CRC, 2003.

- [32] S. Chauhan and L. Vig, "Anomaly detection in ecg time signals via deep long short-term memory networks," in *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2015, pp. 1–7.
- [33] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 1–6, 2004.
- [34] Y. Chen, M. Rao, K. Feng, and M. J. Zuo, "Physics-informed lstm hyperparameters selection for gearbox fault detection," *Mechanical Systems and Signal Processing*, vol. 171, p. 108907, 2022.
- [35] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
- [36] M. C. Chuah and F. Fu, "Ecg anomaly detection via time series analysis," in *International Symposium on Parallel and Distributed Processing and Applications*. Springer, 2007, pp. 123–135.
- [37] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.
- [38] L. Cohen, G. Avrahami-Bakish, M. Last, A. Kandel, and O. Kipersztok, "Real-time data mining of non-stationary data streams from sensor networks," *Information Fusion*, vol. 9, no. 3, pp. 344–353, 2008.
- [39] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.
- [40] D. Dasgupta and N. S. Majumdar, "Anomaly detection in multidimensional data using negative selection algorithm," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, vol. 2. IEEE, 2002, pp. 1039–1044.
- [41] D. Dasgupta and F. Nino, "A comparison of negative and positive selection algorithms in novel pattern detection," in *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no. 0*, vol. 1. IEEE, 2000, pp. 125–130.

-
- [42] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
- [43] Z. Ding, M. Fei, D. Du, and F. Yang, “Streaming data anomaly detection method based on hyper-grid structure and online ensemble learning,” *Soft Computing*, vol. 21, no. 20, pp. 5905–5917, 2017.
- [44] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, “Learning in nonstationary environments: A survey,” *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.
- [45] P. Domingos and G. Hulten, “Mining high-speed data streams,” in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000, pp. 71–80.
- [46] R. Elwell and R. Polikar, “Incremental learning of concept drift in nonstationary environments,” *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.
- [47] D. M. Farid, L. Zhang, A. Hossain, C. M. Rahman, R. Strachan, G. Sexton, and K. Dahal, “An adaptive ensemble classifier for mining concept drifting data streams,” *Expert Systems with Applications*, vol. 40, no. 15, pp. 5895–5906, 2013.
- [48] H. Ferdowsi, S. Jagannathan, and M. Zawodniok, “An online outlier identification and removal scheme for improving fault detection performance,” *IEEE transactions on neural networks and learning systems*, vol. 25, no. 5, pp. 908–919, 2013.
- [49] A. J. Fox, “Outliers in time series,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 34, no. 3, pp. 350–363, 1972.
- [50] A. W.-C. Fu, O. T.-W. Leung, E. Keogh, and J. Lin, “Finding time series discords based on haar transform,” in *International Conference on Advanced Data Mining and Applications*. Springer, 2006, pp. 31–41.
- [51] R. Fujimaki, “Anomaly detection support vector machine and its application to fault diagnosis,” in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 797–802.
- [52] R. Fujimaki, T. Yairi, and K. Machida, “An anomaly detection method for spacecraft using relevance vector learning,” in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2005, pp. 785–790.

- [53] R. Fujimaki, T. Yairi, and K. Machida, “An approach to spacecraft anomaly detection problem using kernel feature space,” in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005, pp. 401–410.
- [54] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.
- [55] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, “Learning with drift detection,” in *Brazilian symposium on artificial intelligence*. Springer, 2004, pp. 286–295.
- [56] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.
- [57] X.-S. Gan, J.-S. Duanmu, J.-F. Wang, and W. Cong, “Anomaly intrusion detection based on pls feature extraction and core vector machine,” *Knowledge-Based Systems*, vol. 40, pp. 1–6, 2013.
- [58] F. A. Gers, D. Eck, and J. Schmidhuber, “Applying lstm to time series predictable through time-window approaches,” in *Neural Nets WIRN Vietri-01*. Springer, 2002, pp. 193–200.
- [59] Z. Ghafoori, S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. A. Leckie, “Anomaly detection in non-stationary data: Ensemble based self-adaptive ocsvm,” in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 2476–2483.
- [60] P. M. Gonçalves Jr and R. S. M. De Barros, “Rcd: A recurring concept drift framework,” *Pattern Recognition Letters*, vol. 34, no. 9, pp. 1018–1025, 2013.
- [61] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [62] N. Görnitz, M. Braun, and M. Kloft, “Hidden markov anomaly detection,” in *International conference on machine learning*, 2015, pp. 1833–1842.
- [63] F. Gorunescu, *Data Mining: Concepts, models and techniques*. Springer Science & Business Media, 2011, vol. 12.
- [64] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

-
- [65] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional lstm and other neural network architectures,” *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.
- [66] M. Gu, J. Fei, and S. Sun, “Online anomaly detection with sparse gaussian processes,” *Neurocomputing*, vol. 403, pp. 383–399, 2020.
- [67] N. Gugulothu, V. TV, P. Malhotra, L. Vig, P. Agarwal, and G. Shroff, “Predicting remaining useful life using time series embeddings based on recurrent neural networks,” *arXiv preprint arXiv:1709.01073*, 2017.
- [68] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, “Outlier detection for temporal data: A survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250–2267, 2013.
- [69] A. Håkansson, “Portal of research methods and methodologies for research projects and degree projects,” in *The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing WORLDCOMP 2013; Las Vegas, Nevada, USA, 22-25 July*. CSREA Press USA, 2013, pp. 67–73.
- [70] F. Harrou, Y. Sun, and M. Madakyaru, “Kullback-leibler distance-based enhanced detection of incipient anomalies,” *Journal of Loss Prevention in the Process Industries*, vol. 44, pp. 73–87, 2016.
- [71] M. A. Hayes and M. A. Capretz, “Contextual anomaly detection in big sensor data,” in *2014 IEEE International Congress on Big Data*. IEEE, 2014, pp. 64–71.
- [72] M. A. Hayes and M. A. Capretz, “Contextual anomaly detection framework for big sensor data,” *Journal of Big Data*, vol. 2, no. 1, p. 2, 2015.
- [73] D. J. Hill and B. S. Minsker, “Anomaly detection in streaming environmental sensor data: A data-driven modeling approach,” *Environmental Modelling & Software*, vol. 25, no. 9, pp. 1014–1022, 2010.
- [74] D. J. Hill, B. S. Minsker, and E. Amir, “Real-time bayesian anomaly detection in streaming environmental data,” *Water Resources Research*, vol. 45, no. 4, 2009.
- [75] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [76] A. Holst, M. Bohlin, J. Ekman, O. Sellin, B. Lindström, and S. Larsen, “Statistical anomaly detection for train fleets,” *AI Magazine*, vol. 34, no. 1, pp. 33–33, 2013.

-
- [77] H. Huang, H. Qin, S. Yoo, and D. Yu, “Physics-based anomaly detection defined on manifold space,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 9, no. 2, p. 14, 2014.
- [78] S.-Y. Huang, J.-W. Lin, and R.-H. Tsaih, “Outlier detection in the concept drifting environment,” in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 31–37.
- [79] G. Hulten, L. Spencer, and P. Domingos, “Mining time-changing data streams,” in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 97–106.
- [80] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, “Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding,” in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 387–395.
- [81] R. J. Hyndman, E. Wang, and N. Laptev, “Large-scale unusual time series detection,” in *2015 IEEE international conference on data mining workshop (ICDMW)*. IEEE, 2015, pp. 1616–1619.
- [82] M. V. Joshi, R. C. Agarwal, and V. Kumar, “Mining needle in a haystack: classifying rare classes via two-phase rule induction,” in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001, pp. 91–102.
- [83] M. V. Joshi, R. C. Agarwal, and V. Kumar, “Predicting rare classes: Can boosting make any weak learner strong?” in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 297–306.
- [84] B. Juba, C. Musco, F. Long, S. Sidiropoulos-Douskos, and M. C. Rinard, “Principled sampling for anomaly detection.” in *NDSS*, 2015.
- [85] S. Kanarachos, S.-R. G. Christopoulos, A. Chroneos, and M. E. Fitzpatrick, “Detecting anomalies in time series data via a deep learning algorithm combining wavelets, neural networks and hilbert transform,” *Expert Systems with Applications*, vol. 85, pp. 292–304, 2017.
- [86] E. Keogh, J. Lin, and A. Fu, “Hot sax: Efficiently finding the most unusual time series subsequence,” in *Fifth IEEE International Conference on Data Mining (ICDM’05)*. Ieee, 2005, pp. 8–pp.

- [87] E. Keogh, J. Lin, S.-H. Lee, and H. Van Herle, "Finding the most unusual time series subsequence: algorithms and applications," *Knowledge and Information Systems*, vol. 11, no. 1, pp. 1–27, 2007.
- [88] F. Knorn and D. J. Leith, "Adaptive kalman filtering for anomaly detection in software appliances," in *IEEE INFOCOM Workshops 2008*. IEEE, 2008, pp. 1–6.
- [89] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *The Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.
- [90] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms—the numenta anomaly benchmark," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2015, pp. 38–44.
- [91] D. Lee, "Anomaly detection in multivariate non-stationary time series for automatic dbms diagnosis," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2017, pp. 412–419.
- [92] A. Legrand, H. Trannois, and A. Courrier, "Use of uncertainty with autoencoder neural networks for anomaly detection," in *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. IEEE, 2019, pp. 32–35.
- [93] J. Li, W. Pedrycz, and I. Jamal, "Multivariate time series anomaly detection: A framework of hidden markov models," *Applied Soft Computing*, vol. 60, pp. 229–240, 2017.
- [94] K. Limthong, K. Fukuda, Y. Ji, and S. Yamada, "Unsupervised learning model for real-time anomaly detection in computer networks," *IEICE TRANSACTIONS on Information and Systems*, vol. 97, no. 8, pp. 2084–2094, 2014.
- [95] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.
- [96] S. Lin, R. Clark, R. Birke, S. Schönborn, N. Trigoni, and S. Roberts, "Anomaly detection for time series using vae-lstm hybrid model," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 4322–4326.
- [97] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell, "Learning to diagnose with lstm recurrent neural networks," *arXiv preprint arXiv:1511.03677*, 2015.

-
- [98] T. Lotze, G. Shmueli, S. Murphy, H. Burkom *et al.*, “A wavelet-based anomaly detector for early detection of disease outbreaks,” in *Workshop on Machine Learning Algorithms for Surveillance and Event Detection, 23rd Intl Conference on Machine Learning*. Citeseer, 2006.
- [99] W. Lu, Y. Cheng, C. Xiao, S. Chang, S. Huang, B. Liang, and T. Huang, “Un-supervised sequential outlier detection with deep architectures,” *IEEE transactions on image processing*, vol. 26, no. 9, pp. 4321–4330, 2017.
- [100] J. Ma and S. Perkins, “Online novelty detection on temporal sequences,” in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 613–618.
- [101] J. Ma and S. Perkins, “Time-series novelty detection using one-class support vector machines,” in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 3. IEEE, 2003, pp. 1741–1745.
- [102] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, “Lstm-based encoder-decoder for multi-sensor anomaly detection,” *arXiv preprint arXiv:1607.00148*, 2016.
- [103] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, “Long short term memory networks for anomaly detection in time series,” in *Proceedings*. Presses universitaires de Louvain, 2015, p. 89.
- [104] C. Marco, “Anomaly detection with lstm in keras,” <https://towardsdatascience.com/anomaly-detection-with-lstm-in-keras-8d8d7e50ab1b>, 2019.
- [105] L. Martí, N. Sanchez-Pi, J. Molina, and A. Garcia, “Anomaly detection based on sensor data in petroleum industry applications,” *Sensors*, vol. 15, no. 2, pp. 2774–2797, 2015.
- [106] J. McBain and M. Timusk, “Feature extraction for novelty detection as applied to fault detection in machinery,” *Pattern Recognition Letters*, vol. 32, no. 7, pp. 1054–1061, 2011.
- [107] H. Z. Moayedi and M. Masnadi-Shirazi, “Arima model for network traffic prediction and anomaly detection,” in *2008 International Symposium on Information Technology*, vol. 4. IEEE, 2008, pp. 1–6.
- [108] G. B. Moody and R. G. Mark, “The impact of the mit-bih arrhythmia database,” *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

-
- [109] M. Moshtaghi, C. Leckie, S. Karunasekera, J. C. Bezdek, S. Rajasegarar, and M. Palaniswami, “Incremental elliptical boundary estimation for anomaly detection in wireless sensor networks,” in *2011 IEEE 11th international conference on data mining*. IEEE, 2011, pp. 467–476.
- [110] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, “Deepant: A deep learning approach for unsupervised anomaly detection in time series,” *IEEE Access*, vol. 7, pp. 1991–2005, 2018.
- [111] O. Olabiyi, E. Martinson, V. Chintalapudi, and R. Guo, “Driver action prediction using deep (bidirectional) recurrent neural network,” *arXiv preprint arXiv:1706.02257*, 2017.
- [112] A. Olivier, M. D. Shields, and L. Graham-Brady, “Bayesian neural networks for uncertainty quantification in data-driven materials modeling,” *Computer Methods in Applied Mechanics and Engineering*, vol. 386, p. 114079, 2021.
- [113] C. O’Reilly, A. Gluhak, M. A. Imran, and S. Rajasegarar, “Anomaly detection in wireless sensor networks in a non-stationary environment,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1413–1432, 2014.
- [114] J. Pang, D. Liu, Y. Peng, and X. Peng, “Anomaly detection based on uncertainty fusion for univariate monitoring series,” *Measurement*, vol. 95, pp. 280–292, 2017.
- [115] C. Phua, D. Alahakoon, and V. Lee, “Minority report in fraud detection: classification of skewed data,” *Acm sigkdd explorations newsletter*, vol. 6, no. 1, pp. 50–59, 2004.
- [116] B. Pincombe, “Anomaly detection in time series of graphs using arma processes,” *Asor Bulletin*, vol. 24, no. 4, p. 2, 2005.
- [117] D. Pokrajac, A. Lazarevic, and L. J. Latecki, “Incremental local outlier detection for data streams,” in *2007 IEEE symposium on computational intelligence and data mining*. IEEE, 2007, pp. 504–515.
- [118] J. A. Quinn and M. Sugiyama, “A least-squares approach to anomaly detection in static and sequential data,” *Pattern Recognition Letters*, vol. 40, pp. 36–40, 2014.
- [119] S. Ramamurthy and R. Bhatnagar, “Tracking recurrent concept drift in streaming data using ensemble classifiers,” in *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*. IEEE, 2007, pp. 404–409.

REFERENCES

- [120] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, “A survey on data preprocessing for data stream mining: Current status and future directions,” *Neurocomputing*, vol. 239, pp. 39–57, 2017.
- [121] C. E. Rasmussen, “Gaussian processes in machine learning,” in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.
- [122] M. A. Rassam, M. A. Maarof, and A. Zainal, “Adaptive and online data anomaly detection for wireless sensor systems,” *Knowledge-Based Systems*, vol. 60, pp. 44–57, 2014.
- [123] A. M. Rather, A. Agarwal, and V. Sastry, “Recurrent neural network and a hybrid model for prediction of stock returns,” *Expert Systems with Applications*, vol. 42, no. 6, pp. 3234–3241, 2015.
- [124] N. Reimers and I. Gurevych, “Optimal hyperparameters for deep lstm-networks for sequence labeling tasks,” *arXiv preprint arXiv:1707.06799*, 2017.
- [125] J. C. Reinhold, Y. He, S. Han, Y. Chen, D. Gao, J. Lee, J. L. Prince, and A. Carass, “Validating uncertainty in medical image translation,” in *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. IEEE, 2020, pp. 95–98.
- [126] N. Reunanen, T. Rätty, J. J. Jokinen, T. Hoyt, and D. Culler, “Unsupervised online detection and prediction of outliers in streams of sensor data,” *International Journal of Data Science and Analytics*, pp. 1–30, 2019.
- [127] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*. John Wiley & sons, 2005, vol. 589.
- [128] O. Salem, Y. Liu, A. Mehaoua, and R. Boutaba, “Online anomaly detection in wireless body area networks for reliable healthcare monitoring,” *IEEE journal of biomedical and health informatics*, vol. 18, no. 5, pp. 1541–1551, 2014.
- [129] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, “Deepar: Probabilistic forecasting with autoregressive recurrent networks,” *International Journal of Forecasting*, 2019.
- [130] M. Sangha, D. Yu, and J. Gomm, “Sensor fault diagnosis for automotive engines with real data evaluation,” *International Journal of Engineering, Science and Technology*, vol. 3, no. 8, pp. 13–25, 2011.
- [131] S. Saurav, P. Malhotra, V. TV, N. Gugulothu, L. Vig, P. Agarwal, and G. Shroff, “Online anomaly detection with concept drift adaptation using recurrent neural

- networks,” in *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*. ACM, 2018, pp. 78–87.
- [132] M. Schneider, W. Ertel, and F. Ramos, “Expected similarity estimation for large-scale batch and streaming anomaly detection,” *Machine Learning*, vol. 105, no. 3, pp. 305–333, 2016.
- [133] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, “Support vector method for novelty detection,” in *Advances in neural information processing systems*, 2000, pp. 582–588.
- [134] M. Schreyer, T. Sattarov, D. Borth, A. Dengel, and B. Reimer, “Detection of anomalies in large scale accounting data using deep autoencoder networks,” *arXiv preprint arXiv:1709.05254*, 2017.
- [135] S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman, “Stock price prediction using lstm, rnn and cnn-sliding window model,” in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2017, pp. 1643–1647.
- [136] M. Sheikhan and Z. Jadidi, “Flow-based anomaly detection in high-speed links using modified gsa-optimized neural network,” *Neural Computing and Applications*, vol. 24, no. 3, pp. 599–611, 2014.
- [137] D. T. Shipmon, J. M. Gurevitch, P. M. Piselli, and S. T. Edwards, “Time series anomaly detection; detection of anomalous drops with limited features and sparse examples in noisy highly periodic data,” *arXiv preprint arXiv:1708.03665*, 2017.
- [138] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, “Anomaly detection in streams with extreme value theory,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1067–1075.
- [139] A. Singh, “Anomaly detection for temporal data using long short-term memory (lstm),” 2017.
- [140] N. Singh and C. Olinsky, “Demystifying numenta anomaly benchmark,” in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 1570–1577.
- [141] S. K. Singh, J. S. Fowdur, J. Gawlikowski, and D. Medina, “Leveraging evidential deep learning uncertainties with graph-based clustering to detect anomalies,” *arXiv preprint arXiv:2107.01557*, 2021.

REFERENCES

- [142] V. A. Sotiris, W. T. Peter, and M. G. Pecht, "Anomaly detection through a bayesian support vector machine," *IEEE Transactions on Reliability*, vol. 59, no. 2, pp. 277–286, 2010.
- [143] I. Steinwart, D. Hush, and C. Scovel, "A classification framework for anomaly detection." *Journal of Machine Learning Research*, vol. 6, no. 2, 2005.
- [144] N. Subrahmanya and Y. C. Shin, "A data-based framework for fault detection and diagnostics of non-linear systems with partial state measurement," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 446–455, 2013.
- [145] A. I. Tambuwal and A. M. Bello, "Deepconad: Deep and confidence prediction for unsupervised anomaly detection in time series," in *Science and Information Conference*. Springer, 2020, pp. 232–244.
- [146] A. I. Tambuwal and D. Neagu, "Deep quantile regression for unsupervised anomaly detection in time-series," *SN Computer Science*, vol. 2, no. 6, pp. 1–16, 2021.
- [147] A. I. Tambuwal, D. Neagu, and M. Gheorghe, "An experimental comparison of ensemble classifiers for evolving data streams," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 2017, pp. 156–162.
- [148] S. C. Tan, K. M. Ting, and T. F. Liu, "Fast anomaly detection for streaming data," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.
- [149] N. Tatbul, T. J. Lee, S. Zdonik, M. Alam, and J. Gottschlich, "Precision and recall for time series," *arXiv preprint arXiv:1803.03639*, 2018.
- [150] M. Teng, "Anomaly detection on time series," in *2010 IEEE International Conference on Progress in Informatics and Computing*, vol. 1. IEEE, 2010, pp. 603–608.
- [151] J. P. Theiler and D. M. Cai, "Resampling approach for anomaly detection in multispectral images," in *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery IX*, vol. 5093. International Society for Optics and Photonics, 2003, pp. 230–240.
- [152] A. Theissler and I. Dear, "An anomaly detection approach to detect unexpected faults in recordings from test drives," in *Proceedings of the WASET International Conference on Vehicular Electronics and Safety*, vol. 7, no. 7, 2013, pp. 195–198.

- [153] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.
- [154] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional lstm with cnn features," *IEEE Access*, vol. 6, pp. 1155–1166, 2017.
- [155] L. van de Wiel, D. M. van Es, and A. Feelders, "Real-time outlier detection in time series data of water sensors," in *International Workshop on Advanced Analytics and Learning on Temporal Data*. Springer, 2020, pp. 155–170.
- [156] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.
- [157] M. R. Vargas, B. S. De Lima, and A. G. Evsukoff, "Deep learning for stock market prediction from financial news articles," in *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*. IEEE, 2017, pp. 60–65.
- [158] R. Vilalta and S. Ma, "Predicting rare events in temporal domains," in *2002 IEEE International Conference on Data Mining, 2002. Proceedings*. IEEE, 2002, pp. 474–481.
- [159] N. Wagner and Z. Michalewicz, "An analysis of adaptive windowing for time series forecasting in dynamic environments: further tests of the dyfor gp model," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, 2008, pp. 1657–1664.
- [160] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 226–235.
- [161] W. Wang, B. Zhang, D. Wang, Y. Jiang, S. Qin, and L. Xue, "Anomaly detection based on probability density function with kullback–leibler divergence," *Signal Processing*, vol. 126, pp. 12–17, 2016.
- [162] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, 2016.
- [163] G. I. Webb, L. K. Lee, B. Goethals, and F. Petitjean, "Analyzing concept drift and shift from sample data," *Data Mining and Knowledge Discovery*, vol. 32, no. 5, pp. 1179–1199, 2018.

-
- [164] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, “A multi-horizon quantile recurrent forecaster,” *arXiv preprint arXiv:1711.11053*, 2017.
- [165] R. Wu and E. Keogh, “Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress,” *IEEE Transactions on Knowledge and Data Engineering*, 2021.
- [166] K. Yang and C. Shahabi, “A pca-based similarity measure for multivariate time series,” in *Proceedings of the 2nd ACM international workshop on Multimedia databases*, 2004, pp. 65–74.
- [167] Y. Ye, S. Squartini, and F. Piazza, “Online sequential extreme learning machine in nonstationary environments,” *Neurocomputing*, vol. 116, pp. 94–101, 2013.
- [168] K. Yu and R. A. Moyeed, “Bayesian quantile regression,” *Statistics & Probability Letters*, vol. 54, no. 4, pp. 437–447, 2001.
- [169] J. Zhang and H. Wang, “Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance,” *Knowledge and information systems*, vol. 10, no. 3, pp. 333–355, 2006.
- [170] J. Zhang, F.-C. Tsui, M. M. Wagner, and W. R. Hogan, “Detection of outbreaks from time series data using wavelet transform,” in *AMIA Annual Symposium Proceedings*, vol. 2003. American Medical Informatics Association, 2003, p. 748.
- [171] Z. Zhao, K. G. Mehrotra, and C. K. Mohan, “Ensemble algorithms for unsupervised anomaly detection,” in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2015, pp. 514–525.
- [172] L. Zhu and N. Laptev, “Deep and confident prediction for time series at uber,” in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2017, pp. 103–110.
- [173] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, “Deep autoencoding gaussian mixture model for unsupervised anomaly detection,” in *International Conference on Learning Representations*, 2018.

Appendix **A**

Results of the Experiment conducted
using UCR Time Series Anomaly
Benchmark Datasets.

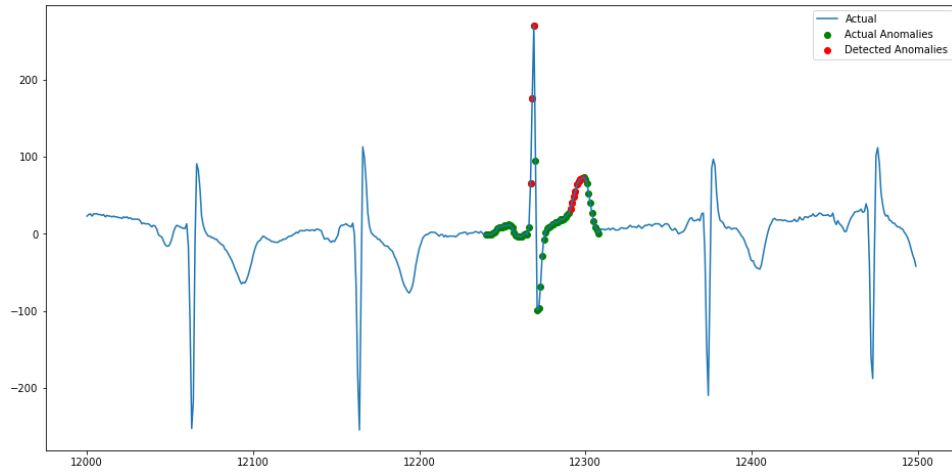


Figure A.1: Detected vs Actual Anomalies on apneaecg1 time series

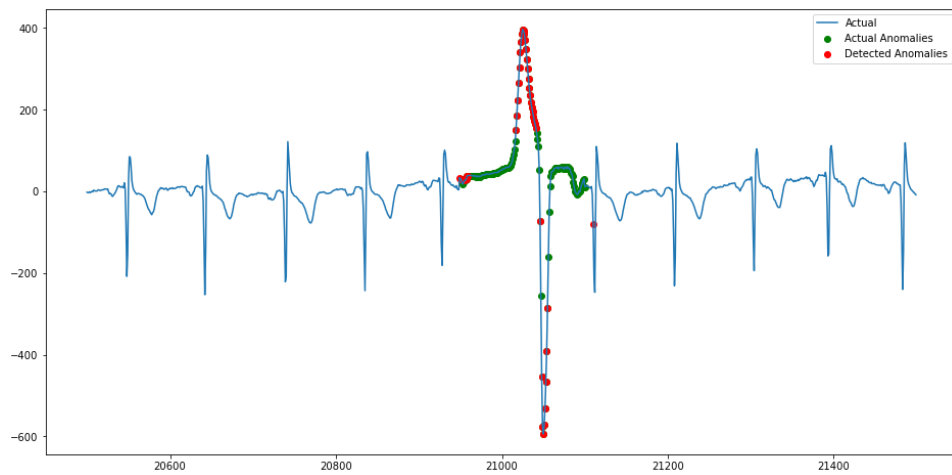


Figure A.2: Detected vs Actual Anomalies on apneaecg2 time series

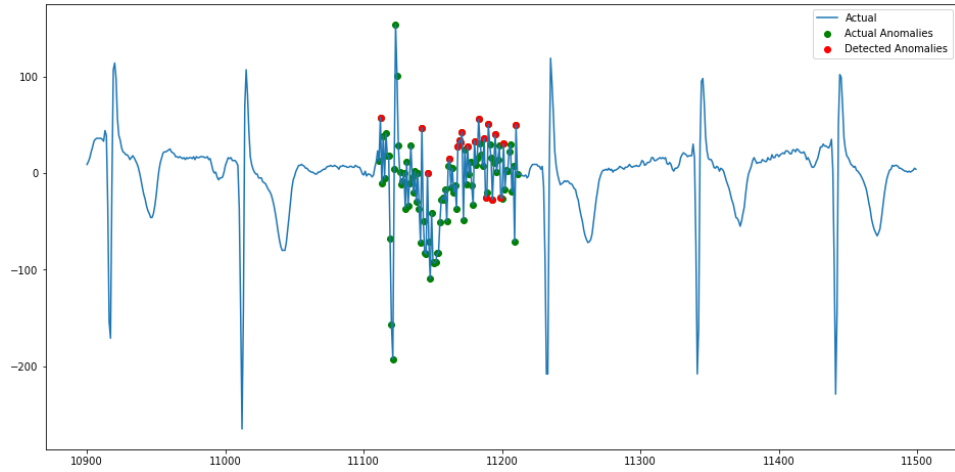


Figure A.3: Detected vs Actual Anomalies on apneaecg3 time series

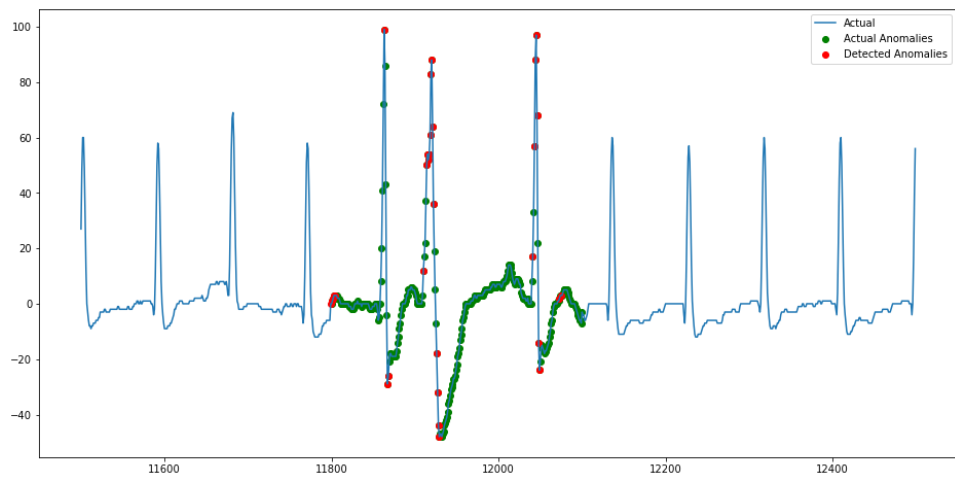


Figure A.4: Detected vs Actual Anomalies on ECG1 time series

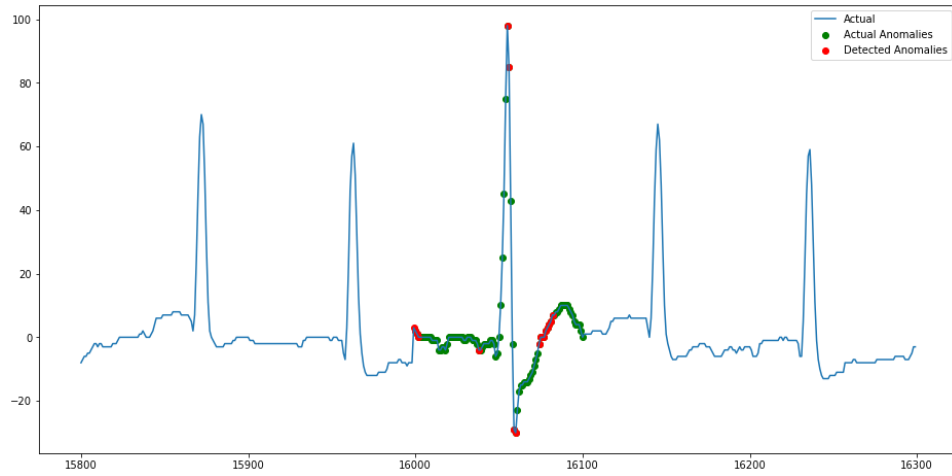


Figure A.5: Detected vs Actual Anomalies on ECG2 time series

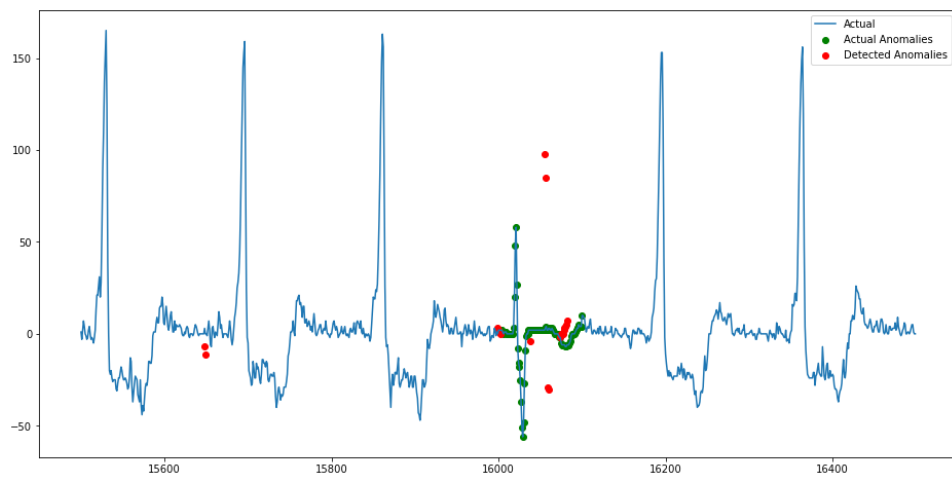


Figure A.6: Detected vs Actual Anomalies on ECG3 time series

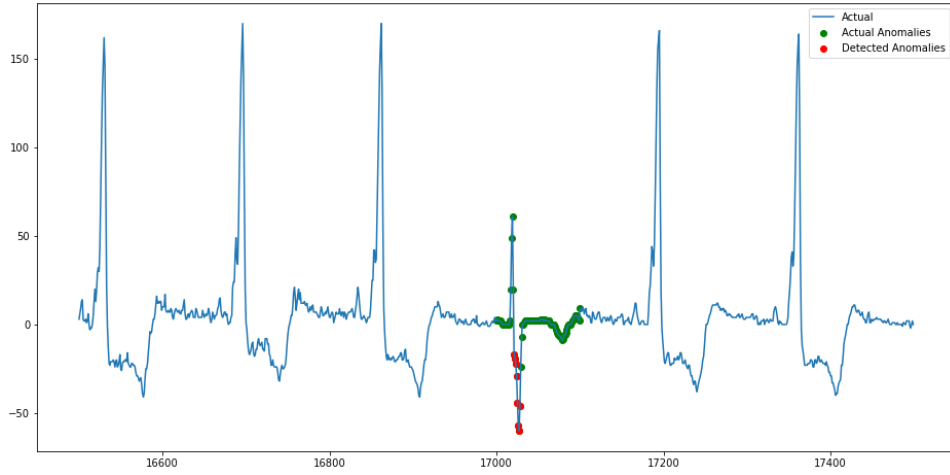


Figure A.7: Detected vs Actual Anomalies on ECG4 time series

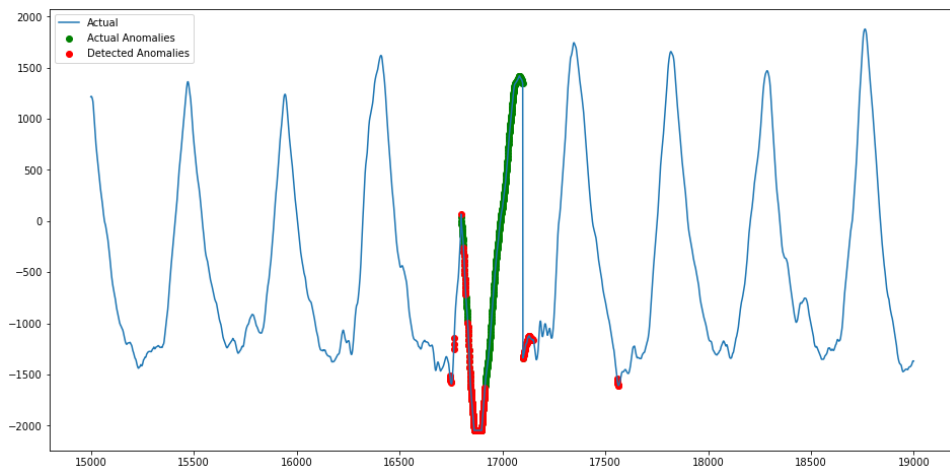


Figure A.8: Detected vs Actual Anomalies on ECG5 time series

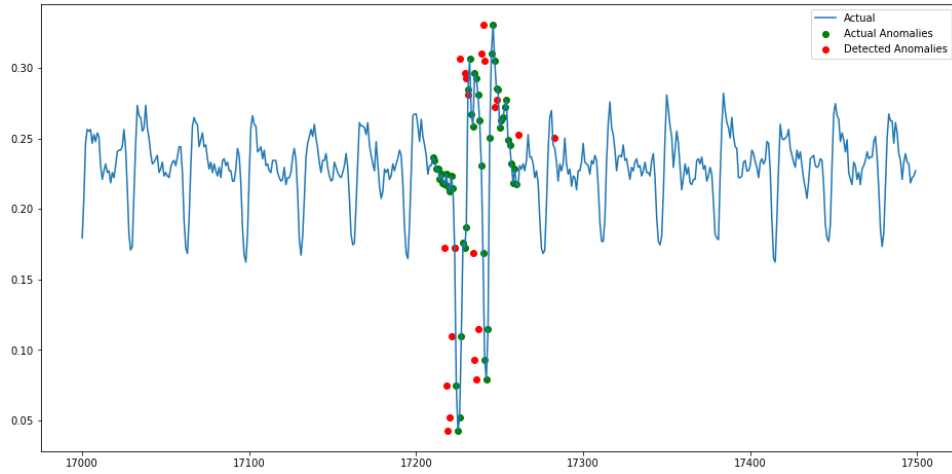


Figure A.9: Detected vs Actual Anomalies on EPG1 time series

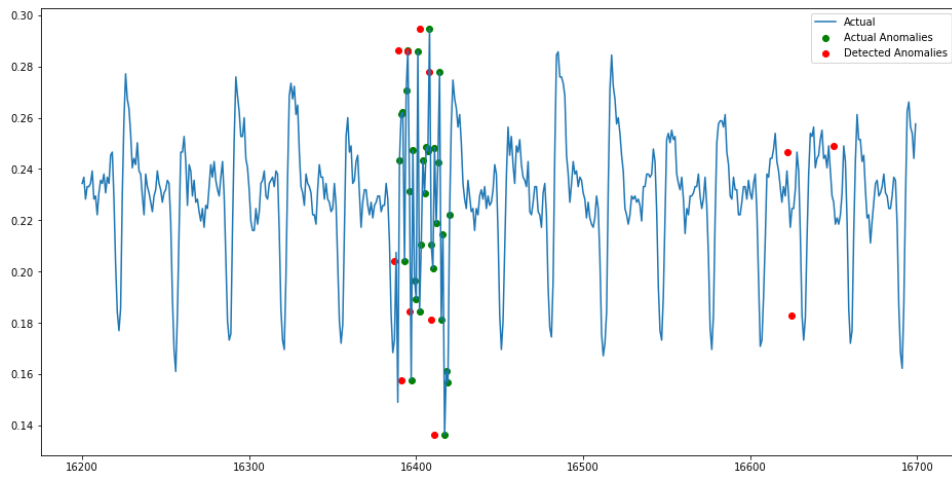


Figure A.10: Detected vs Actual Anomalies on EPG3 time series

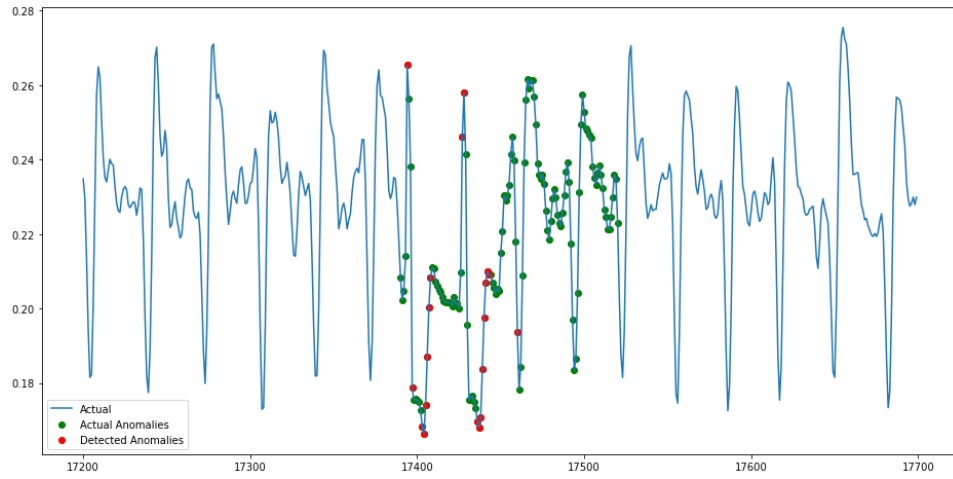


Figure A.11: Detected vs Actual Anomalies on EPG5 time series

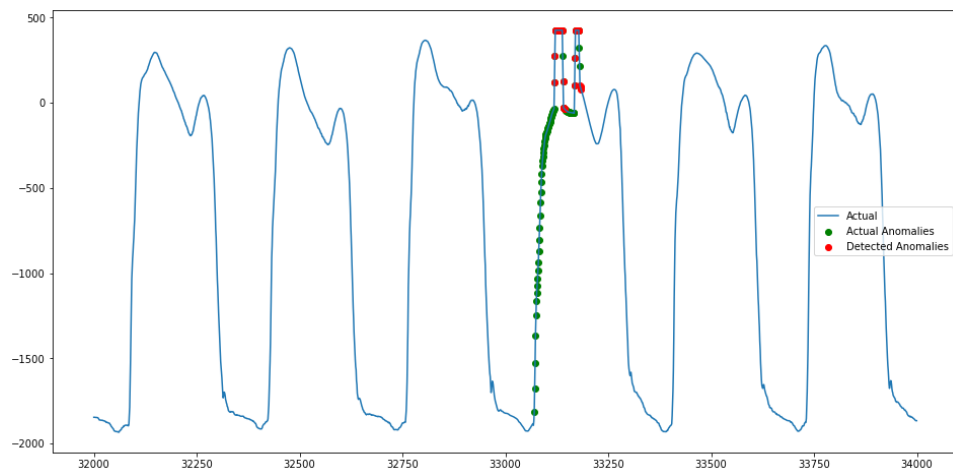


Figure A.12: Detected vs Actual Anomalies on Gaithunt time series

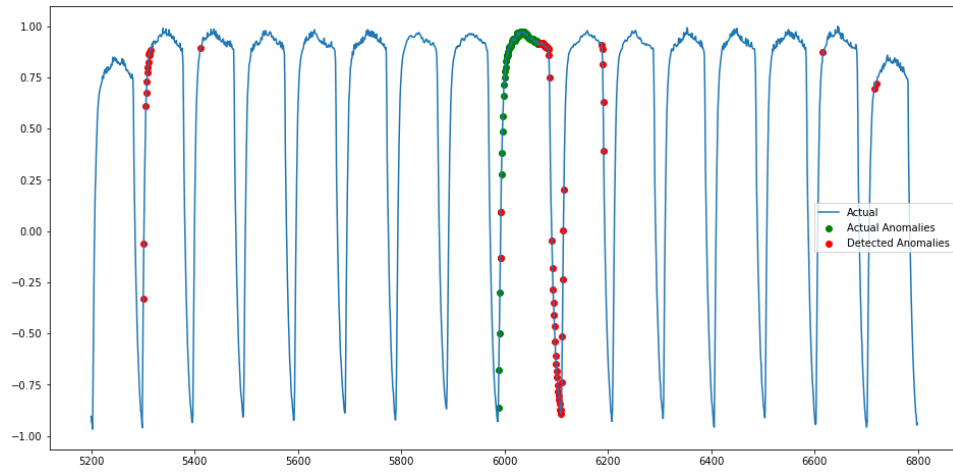


Figure A.13: Detected vs Actual Anomalies on MARS1 time series

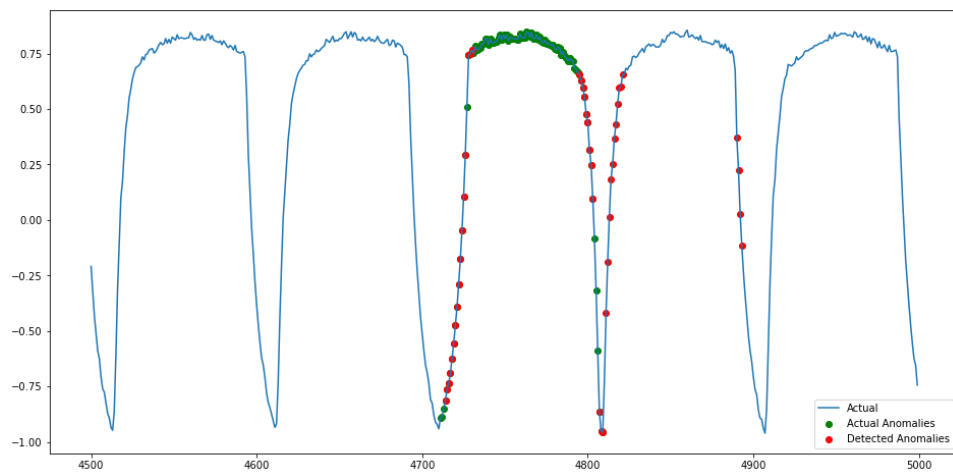


Figure A.14: Detected vs Actual Anomalies on MARS5 time series

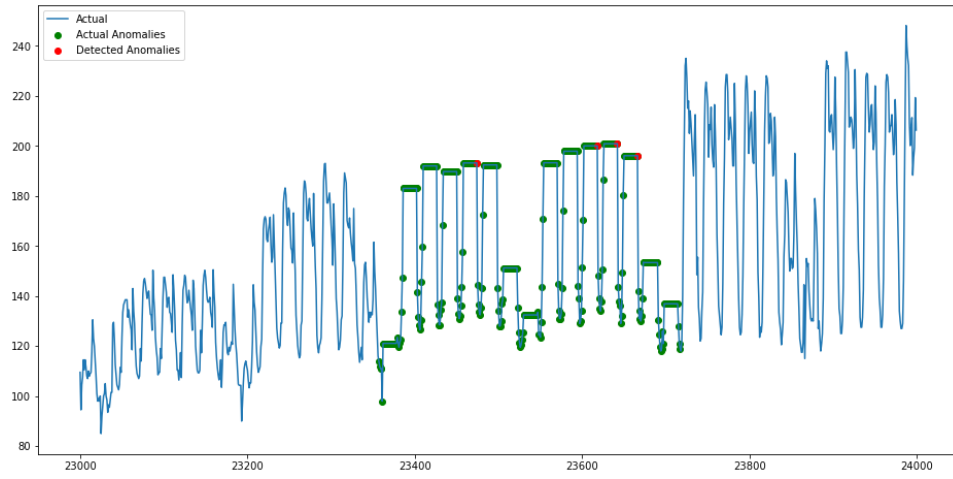


Figure A.15: Detected vs Actual Anomalies on Power Demand2 time series

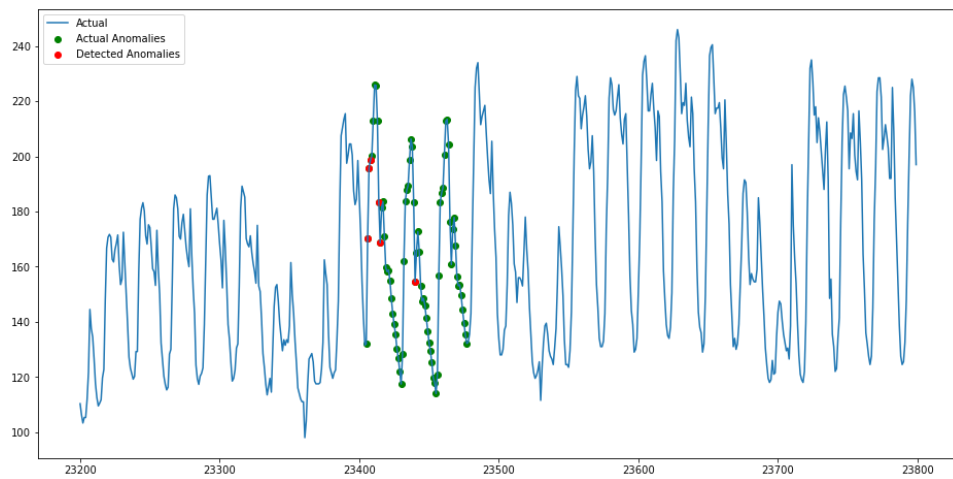


Figure A.16: Detected vs Actual Anomalies on Power Demand3 time series

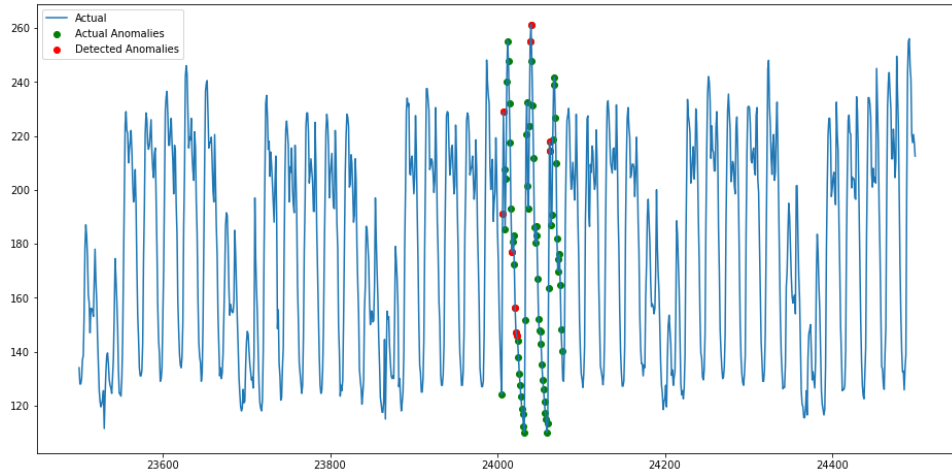


Figure A.17: Detected vs Actual Anomalies on Power Demand4 time series

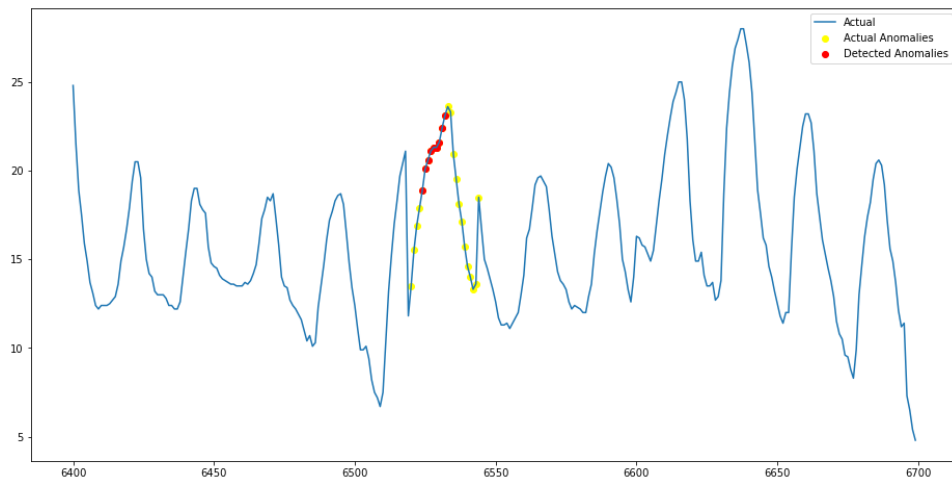


Figure A.18: Detected vs Actual Anomalies on Temperature time series