A Fully Online Approach for Anomaly Detection and Change-Point Detection in Streaming Data using LSTMs

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCIENCE AND ENGINEERING

2022

Memoona Khanam

Department of Computer Science

Contents

Word Counts: 46,223

List of Figures

List of Tables

List of Algorithms

Abbreviations

AD	Anomaly Detection
Adam	Adaptive Moment Estimation
AUC	Area Under Curve
BPTT	Backpropagation Through Time
CPD	Change-Point Detection
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
LSTM	Long Short-Term Memory
MVN	Multivariate Normal Distribution
NAB	Numenta Anomaly Benchmark
OLACD	Online Anomaly and Change Detection
RNN	Recurrent Neural Network
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
UVTS	Univariate Time Series

Abstract

In this thesis, we propose a novel online anomaly detection and change detection algorithm based on contemporary Recurrent Neural Networks such as Long Short-Term Memory (LSTM) for time series anomaly detection in a changing environment posed by either transitory change (point anomaly) or permanent change (change-point) in normal behaviour in streaming data. The proposed online anomaly and change-point detection model is trained incrementally as new data stream becomes available and is capable of adapting to the changes in the data distribution of underlying pattern. LSTM is used to make single or multi-step predictions of the time series which could be from 1 step up to 10 steps ahead, and the prediction errors are used to detect anomalies and change-points as well as update the model. Fundamentally, the proposed algorithm is developing the online model of prediction error such that the large prediction error is employed to indicate the anomalous behaviour in changing environment. Additionally, the prediction errors are used to update the proposed online model in such a way that transitory anomalies do not lead to a radical change in the model. Whereas high computed prediction errors over a period of time due to permanent changes/change-points lead to substantial updates in model. The model automatically and swiftly adapts to the changing statistics and new custom of input data distribution. The proposed online anomaly detection and change-point detection technique is striving for fully on-line performance; therefore, model will not assume any labels in data stream except during evaluation. Furthermore, our novel proposed online anomaly and change detection model is not relying on any user define parameters (such as *threshold*) but automatically defined and updated by model itself.

We validate the efficiency of the proposed novel model-based parametric unsupervised online approach through experiments on publicly accessible and proprietary three real-world and four synthetic benchmark datasets, which are taken and derived from Yahoo Labs Benchmark Dataset (Yahoo Webscope) and Numenta Anomaly Benchmark (NAB) that contains labelled anomalies. We compare the results in term of Area Under Curve (AUC) with state of-the-art algorithm available in literature.

We perceive that proposed online anomaly and change-point detection model perform reliably better for multistep predictions as compared to models with single step predictions for all data sets in terms of AUC. We also observe that model trained for predictions of multi-step such as 5 or 10 steps ahead are more competent to detect the temporal changes in the target output distribution and are consequently able to detect changes point to and adapt to the changes much early as compared to model trained to predict only with one-step. Where step is the number of future predictions of the model. Unlike other methods our method has the advantage that it not only detects the anomalies as quickly as possible but did not let the anomalies to make drastic change in distribution. Whereas in case of short-term changes/point anomalies the proposed online model suggests that there is a trade-off between how early the anomaly detection happened and how long do the false positives last is depending on prediction length.

It is concluded that our proposed online anomaly and change-point detection model consistently outperform and give the full advantage when it utilizes for multistep time series predictions.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgments

It is a great honour to have reached the point where I am writing the acknowledgements. I would like to thank my academic supervisors **Dr. Jonathan Shapiro** for all of his help and supply of ideas that keep me focussed throughout this project and for supporting me out to gain immensely valuable experience.

I am profoundly grateful to the **Schlumberger Foundation**, Faculty for the Future for funding this research. This funding has enabled me to undertake the incredibly enjoyable, even though difficult, time of research which has not only developed my interests but also introduced me to more. It has also massively increased my self-confidence and self-esteem. Taking on this endeavour on a full-time basis has been a challenging and at times, isolating experience. It is to the wonderful credit of the people around me that made my spirit remained unbroken. I am deliriously happy to say that a dream which my mother (late) **Asifa Khanam** and father **Malik Nazar Hayat Khan** seen years ago is fulfilled by my taya Abu (father-in-law) **Malik Sikander Hayat Khiyal** and khala (mother-in-law) **Nasira Begum**. My parent in-laws always support and appreciate me at every step throughout the past years which is intensely thankful. I am extremely gratifying to my parents for their support and prayers which always keep me unbreakable, resilient and for them being honoured of what I do. There are too many acknowledgements to list individually. My thanks go to my brothers, sisters, friends and all my bonding relations for their support and prayers.

I would like to special and heartiest thank my husband **Malik Awais Sikander** for his endless support and relinquishing the Banking career at the time of his progression to fulfil my dream. Finally, many thanks go to my three lovely boys, **Malik Aaghar Hayat**, **Malik Zaafar Hayat** and **Malik Faatir Hayat**, for allowing me to work by forgoing outings and fun; for them being proud of what I do. They have been so supportive, and strong in themselves, even though the many difficulties currently affecting the world. I hope that I can encourage them to never stop learning and to make

Chapter 1

Introduction

A critical problem in streaming temporal data analysis is an *anomaly* detection [7] which identifies the points when the underlying distribution of a time series abruptly changes and deviate from normal behaviour, either transitory (outlier/point anomaly [8]) or permanent (change-point [9]).

Anomaly detection while forecasting of streaming data plays an important role in many real applications, ranging from IoT systems [10], cyber-networks [11], healthcare [12] and to industrial systems [13]. However, the real streaming data is habitually complicated with anomalies and change points, which can lead the training models deviating from the underlying patterns of the time series, particularly when the model trains in the context of online learning mode. Though, several limitations confine to the use of some existing methods in real-world applications. First, several anomaly detection techniques work in offline manner, where the whole time series needs to be stored before anomaly detection can be performed [8, 14]. These approaches of training one model in an offline way using historical data is likely to fail under non-stationary and dynamically changing environments where the definition of normal behaviour of streaming data changes over time making the model irrelevant and ineffective to *streaming* especially for very *imbalance class* and *rare-class* time series [16]. Second, the development made in for anomaly detection has been mostly based on supervised machine learning algorithms that require labelled datasets to be trained either in online or offline way. However, collecting and annotating labelled streaming large-scale datasets is hard, time-consuming and even too expensive, while it requires domain knowledge from experts. Therefore, anomaly detection has been great challenge for researchers and constrained by the availability of labelled datasets [16]. Finally, most anomaly detection techniques require user-defined parameters (e.g., threshold) that need to be chosen based on the observed data, which limits their applicability to new unseen data [16].

Generally, anomaly detection methods involve two main phases, *learning phase* and *anomaly detection phase*.

Initial phase that is called *learning phase,* where the estimating of an unknown desired signal is done which is known as regression problem and is one of the main subjects of interest in contemporary online learning literature, where we sequentially receive a data sequence related to a desired signal to predict the signal's next value [17]. The online regression is extensively studied in the neural network and machine learning [18], especially for prediction tasks [19] using time series data. The *learning phase* can be as supervised or unsupervised and that enable the anomaly detection model to decide the essential parameters using training dataset before actual anomaly identifications are made in *anomaly detection phase.* In unsupervised techniques the identification of anomalies is not based on label rather based on their deviation from the normal data points [19]. The key focus of this thesis lies on the grounds of unsupervised methods.

Though, during *learning phase* the learning process can be perceived as *offline* if it learns on a static dataset or *online/incremental* if that learns on a streaming dataset and training with RNNs in an incremental manner for streaming data is a lesser explored area [20]. Learning on the static dataset is a conventional [21] way of learning for anomaly detection algorithms. In *offline learning* all essential information can be obtained from the static training dataset prior to making actual detections, which means the dataset with sufficient information must be available all at once. A recent work in [22] explores incremental training of RNNs by using adaptive Real Time Recurrent Learning (RTRL) learning. Another study derived the online RNN learning updates based on Stochastic gradient descent (SGD), Extended Kalman Filter (EKF) and Particle Filter (PF) algorithms [17]. An online gradient learning algorithm, RoAdam for LSTM is proposed in [23] for time series prediction which is modified on the basis of Adam and robust to outliers. The focus of these studies is primarily on forecasting of time series and but do not directly detecting anomalies.

The presence of outliers and change points in a time series can adversely affect in its analysis and complicate the learning process [21]. Traditional approaches for anomaly detection in time series learn a model of normal behaviour in an offline manner and use this model to detect anomalies in new unseen time series data [8, 24]. As the normal behaviour changes to an anomalous, the anomaly detection models likely to fail to adapt to the changes of normal behaviour because the model trained in an offline fashion do not address the challenges of changing behaviour, thus making the offline model irrelevant and ineffective for real streaming data [5, 20]. The *incremental* or *online learning* methods are thus required, which learn and update models incrementally under dynamically changing and non-stationary environments as new data arrives [7]. The anomaly detection techniques must be able to start recognizing anomalies as soon as possible with a small amount of initial knowledge and adapt this information as new data stream is available.

Anomaly detection has been undertaken using different approaches over time and finds its earliest history in the statistics community dated back in the 19th century [15]. Various of the existing machine learning models for streaming data analysis focuses entirely on either outlier/point anomaly detection or change point detection [15, 20]. Our research aims to provide modern RNN [1, 25] based approach to handle both outlier/point anomalies and change points simultaneously in an online manner for streaming data.

To address these issues, we propose an unsupervised model-based parametric *online Anomaly Detection and Change Detection Model for Streaming Data (OLACD)* based on LSTM that uses an on-line estimate of the error distribution for anomaly detection that does not require prior distributional knowledge for detecting point anomalies/outliers and change points and effectively handles un-labelled univariate time series.

The proposed OLACD model is trained incrementally as new data stream becomes available and is capable of adapting to the changes in the data distribution. LSTM is used to make single or multi-step predictions of the time series, and the prediction errors are used to detect anomalies and change points as well as update the OLACD model. Large prediction error is used to indicate anomalous behaviour. Further, the prediction errors are used to update the proposed model in such a way that anomalies and change points do not lead to a drastic change in the model but rapidly adapts to the new norm of input data distribution. This approach was publicised to be effective in off-line mode and supervised setting [8, 26] where the anomaly detection parameters (e.g., threshold) are pre-defined and selected on the bases of labelled time series that explicitly showing normal behaviour. The proposed anomaly detection technique is striving for fully on-line performance; therefore, we will not assume any labels in data stream except during evaluation of proposed method. Furthermore, our novel proposed online anomaly detection model is not relying on any user define parameters but automatically defined and updated by model itself.

We validate the efficiency of the proposed novel model-based parametric unsupervised approach via experiments on publicly available and proprietary three real-world and four synthetic benchmark datasets [6, 27] and compare the results in term of Area Under Curve (AUC) with state of-the-art algorithm available in literature proposed by Saurav *et. al.* in 2018 [5] and are demonstrated in chapter 4 and 5.

1.1 Motivation

In the current era of digitally computation of Bid Data, time series are being generated in enormous amounts. Nowadays, sensors and Internet of Things (IoT) devices are ubiquitous [10] and produces unbounded streaming data continuously at a rapid pace. Streaming data analysis applications enforce unique constraints and challenges for researcher to developed machine learning models for anomaly detection. These applications involve analysing a continuous sequence of data occurring in real-time, in contrast to offline or batch processing, where the full dataset is available. The model observes each data record in successive order as they arrive, and any processing or learning must be done in an online fashion [7].

We define anomalies either short-term or long-term in a streaming data to be patterns that do not conform to past patterns of behaviour for the data stream [15]. This definition incorporates both point anomalies (or spatial anomalies) as well as temporal anomalies. Anomalies can be spatial, where an individual data instance can be considered as anomalous with respect to the rest of data stream, independent of where it occurs in the time series, for example, a spiking point anomaly happens when a single data point outspreads well above or below the expected range, like the first and third anomalous spikes in figure 1.1. Streaming data commonly also contaminated by temporal, or contextual anomalies, such as a change in the frequency, sudden erratic behaviour of a metric, or other temporal deviations and these anomalies are relevant to temporal sequence of data stream, i.e., a data instance is anomalous only in a specific temporal context, but not otherwise. The middle anomaly of figure 1.1 is an example of temporal anomalies, and such anomalies are often subtle and hard to detect in real time series.

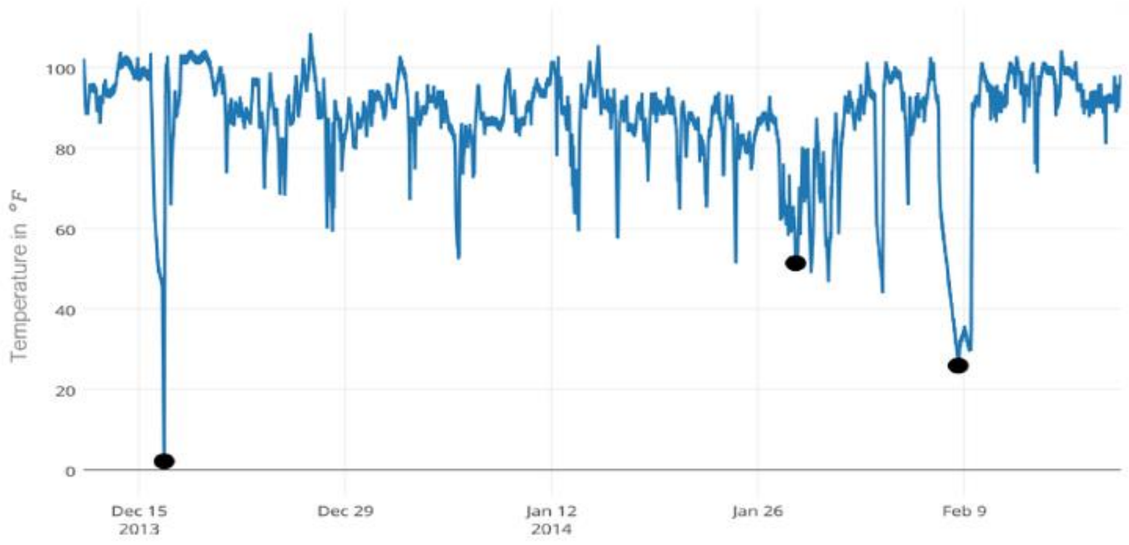


Figure 1.1: The real-world temperature sensor data from an internal component of large industrial machine.

Anomalies are labelled with black circles. The first anomaly was a planned shutdown. A catastrophic system failure was representing as the third anomaly in figure. The second anomaly shows a subtle but observable change in the behaviour, indicated the actual onset of the problem that led to the eventual system failure. Data in figure is included in the Numenta Anomaly Benchmark corpus [6]

As an example, consider the task of monitoring a data processing centre. Let the vector x_t represent the one-dimensional state of a real-time system at time t. The model receives a continuous stream of univariate inputs: $(\dots,\ x_{t-2}, x_{t-1}, x_t, x_{t+1}, x_{t+2}, \dots)$. Component of x_t is one- dimensional and might include CPU usage for various servers. At each point in time t we would like to determine whether the unusual behaviour of the system. The determination must be made in real- time, before next time $t+1$. Such that, before seeing the next input x_{t+1}, the algorithm must consider the current and previous system states to decide whether the system behaviour is normal or anomalous, as well as performing updates and retraining model parameters. Unlike offline processing, data is not split into train/test sets, and algorithms cannot look ahead. Practical streaming applications impose additional constraints on the problem. Typically, the sensor streams are unbounded and at high speed, leaving less opportunity for expert intervention; manual parameter tuning, and data labelling are not viable in such cases. Thus, there is a growing need for developing algorithms that can process these data efficiently in an unsupervised and automated manner.

In many real-world scenarios, the statistics of the data can change over time, a problem known as change point [9] and sometime refers as concept drift [20]. Consider again the example of a data centre. Software configuration changes and upgrades can occur at any time and may alter the behaviour of the system (figure 1.2). In such cases anomaly detection model must adapt to a new definition of “normal” in an unsupervised, automated manner. In streaming applications detecting temporal anomalies in practical streaming applications is valuable and can provide meaningful insights, as they can serve as an early warning for problems with the underlying system.

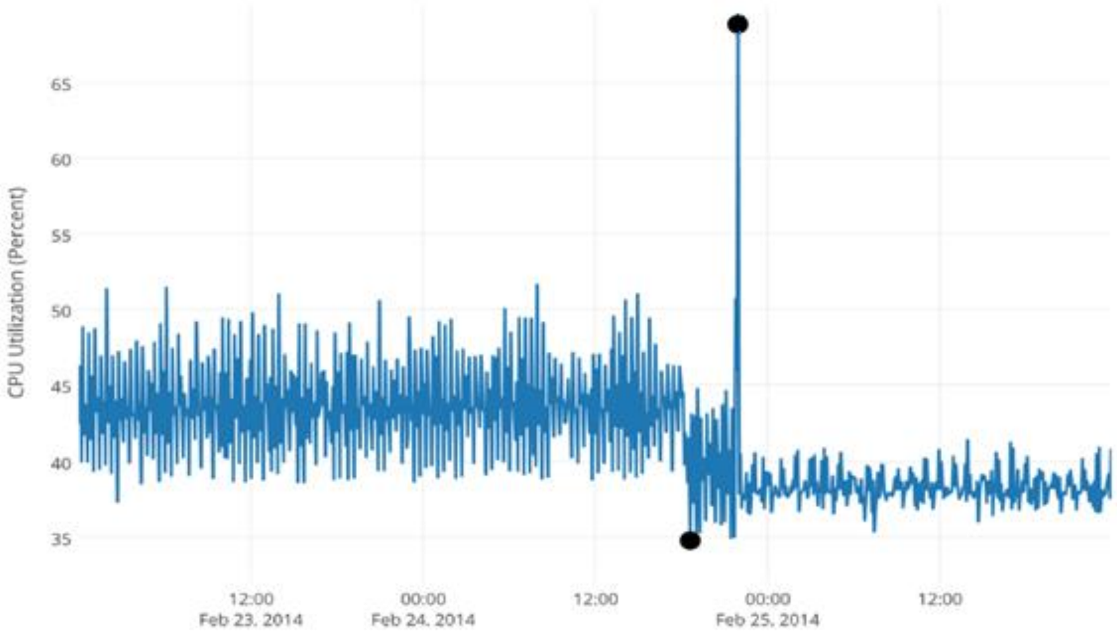


Figure 1.2: CPU utilization for an Amazon EC2 instance.

A change to CPU usage is caused the by modification to the software running on the machine. The initial anomaly embodies a changepoint, and the new system behaviour that follows is an example of concept drift. The second peak is transitory anomaly The dataset is taken from the Numenta Anomaly Benchmark [6]). Incremental learning is essential for performing anomaly and change detection on streaming data like this.

Consider a system that continuously monitors the health of a cardiac patient's heart. An anomaly in the data stream could be a predecessor to a heart attack and detection of such an anomaly few minutes in advance is far better than detecting it some seconds ahead or detecting it after the fact. Detecting anomalies in data stream often gives critical information, and we need this information early enough that it is actionable, possibly prevent the system failure.

Anomaly detection in time series has been extensively researched in data science and machine learning since [5]. Many techniques to anomaly identification exist, both supervised (e.g., support vector machines and decision trees [6]) and unsupervised (e.g., clustering), however the great majority of anomaly detection methods are designed for batch processing and are inappropriate for real-time streaming applications. Industry examples include Netflix's resilient principle component analysis (RPCA) approach [7] and Yahoo's EGADS [8, both of which need comprehensive dataset analysis]. Similarly, Symbolic Aggregate Approximation (SAX) [9] entails decomposing the entire time series to form symbols before detecting anomalies. However, other kernel approaches, such as EXPoSE [23], meet our real-time anomaly detection criterion. The majority of approaches used in practise for detecting streaming anomalies are statistical techniques that are computationally light. Sliding thresholds, outlier tests such as extreme studentized deviation (ESD, also known as Grubbs') and k-sigma (e.g., [24,25]), changepoint detection [26], statistical hypotheses testing, Holt-Winters [27], typicality and eccentricity analysis [28,29] are examples of these approaches. The majority of these strategies are spatially focused, limiting their utility in applications with time dependencies. More sophisticated time-series modelling and forecasting algorithms can uncover temporal abnormalities in complicated circumstances. ARIMA is a general-purpose approach for modelling seasonal temporal data [30]. The approach in [32] based on relative entropy is a more contemporary example that can handle temporal anomalies. Although model-based techniques are frequently computationally efficient, their lack of generalizability restricts their applicability to general streaming applications. Other constraints, such as computing limits that hamper scalability, might render approaches unsuitable for real-time streaming anomaly detection. Dimensionality is another aspect that might limit some approaches.

Online PCA versions, such as osPCA [40] or window-based PCA [41], can only function with high-dimensional, multivariate data streams that can be projected into a low-dimensional space. Techniques requiring data labels, such as supervised classification-based algorithms [42], are often ineffective for real-time anomaly detection and continuous learning [28].

There are many benefits to adopting unsupervised learning as do not require annotated data. Perhaps the most significant benefit is that it can be effective to find hidden patterns of data distribution, which is especially helpful for tasks like anomaly detection, where traditional supervised learning methods may not be able to identify the underlying patterns [28, 29].

Another benefit of unsupervised online learning is that it can be used to pre-train models before using them for supervised tasks. This can improve the accuracy of the anomaly detection models and help reduce the amount of training data required. Another benefit of using the unsupervised online learning and anomaly detection approach is that it does not assume any labels in the data stream except during the burn-in period and assessment of the proposed method. There is less availability of annotated data sets for anomaly and change-point detection that encourage the researchers to utilise the unsupervised approaches for such problem [28, 29].

There is a trade-off between early detections true positive and false positives, as an algorithm that predicts frequent inaccurate detections is likely to be ignored. Given the requirements, a real-world anomaly detection algorithm for univariate time series should have following ideal characteristics:

1. Predictions must be work in online manner, i.e., the algorithm must identify state of x_t from input data stream as normal or anomalous before receiving the sub- sequent x_{t+1}.

2. The algorithm must learn continuously in an online way without a requirement of storing the entire time series.

3. The algorithm must run in an unsupervised, automated manner, i.e., without data labels except assuming labels during, burn-in period and model evaluation. It is assumed that algorithm get the correct answer but only after it makes the prediction.

4. The algorithm must integrate information about all previous data points and incorporate into the current detection.

5. The algorithm must be independent of manual parameter setting and tweaking; the system will not rely on the choice of user-define threshold.

6. Algorithms must adapt to dynamic environments and change points, as the underlying statistics of the streaming data is habitually non-stationary.

7. Algorithm should be able to target the class imbalance/rare class in streaming data and should detect anomalies as early as possible.

8. Algorithms should minimize false positives and false negatives; and maximize the true positives and true negatives.

Taken collectively, the above-mentioned requirements recommend that online anomaly detection for streaming data is a fundamentally different problem than static offline/batch anomaly detection. As discussed in related work given in chapter 2, the majority of existing anomaly detection algorithms are not appropriate for streaming applications, even those designed for streaming data.

1.2 Statement of Purpose

"The purpose of this study is to explore and examine the processes of detecting anomalies using the contemporary recurrent neural networks, such as Long Short-Term Memory Neural Networks (LSTMs) that trained for nonlinear regression of continuous stream of time series in an online setting and predict the underlying patterns from the observed time series for real-time learning."

In particular, this research attempts to address the following research questions denoted by RQ to support the purpose statement and further divided into sub-research questions SRQs:

RQ: *Does the proposed novel online method of anomaly detection for nonlinear regression of time series using LSTM trained in an online setting and predict the underlying patterns from the observed time series for real-time learning is an effective and efficient methodology?*

SRQ1: How and which parameter would be computed and used for detecting an anomaly using incrementally trained LSTM for non-stationary time series?

SRQ2: How and which evaluation parameter would be used to evaluate the performance of proposed model of anomaly detection.

SRQ3: How and which existing anomaly detection would be used for the comparative analysis of proposed model results?

1.3 Aim and Objectives

Anomaly detection and change-point detection has been undertaken using different approaches over time and several machine learning models for streaming data analysis available in literature [15, 20]. It is very challenging task to detect the anomalies for non-annotated streaming data and such case unsupervised methods are required. To best of our knowledge the unsupervised methods for anomaly and change-point detection are model-free and require manual parameters tweaking which is main constraint in *online* process.

The aim of this study is to develop an effective and efficient unsupervised model-based parametric methodology for anomaly detection in continuous time series in an online setting by modelling the noise, such that the model computes the statistic of the error incrementally to detect the anomalies.

To achieve this aim, the objectives of research are defined as follows:

1) To investigate and identify the knowledge gaps by critically analysing the current offline anomaly detection method and need of online anomaly detection method.
2) To provide contemporary RNN [25] based approach to handle both outlier/point anomalies and change points simultaneous in an online way for streaming data.

3) To introduce and evaluate a novel online anomaly detection for non-linear time series regression problem of multistep ahead predictions using LSTMs.
4) To validates the performance achieved by proposed novel online anomaly detection algorithm with respect to conventional methods.
5) To investigates that various architecture of LSTM which gives the better performance.

1.4 The Novel Research Contributions

In a broader sense, we are interested in developing an online temporal anomaly detection model based on Recurrent Neural Networks (RNNs). Specifically, the Long Short Memory Networks (LSTMs) is used for time series anomaly detection in order to address the challenges posed by sudden abrupt changes or permanent changes in normal behaviour. Some examples of online anomaly detection models based on artificial neural networks for streaming data based on ANN are Online Evolving Spiking Neural Network for Unsupervised Anomaly Detection (OeSNN-UAD) [78], DeepAnT: Anomaly Detection Using Deep Learning for Time Series Using CNN [79], Online Disturbance Detection in Satellite Image Time Series (LSTM-SITS) [80], Sparse Recurrent Neural Network Based Anomaly Detection (SPREAD) [81], Online RNN-AD [3]. Most of the online model either detect anomaly or handle change-points [15, 20].

In specifically, we propose and designate an online anomaly detection and change-point detection model based on LSTM that is gradually trained by LSTMs as fresh data streams come and detects anomalies and change-points whenever data points exhibit any suspected aberrant behaviour. The model is trained in online manner using LSTM as new data becomes available, and it is capable of responding to changes in data distribution and predicting multiple time steps using past readings.

Recently, a stacked LSTM networks for anomaly detection in time series have been proposed in [26]. RNNs with LSTM cells are used to model normal time series behaviour on historical normal data in an offline manner by training for multi-step forward prediction. The prediction errors are then fit to a multivariate Gaussian distribution, which is then used to assess the likelihood of anomalous behaviour at a given time instant. [7] proposes a similar technique for detecting anomalies in ECG waveforms. While these techniques employ RNNs for anomaly detection and have been shown to be successful in off-line mode and supervised settings [7, 24], they rely on a model trained offline and do not address the issues of changing data behaviour.

They are based on fixed normal behaviour(s) and may not be appropriate for non-stationary time series. As a result, because they are not learning online, these approaches will likely perceive a shift or abnormality in typical behaviour as aberrant and fail to adapt. Because the standard method for training RNN for modelling time series using historical data is offline [18], it is less likely to operate when real data streams arrive over time with changing behaviour, rendering the model useless and ineffective [27]. These algorithms are trained on full normal data before beginning to detect anomalies and change-points, rendering them ineffectual in a continuously changing environment. As a result, establishing an online model that can learn in real time and adapt to new input data norms is required.

Furthermore, our innovative proposed online anomaly detection model does not rely on any user-specified parameters but is defined and updated automatically by the model itself. Because the proposed anomaly detection strategy aims for totally online performance, we will not assume any labels in the data stream except during the burn-in time and assessment of the proposed method. Furthermore, the prediction errors are utilised to update the proposed model such that anomalies and change points do not result in a severe change in the model but rather quickly adjust to the new norm of input data distribution.

Long Short-Term Memory (LSTM) is a contemporary Recurrent Neural Network (RNN) with advanced recurrent hidden and gated units that is effective and widely used owing to its capacity to learn hidden long-term sequential dependencies and to allow online training of indefinitely long-term sequential data. RNN training with BPTT has yielded encouraging results in online time series modelling and forecasting [21, 22, 71]. Anomaly and change-point detection in streaming data using RNNs trained in online manner is a less researched area [3].

For anomaly and change-point detection, a unique technique is used in which learning of each instance of incoming streaming data is supplied to LSTMs for training of l multi-step forward prediction that produces the l-dimensional error vector. Following that, the online error distribution algorithm developed by [30] is used to detect anomalies and effectively react to changes. One of the unique contributions is that we use the error distribution for anomaly and change-point detection in an online context, which has previously been shown to be successful in off-line mode and supervised settings. Chapter 3 contains all of the technical details.

Because the anomaly and change-point detection model is working online using unseen incoming data point x_t from time series at time t , the model's parameter updating is entirely dependent on *normal tagging* of observation x_t , and the reason for not updating the model's parameter is that we do not intend to learn the anomaly by LSTM model but only wants to detect the anomaly and adapt the changes in data distribution, whether transitory or permanent.

Precisely, following are the novel contributions of research:

Thesis Novel Contributions:

- The proposed online model handle anomalies and change-points simultaneously.
- Model the prediction error to detect anomalies and change-points in an online manner at each time-step with each instance of incoming data stream.
- The unsupervised online anomaly and change-point detection model does not require prior distributional knowledge and annotated data except during evaluation and burn-in learning period of LSTM.
- Does not relying on the user-define threshold. A new feature is added for dynamic threshold estimation to propose novel algorithms which updated with every time step to make it fully online and responds to changing behaviour of data either transitory (i.e., outlier) or permanent (i.e., change points). Threshold is initialized by model after LSTM burn-in period which is than self-adapt.

- Does not lead to a drastic change and adapt to the latest norm. Multi-dimensional normal model of the prediction error is updated.
- The length of the burn-in period is a pre-set parameter

1.5 Thesis Organisation and Contributions

This section summaries the organization and contributions of this thesis. A brief overview of each chapter along with its main contributions is presented next. Anomaly detection and change-point detection have been considered as one of the core research areas for a long time due to its ubiquitous nature [15, 20].

Chapter 2 – Literature Review

This chapter presented some related work on Anomaly Detection (AD) and Change-Point Detection (CPD) as well as an explanation of the importance of unsupervised anomaly detection in the context of univariate time series. This chapter go through the anomaly detection techniques from the perspective of being supervised or unsupervised, offline or online, parametric or non-parametric and model-based or model-free. We mainly consider unsupervised offline and online techniques that specifically trained using neural networks. Preliminary learning concept are given in section 2.2. Related work for anomaly detection for time series is studied in section 2.3 that followed to the literature of online anomaly detection methods in section 2.4. Available literature for time series analysis for anomaly detection using neural networks is analyses in section 2.5. The need for model-based unsupervised non-parametric techniques and dynamic estimation of threshold discussed in section 2.6 and 2.7. Finally, the fundamental structure of Long Short-Term Memory neural network and training through BPTT is explained in section 2.8 and chapter concluded in 2.9.

Chapter 3 –Online Anomaly and Change Detection for Time Series using LSTM: A Methodology

This chapter gives the answer for SRQ1 (Sub research question no. 1). Anomaly detection has been great challenge for researchers and confined by the availability of labelled datasets [16] and most these techniques require user-defined parameters (e.g., threshold) that need to be selected based on the observed data, which limits their pertinence to new unseen data [16].

To address these issues, we propose an *online Anomaly Detection system for univariate time series (OLACD)* based on LSTM [25] in chapter 3 that uses an on-line estimate of the error distribution for anomaly detection that does not require prior distributional knowledge for detecting point anomalies/outliers and change points and effectively handles un-labelled univariate time series.

In the first phase known as *training phase* of proposed novel online technique the learning is take place incrementally as new data stream becomes available and is capable of adapting to the changes in the data distribution. LSTM is used to make single or multi-step predictions of the time series. In the second phase known as *anomaly detection phase* and the prediction errors are used to detect anomalies and change points as well as update the model in an online manner. Large prediction error is used to indicate anomalous behaviour. Further, the prediction errors are used to update the proposed model in such a way that anomalies and change points do not lead to a drastic change in the model but rapidly adapts to the new norm of input data distribution. Our novel proposed unsupervised model is not relying on any user define parameters but automatically defined and updated by model itself.

Chapter 4 –Online Point Anomaly Detection for Time Series

The main objective of the report is to investigate, understand and build a comprehensive knowledge for experimental design and implementation for proposed online anomaly detection model, that justify the SRQ 2 and 3 (Sub research question no. 2 and 3). We consider two real-world and one synthetic univariate time series which are taken from Yahoo Webscope3 [27] to experimentally validate the model for univariate time series and compare the results in term of Area Under Curve (AUC) with algorithm proposed by Saurav *et al.* in 2018 [5], where they found single/multi-dimensional error and compute anomaly score using error vector, afterward the RNN was updated based on anomaly score and pre-process the input data using local normalization

Chapter 5 –Online Change Point Detection for Time Series

This chapter presented the experimental design and implementation of proposed novel anomaly detection technique that utilised to experimentally validate the model for the change point detection of univariate time series, that defend the SRQ 2 and 3.

We validate the efficiency of the proposed novel approach for change point detection of univariate time series via experiments on publicly available and proprietary three real-world [6] and two synthetic datasets [base paper] and evaluate the results in term of Area Under Curve (AUC) with online algorithm proposed by Saurav et al. in 2018 [5].

Chapter 6 – Discussion and Future Work

Learning algorithms often need to operate in dynamic environments, which are changing unexpectedly. One desirable property of these algorithms is their ability of incorporating new data. If the data-generating process is not strictly stationary, the underlying concept, which we are predicting, may be changing over time. The ability to adapt to such *change point* can be seen as a natural extension for the incremental learning systems [31] that learn predictive model example by example. *Adaptive learning algorithms* can be seen as advanced incremental learning algorithms that are able to adapt to evolution of the data-generating process.

In chapter 5 we will conclude our incremental adaptive learning algorithm and as discuss in chapter 2 the anomaly detection is active area of research into unsupervised techniques, so, we will discuss some future work in this chapter.

Chapter 2

Literature Review

Unforeseen unusual patterns in data stream are called anomalies, which are either short-term transitory anomalies/outliers or long-term permanent anomalies/change-points. Detection of such bizarre patterns is important in many applications such as intrusion detection for cyber-security, behavioural analytics for fraud detection and diagnostics for healthcare. Various anomaly detection and change-point detection surveys [10, 20] categorise existing techniques based on their methodology, as well as whether they are supervised, unsupervised or semi-supervised. Supervised learning is reviewed briefly due to its functional challenges, e.g., dealing with a lack of annotated labelled data. We look at anomaly detection methods from the perspective of being devised Artificial Neural Networks. We start with some basic definitions, than discussed anomaly detection techniques for time series using RNN in offline and online manner. We debate the requirement of developing unsupervised model-based anomaly and change-point detection techniques.

2.1 Overview

There is exponential increase in the streaming, time-series data due to sensors infiltrating our everyday lives which is largely driven by the real-time data sources. Effective analyses of these streams can provide valuable insights. A challenge, for both humans and machines, is recognizing an anomaly. Very often the anomaly detection is ill-posed, getting hard to identify and tell what an anomaly is. The anomalies detection in real-time streaming data has significant and practical across many industries such as fault detection, preventative maintenance, fraud prevention, security, IT, medical, e-commerce, agriculture, energy, and social media. Detecting anomalies can give critical and actionable information in different scenarios.

Time series, a sequence of data points consisting of successive measurements made over time, often arises when monitoring dynamic processes [21] and consistent with the definition in [15], anomaly is point in time where the system shows unusual behaviour and significantly different from the normal behaviour. In streaming data or time series signals, an anomaly is any unforeseen change in a pattern, it would be a point in time where the behaviour of the system is strange and unusual and substantially different from previous, normal behaviour. An anomaly may signify a negative change in the system, like possibly indicating an imminent failure of sensor or machine. An anomaly can also be positive, like an abnormally high number of web clicks on a page to buy new product, indicating more than normal demand. Anomalies in data distinguishes the abnormal behaviour with potentially useful information in either way. Anomalies could be spatial, where a specific data point can be deemed as anomalous with respect to the rest of data. An anomaly can also be temporal, or contextual which are often tricky and hard to detect in real data streams. Detecting temporal anomalies is valuable as they can serve as an early warning for problems with the underlying system.

The input dataset is said to be univariate if each data point in a series is single-component vector and respectively multivariate if each data point is a vector with more than one component, where we are considering univariate time series data set throughout the research.

In general case, anomaly detection methods involve initial phase that is called *learning* before actual detections are made. Learning process can be categorized as supervised and unsupervised. During the learning phase, the parameters of the fundamental detection model are decided, and the dataset used for learning phase is known as training dataset. Unsupervised anomaly detection is approach, which does consider the labels. The methods based on unsupervised learning endeavour to detect the anomaly without precise knowledge whether the learning data points are actual anomalies or not. In unsupervised techniques the identification of anomalies is based on their deviation from the other (normal) data points, such that these techniques assume that [15] normal data points are far more frequent than anomalous instances. The primary focus of this thesis lies in the unsupervised methods.
The supervised anomaly detection techniques require label for each data point in the training dataset which they use to learn differences between normal and anomalous instances. The supervised approach in the context of anomaly detection can be assumed as a significantly imbalanced binary classification.

However, the learning process can be deemed by another aspect whether the detection method learns on a static dataset (*offline*) or a streaming dataset (*online*). Learning on the static dataset is a conventional [8, 14] way of learning for anomaly detection algorithms. It is assumed that all essential information can be acquired from the static training dataset prior to making actual detections which we usually refer to as *offline learning*. It means that the dataset must be available all at once and contain sufficient information. This methodology is not appropriate for situations where the complete training dataset is unavailable beforehand. In actual, there are numerous cases (such as medical, agriculture, IT security), where there is a bit to no data in the beginning is available but latest data arrives over time (streaming dataset). The anomaly detection methods must be able to start identifying anomalies as soon as possible with little of initial knowledge and adapt this information as brand-new data are available, which is refer to as *incremental* or *online learning*.

Our research aspires to provide contemporary RNN [1, 25] based approach to handle both outlier/point anomalies and change points simultaneously in an online way for stime series.

The chapter will present preliminary learning concept are given in 2.2. Related work for anomaly detection for time series will presented in 2.3 that followed to the literature of online anomaly detection methods in 2.4. In a nutshell the available literature for time series analysis for anomaly detection using neural networks analyses in 2.5. The need for model-based unsupervised non-parametric techniques and dynamic estimation of threshold debated in 2.6 and 2.7. Finally, the fundamental structure of Long Short-Term Memory neural network and training through BPTT is described in 2.8.

2.2 Preliminaries of Learning Concepts

Machine learning employs two techniques: supervised learning and unsupervised learning. In the context of uncertainty, supervised learning constructs a model that makes predictions based on data. The model is trained on known input and output data so that it can reasonably predict future results. Unsupervised learning, on the other hand, seeks for patterns and structures in datasets. The system learns to make conclusions from unlabelled datasets.

Simply said, a machine learning model learns by repetition and is fed information repeatedly until it improves at carrying out its intended job, such as properly recognising objects, producing accurate weather forecasts, or detecting anomalies and change-points. This may be accomplished using either online or offline (batch) machine learning.

2.2.1. Online Learning

Online machine learning is a type of machine learning in which data is collected sequentially and used to update the best predictor for future data at each step.

In other words, online machine learning occurs as data becomes available, either individually or in small groups known as mini-batches. The parameters of the learning algorithm are modified after learning from each unique training instance in online learning. Each learning step in online learning is rapid and inexpensive, and the model may learn from new information as it enters in real-time. The figure 2.1 depicts the online learning.

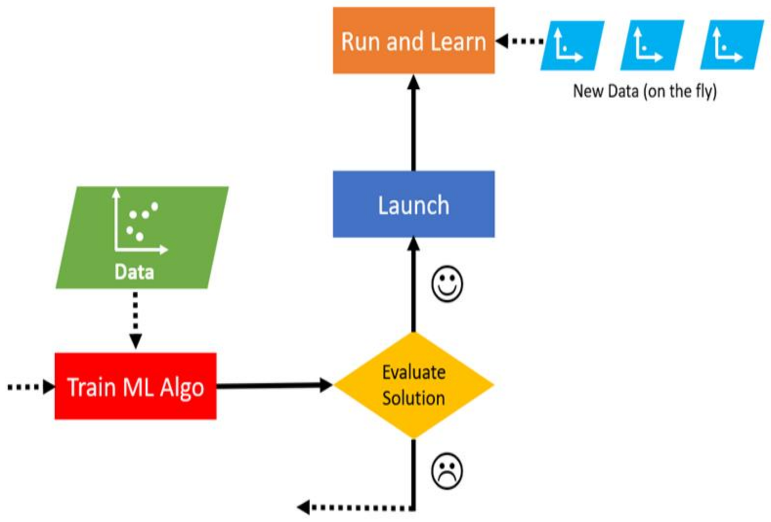


Figure 2.1: Online Learning. Image adapted from [2]

Online learning is appropriate for machine learning systems that need to adapt to quickly changing situations and receive data in a continuous flow. Because models are trained with enormous amounts of collected data, additional time and resources, such as CPU, memory space, and disc input/output, are required. It also takes longer to deploy models to production because this can only be done at specific intervals based on the model's performance after being trained with new data. If a batch learning-trained model has to learn about new data, it must be retrained using the new dataset. This indicates that the fundamental drawback of batch learning over online learning is that batch learning requires a significant amount of time and resources for re-training, whereas online machine learning does not [2, 32].

The changes in weights and parameters that occur at a given phase in an online machine learning process are reliant on the example that is being displayed. If the model has already been deployed, the present condition of the model may also be a factor. As a result, the machine learning model is constantly exposed to new data and may improve itself through learning. While the machine learning model is offline trained and updated for the entire batch of data until it is ready for deployment or the use case that it is designed for [2, 32].

Online learning methods may also be used to train systems on massive datasets that would be too large to put in a single machine's main memory (this is also called out-of-core learning). The algorithm loads a subset of the data, performs a training step on that data, and then continues the process until all of the data has been processed[2, 32].

Online learning is data-efficient and adaptable. The learning rate is an essential characteristic of online learning systems that determines how quickly they should adapt to changing input. If we specify a high learning rate, the system will adapt quickly to new data. However, it has a tendency to forget previous data rapidly, and we don't want a spam filter to highlight just the most recent types of spam it has seen. If we pick a low learning rate, the system will have greater inertia, meaning it will learn more slowly, but it will also be less sensitive to noise in fresh data or sequences of non-representative data points [2, 22].

A big challenge with online learning is that if bad data is fed to the system, the system performances will gradually decline. To diminish this risk, the system needs to be monitored closely and promptly switch learning off and possibly it reverts to a previous working state if drop-in performance is detected. The input data is to be monitored and respond to abnormal data (i.e., using an anomaly detecting algorithm) [10].

A generalised online learning algorithm is demonstrated in Algorithm 2.1. This algorithm can be instantiated by substituting the prediction function f, loss function l and update function Δ [32]. Online learning operates on a sequence of data examples with time stamps. At each step t, the learner receives an incoming example $x_t \in X$ in a d-dimensional vector space, i.e., $X = \mathbb{R}^d$. It first attempts to predict the class label of the incoming instance and $Y = \{-1, +1\}$ for binary classification tasks. After making the prediction, the true label $y_t \in Y$ is revealed, and the learners then computes the loss $l\ (y_t,\ \hat{y}_t)$ based on some criterion to measure the difference between the learner's prediction and the revealed true label y_t. Based on the result of the loss, the learner finally decides when and how to update the classification model at the end of each learning step. The following algorithmic framework gives an overview of most online learning algorithm 2.1 for linear classification, where $\Delta(w_t;\ (x_t, y_t))$ denotes the update of the classification models. Different online learning algorithms in general are distinguished in terms of different definitions and designs of the loss function $l(.)$ and their various updating functions $\Delta(.)$ [32].

Algorithm 2.1: Online Learning

```
Begin
Step 1: Initialise: $w_1 = 0$
Step 2: for $t = 1,2, \dots, T$ do
Step 3: The leaner receives an incoming instance: $x_t \in X$;
Step 4: The learner predicts the class label: $\hat{y}_t = f(x_t;\ w_t)$ ;
Step 5: The true class label is revealed from the environment: $y_t \in Y$ ;
Step 6: The learner calculates the suffered loss: $l\ (w_t;\ (x_t, y_t))$ ;
Step 7: if $l\ (w_t;\ (x_t, y_t)) > 0$ than
Step 8:     The learner updates the classification model: $w_{t+1} \leftarrow w_t + \Delta(w_t;\ (x_t, y_t))$;
Step 9:  end
Step 10: end
```

2.2.2. Incremental Learning

Incremental learning refers to online learning strategies which work with limited memory resources. This eliminates approaches that essentially work in batch mode for inference by storing all examples up to time step t in memory; instead, incremental learning must rely on a compact representation of the previously observed signals, such as efficient statistics of the data, an alternative compact memory model, or an implicit data representation in terms of the model parameters themselves. At the same time, despite its restricted memory resources, it must offer accurate results for all important parameters and relevant settings [33, 34].

Although the definitions of the two concepts are fuzzy, there is a subtle distinction between online learning and incremental learning approaches. In online learning, we merely know the features of the new data point, but without the label; then, respond and suffer the loss or penalty; and then utilise the loss to update the "model". The model here may not be a single model, but rather a collection of models that may be updated or removed as necessary. In incremental "learning," we acquire new labelled data points and attempt to update the previously trained model without using the complete dataset [33, 34].

Both learning approaches (online and incremental) strive to learn (update) a model on the fly in order to acquire the same model as the one learnt in a batch setting (i.e., on static data). Online learning trains a model when training examples arrive one by one (1-by-1), whereas incremental learning updates a model when a fresh batch of data instances arrives. It should be emphasised that all existing online learning frameworks may also be utilised for incremental learning since online learning algorithms can handle a batch of fresh data one at a time [34].

The model is changed in the online learning technique to accommodate fresh data. It is conceivable for the model to forget previously learnt inferences; a condition known as Catastrophic Interference. Whereas in the incremental approach, even as the model is updated, previous inferences are not forgotten. Hence an online learning is always incremental learning but incremental learning does not have to be online [33, 34].

2.2.3. Adaptive Machine Learning

Traditional ML comprises only two basic pipelines: one for training (data gathering) and another for prediction (responsible for data analysis). Before an ML model is released into the wild, it goes through a training phase in which its parameters for data gathering and analysis are determined. Developers employ batch learning approaches to train the model, in which the model receives the complete data set at once and generates the best predictions. Although we generally perceive AI motivations as "always learning, constantly developing," this is usually not the case if the model depends on old batch-based ML approaches. Traditional machine learning is static; it is based on characteristics that do not change, which makes it wonderful for horizontal scaling but causes issues in dynamic settings and sectors where data changes rapidly [15, 35].

Because there are only two pipelines for data gathering and analysis, and because typical ML models rely on prior data to produce new predictions, true, real-time insights are impossible to get in industries such as e-commerce, where trends are continuously changing[15, 35]. To address the problems inherent in classic ML models, researchers often choose one of two approaches:

- Manually training for new data
- Scheduling automatic training for new data

Manual training for fresh data is a time-consuming approach that yields little improvement; thus, most developers use the second method. However, it is far from perfect. Even if automatic training and deployment are planned on a regular basis, the ML model will still be utilising stale data to generate predictions, even if it is only an hour old. To successfully execute a digital transformation and get as near to real-time predictions and learning as feasible, we require a model that is based on adaptive ML. Adaptive machine learning is a more sophisticated method that prioritises real-time data collecting and processing. As the name implies, it adapts quickly to new information and gives insights practically instantly.

Unlike regular ML, adaptive ML uses a single channel rather than two channels or two pipelines. Unlike batch learning, adaptive learning collects and analyses input in a progressive sequence rather than all at once. This permits adaptive ML models to monitor and learn from changes in input and output values; it allows the model to alter its data gathering, grouping, and analysis techniques in response to new information. As a result, as long as data is flowing in, adaptive machine learning models will continue to update and change to produce the best predictions for future data. We may obtain excellent performance and extreme accuracy. Perhaps more crucially, we will have a real-time system that will not become old or obsolete, making the investment of operating AI infrastructure well worth it [15, 35].

Adaptive machine learning has various distinct advantages that might be beneficial to researchers and companies alike. Its key advantages are as follows [15, 35]:

- Robustness and Efficiency: The adaptive ML model's robustness and efficiency stem from its ability to easily manage enormous amounts of data.
- Agility: Its agility is defined by its ability to respond to changes and modify operational circumstances to match your present requirements.
- Accuracy: Because of their single-channel methodology and real-time data collecting and processing capabilities, adaptive ML models may deliver exact insights and forecasts in seconds.
- Sustainability: When all of these benefits are combined, they provide a framework that makes ML models simply scalable and capable of managing enormous datasets in real-time.

However, Adaptive machine learning models are more prone to catastrophic interference where the artificial neural networks tend to forget old information as they acquire new information. Fortunately, this can be easily avoided with online and incremental learning. Adaptive machine learning methods, on the other hand, are more vulnerable to catastrophic interference because artificial neural networks tend to lose old knowledge as they learn new information. Fortunately, with online and gradual learning, this is readily avoided [15, 35].

The most advanced technology on the market is adaptive or online machine learning. It offers a wide range of capabilities and is poised to change the way we collect, analyse, and process data. Adaptive ML is significantly more dynamic than regular ML since it continually learns from changes in both input and output values. As more and more industries start relying on adaptive ML technology, it will become evident just how powerful these models can be [35].

2.3 Related work for Anomaly Detection and Change-Point Detection for Time Series: A Generic Overview

Anomaly detection has been considered as one of the core research areas for a long time due to its ubiquitous nature [9, 15]. The diverse domains and the practical requirement of anomaly detection applications have created a huge literature dedicated to proposing new techniques or improving the existing ones. The common goal of anomaly detection techniques is to find patterns in streaming data that do not conform to the *expected normal* behaviour.

Several anomaly detection (AD) techniques for time series exist in literature [16]. The surveys [9, 15, 20] give the overview of anomaly detection and change-point detection techniques in general and categorise the existing AD techniques based on their methodology (i.e. supervised, semi-supervised or unsupervised, model based or model free, parametric or non-parametric) and their application domain.

In real-world applications, the time series is usually contaminated by anomalies or outliers that are abrupt observations deviating from the normal behaviour [15] where the underlying patterns of time series generally keep changing over time. The presence of outlier and change points can adversely affect the time series analysis and complicate the learning process [20].

Yet the vast majority of methods summarized in [15] are for processing data generally in batches, and unsuitable for real-time streaming applications. Some anomaly detection algorithms are partially online [16]. They either have an initial phase of offline learning or rely on look- ahead to flag of previously seen anomalies. Most clustering based and kernel based unsupervised techniques fall under the umbrella of such algorithms. Some examples include Online Novelty and Drift Detection Algorithm (OLINDDA) [36], Multiclass learning Algorithm for data Streams (MINAS) [37], and Distributed Matching-based Grouping Algorithm (DMGA) [38]. Self-adaptive and dynamic k-means [39] is another example that uses training data to learn weights prior to anomaly detection. Kernel-based recursive least squares (KRLS) [40] violates the principle of no look-ahead as it resolves conditionally flagged data instances a few time steps later to decide if they were anomalous or not.

Examples from industry include Yahoo's Extensible Generic Anomaly Detection System (EGADS) [41] and Netflix's Robust Principle Component Analysis (RPCA) method [42] both of which require analysing the full dataset. Similarly, Symbolic Aggregate Approximation (SAX) [43] decomposing the full time series to generate symbols before detection anomalies. Other recent partial online anomaly and change-point detection techniques includes in [9, 10, 20], although these techniques may work well in certain situations, but they are traditionally batch methods, and the focus of our research is on methods for online anomaly detection. Additional techniques proposed for industrial application includes Facebook Prophet [44], Donut: Anomaly detection method based on variational auto-encoders (VAE) [45], Twitter's open-source technique based on Seasonal Hybrid ESD [46]. Skyline [47] is another popular open-source project, which uses an ensemble of statistical methods for detecting anomalies in time series and HTM JAVA (Hierarchal Temporal Model Implemented in Java) [48], the both projects are part of Numenta Benchmark-NAB [7]).

For anomaly detection of streaming data, the majority of methods used in practice are statistical techniques that are computationally lightweight [28]. Statistical based anomaly detection methods for streaming data includes Extreme Studentized Deviate (ESD) based outlier tests such as (also known as Grubbs') [49] , k-sigma [50], Bayesian changepoint detection [51], Sliding Thresholds [28], Mu-Sigma (Threshold based on mean and standard deviation) [52], One and Two Class SVM for Cloud Computing [53], Gaussian-based Anomaly Detection [54], High Order Statistics (HOS) [55], Tukey Method [56], Relative Entropy Statistics [56], Typicality and eccentricity analysis [57] is an efficient approach as it does not requires any user-defined parameters. Most of these methods focus on spatial anomalies and limiting their usefulness in dealing with long short-term temporal dependencies [28].

Advanced time-series modelling and forecasting models are capable of detecting temporal anomalies in complex scenarios. Autoregressive Integrated Moving Average (ARIMA) [58] is a general-purpose approach for modelling temporal data stream with seasonality and it is effective at detecting anomalies in data steam with regular daily or weekly patterns. Automatic determination of seasonality is enable with extensions of ARIMA for certain applications [58]. A more recent model-based technique based on Tukey method [56] and relative entropy proposed in [56] is capable of handling temporal anomalies and have been developed for specific use cases but require precise domain knowledge and are not generalizable. Model-based methods are often computationally efficient but lack of generalizability of these methods confines their applicability to general streaming applications [16]. Kalman filtering is very a common technique for detecting anomalies for streaming data, but their parameter tuning often requires explicit domain knowledge and depend on the choice of specific residual error models [59].

Dimensionality is another reason that can make some technique restrictive for real-time streaming anomaly detection. For instance, online variants of principle component analysis (osPCA) [60] or window-based PCA [60] that can only effectively work with high-dimensional, multivariate time series but can be projected onto a low dimensional space, techniques that require data labels, i.e., supervised classification-based methods [16], are typically inappropriate for real-time anomaly detection and incremental learning.

Class imbalance [16] is another problem in streaming data analysis in addition to transitory anomalies and change-point, when there are only few anomalous data points in very large stream of data. Research efforts has been made to develop effective learning algorithm for streaming data to tackle anomalies, change points and class imbalance problem at the same time [61] . Synthetic Minority class Oversampling TEchnique (SMOTE) [62] learning with imbalanced class. Another method is Ensemble of Subset Online Sequential Extreme Learning Machine (ESOS-ELM) [63] which uses Online Sequential Extreme Learning Machine (OS-ELM) as a basic classifier to improve the performance with class imbalanced data which is extended to tackle multi-class imbalanced data in [63]. Over sampling based Online Bagging (OOB) and Under sampling-based Online Bagging (UOB) [64] are learning algorithms which build an ensemble model to overcome the class imbalance in real time through resampling and time-decayed metrics. An ensemble method which handles the anomalies and change-points and class imbalance with additional true label data limitation is developed in [65] while learning with random ensemble decision trees is introduced in [66].

Thus, an important characteristic of real-world time series is that the definition of normal behaviour changes over time [28], and as the distribution of data changes, the model built on old data becomes inconsistent with the new data since a model trained on one input distribution is not guaranteed to generalize to the changed new input distribution. Under these circumstances, relying on a model trained in an offline manner on historical data may not be appropriate. Therefore, an online anomaly detection model is required which learn and update models incrementally under dynamically changing and non-stationary environments as new data arrives, in order to better capture the timely trend and the underlying patterns in the time series and does not require user pre-define parameter (i.e., pre-define threshold) and successfully handle the imbalanced class time series.

2.4 Online Anomaly and Change-Point Detection Methods

The identification of anomalies/outliers in temporal data has received a lot of attention in the literature [15]. The basic concept underlying the online anomaly detection and change-point detection methodologies is that the model's parameters are incrementally adjusted as new data comes in order to capture changing normal patterns in the data. Hierarchical Temporal Memory (HTM)-based models [7], self-learning online dynamic clustering approach [67], and Kernel Density Estimates [68], and piecewise linear models of time series [69] are examples of online anomaly detection models. Research in [70] proposes a probabilistic auto-regressive (AR) model for determining both outliers and change points in non-stationary situations. The model is learned incrementally so that it forgets the effect of past data gradually and learns the new normal. In [71] proposes statistical strategies based on the multinomial goodness-of-fit test and the Tukey method. Research in [72] describes a modified k-means clustering and scoring algorithm for detecting abnormalities in periodic time series in real time.

ARMA models, for example, are used to analyse and anticipate future values by regressing the variable on its own previous values (AR) and modelling the error term as a linear mixture of present and past error terms (MA) for a particular time series. The time series can presumably be rendered stationary by differentiating between present and past values (ARIMA) [20]. When seasonality is factored in, we get a SARIMA model. Although there are basic principles [73] and libraries that can automatically identify parameters, these approaches are not flawless and can be quite sluggish [73, 74]

Facebook Prophet is an additive regression model that starts with a unique time series decomposition method $(y(t) = g(t) + s(t) + h(t) + \epsilon_t)$ that includes a piecewise linear or logistic growth curve trend $g(t)$, a yearly seasonal component modelled using Fourier series or a weekly seasonal component $s(t)$, and a user-supplied list of holidays $h(t)$ [44]. The error term is ϵ_t, which is believed to be normally distributed. MAP (maximum a posteriori) optimization is used to determine parameters. By identifying transition points between concept drifts, Prophet may automatically detect changes in trends. If desired, the user can additionally designate custom change points. However, knowing where the transition points are ahead of time is required. Prophet was initially intended for daily time steps and modifications have been made to deal with sub-day time steps, time steps bigger than a day may exhibit unexpected behaviour [74]. Multi-step forecasting RNNs may be trained to predict a window of time steps in the future given a window of time steps in the past. After an initial period of training, the multistep prediction RNN with GRU units described in [5] can adapt to concept drifts.

The RNN is then progressively trained and tested to determine anomaly scores as an average of the prediction error. It is critical to compute the score as an average; a very short-term anomaly will only effect the anomaly score for a limited amount of time. If the anomaly score is continuously high, the RNN is alerted to adjust to the new normal by modifying its settings.

Anomaly Detection (AD) on Twitter [46, 75] discovers anomalies by modifying the extreme studentized deviation test (ESD). As a pre-processing step, the time series is rendered stationary by employing a modified form of STL (seasonal and trend decomposition using LOESS), where the trend component is represented by the time series' median. According to the authors of [75], the mean and standard deviation are extremely susceptible to anomalous data, but the median is statistically more resistant. Thus, instead of the mean, the authors use the median to compute the ESD test statistic, and instead of standard deviation, the authors utilise MAD (median of the absolute deviations from the sample median) to obtain the test statistic. Unfortunately, this strategy may not be suitable for dealing with concept drift. Twitter responds by explaining that anomalies are abnormal data points that occur at a certain point in time, whereas concept drifts (or breakouts) "ramp up from one steady state to another" [46]. The disadvantage of utilising ESD is that an upper constraint on the number of anomalies must be stated, or ESD will presume that up to half of the data is abnormal ([76].

Despite STL's inherent ability to accommodate missing time steps, Twitter's AD library will fail and recommend replacing NANs with interpolated values because Twitter AD utilises R's default *stl* library rather than the *stlplus* library. Donut [45] is a Python-based unsupervised anomaly detection system based on variational auto-encoders (VAE) [45]. VAE is a deep Bayesian network that uses a sliding window over the time series because it is not a sequential model. To train VAEs, the evidence lower bound (ELBO) is generally optimised using Stochastic Gradient Variational Bayes (SGVB).

However, when training the VAE-based model for anomaly detection, abnormal patterns and/or missing points must be avoided as much as possible; thus, the authors use a modified version of ELBO to indicate when an anomaly has occurred (if a supervised task, but not required) or if there are missing points. This change allows Donut to recreate missing points using a missing data imputation approach based on MCMC (Markov chain Monte Carlo). The shortest time step is utilised to resample the time series, according to [45]. The VAE can return the likelihood of a window (using the final data point of the window) and compute a probability density; however, the authors observe that this approach does not function well in reality. As a result, they use the variational posterior to calculate the "reconstruction likelihood" [74]. This probability density is not well-defined. It should be noted that several common anomaly detection "methods" were not taken into account because we do not want to compare frameworks but rather the anomaly detection methods themselves. Yahoo's EGADS [77] is an example of an anomaly detection framework. It consists of three distinct components:

forecasting, detection, and alerting. The forecasting component might be ARIMA, the detecting component could be prediction error, and the alerting component could be k-sigma. LinkedIn's Luminol, Etsy's Skyline, Mentat Innovation's datastream.io, eleme's banshee, Opprentice, and Lytics Anomalyzer are some more notable frameworks [74].

Relative entropy and Pearson correlation were also employed to detect abnormalities dynamically [78]. Many of these strategies ignore the temporal element of time series and instead detect abnormalities using statistical features over a time interval. Our suggested method makes use of the features of RNNs, which are explicitly temporal models capable of coping with temporal dependencies. A lot of studies have recently gone into employing RNNs for anomaly identification in time series [5]. Recent research in [24] investigates incremental training of RNNs using adaptive gradient learning. This study focuses on robust time series forecasting in the face of noise, which includes anomalies and change points. It investigates local statistical aspects of time series in real time to automatically weight the gradients of loss of newly available observations with distributional properties of the data. For online learning in RNN, it employs adaptive Real Time Recurrent Learning (RTRL) [42]. Unlike our technique, the focus of this approach is on time series forecasting rather than anomaly identification.

Hierarchical Temporal Memory (HTM) for anomaly identification provides a neuroscience-inspired machine learning technique that models spatial and temporal patterns in streaming data [7]. HTMs are continually learning systems that automatically adjust to changing statistics and strive to capture the structural and algorithmic qualities of the neocortex. HTMs, on the other hand, do not require multi-step predictions to compute the anomaly score, which may be significant for early detection and hence early adaptation to certain sorts of changes (for example, in the frequency domain) for prediction-based models like HTMs. To the best of our knowledge, our study is one of the few that uses contemporary RNNs trained online for multi-step prediction with applications to online anomaly detection and change-point adaptation.

2.5 Recurrent Neural Network for Time Series Analysis and Anomaly Detection

Recurrent neural networks (RNNs), a class of deep learning methods particularly designed to model sequential data, currently receive increasing attention due to the capacity on learning insightful representations of sequences [1]. In addition, these methods, especially complex RNNs-based methods [19], are capable of effectively processing temporal data [19] and have been successfully applied to time series forecasting [21], especially when there exist sequential dependencies in data [79]. Complex RNNs, e.g., LSTM networks, provide this performance thanks to their memory to keep the past information and several control gates to regulate the information flow inside the network.

Long Short-Term Memory (LSTM) are modern Recurrent Neural Network (RNN), with sophisticated recurrent hidden and gated units, are successful and widespread due to its ability to learn hidden long-term sequential dependencies and to allow online training of sequential data of indefinite duration. Many learning algorithms are proposed in literature for training of these modern recurrent networks i.e., Stochastic Gradient Descent (SGD) [80], Back Propagation Through Time (BPTT) [81], Real Time Recurrent Learning (RTLL)[82] that can train RNN in both ways, either offline or online. BPTT is standard training method for RNN and is a generalization of back-propagation for feed-forward networks [81], it is an instance of automatic differentiation in the reverse accumulation mode [83]. RTLL [82] is more computationally expensive online variant, which is an instance of automatic differentiation in the forward accumulation mode [84]. Along with training algorithm, a good choice of optimization algorithm boosts the training process. Literature in machine learning have several proposed optimization algorithms includes Root Mean Square Propagation (RMSProp) [85], Adaptive Gradient (AdaGrade) [86], Nesterov Accelerated Gradient (NAG) [87], Adaptive Momentum Estimation (Adam) [88], Nesterov-accelerated Adaptive Moment Estimation (Nadam) [89], Adaptive Delta (AdaDelta) [90] and etc.. RMSProp, AdaDelta and Adam are very similar algorithm, but Adam was found to slightly outperform, Adam is generally selected as the overall best choice for RNN training [19].

RNN training using BPTT combining with different optimization algorithms has shown promising results in time series modelling and forecasting in both ways; offline [19] and online [23, 24, 91]. LSTM has been refined and promoted by many researchers in their work [19, 23, 24, 91] and since of 2016, major technology companies including Google, Apple, Microsoft, and Baidu are using LSTM networks as one of the fundamental mechanisms in their products [7].

Anomaly detection and training RNNs in an incremental manner for streaming data is a lesser explored area [5]. A recent work in [24] and [92] explores incremental training of RNNs by using an adaptive gradient learning. The focus of this work is robust time series forecasting in the presence of noise including anomalies and change points. It uses adaptive Real Time Recurrent Learning (RTRL) [92] for online learning in RNN. Recently, a study derived the online RNN learning updates based on stochastic gradient descent, Extended Kalman Filter and Particle Filter algorithms while introducing a new gate in traditional LSTM architecture [91]. An efficient online gradient learning algorithm, Robust Adam (RoAdam) for LSTM is proposed in [23], to predict time series which is modified on the basis of Adam and robust to outliers. These existing approaches do not directly detect the outliers and adaptively tunes the learning rate of the stochastic gradient method when facing suspicious outliers. Unlike our approach, the focus of these approaches is primarily forecasting of time series and not anomaly detection.

Recently, a stacked LSTM networks for anomaly detection in time series have been proposed in [26]. RNNs with LSTM cells are used to model the normal time series behaviour in an offline manner on historical normal data by training for multi-step ahead prediction. The prediction errors are then modelled to fit multivariate Gaussian distribution which is then used to estimate the likelihood of anomalous behaviour at a time instant. Similar approach for anomaly detection in ECG signals is proposed in [8]. While these approaches use RNNs for anomaly detection, they rely on a model trained in an offline fashion and do not address the challenges of changing behaviour of data. They assume fixed normal behaviour(s) and thus may not be suitable for non-stationary time series. Hence these methods would tend to detect a change or anomaly in normal behaviour as anomalous and fail to adapt as they are not learning in an online manner. As the common approach for training RNN for modelling time series using historical data is in an offline manner [18] which is less likely to work where the real data streams come over the time with changing behaviour thus making the model irrelevant and ineffective [27][298]. Some more examples of offline anomaly detection model for streaming data are multi-scale LSTM (MS-LSTM) [93] Convolutional Neural Network (CNN and 1D-CNN) [94], Deep Brief Networks (DBN) [95], Weighted Convolutional Autoencoder LSTM (WCAE-LSTM) [96] , Convolutional Autoencoder LSTM (CAE-LSTM) [96], Attention Based Multi-Flow LSTM(AMF-LSTM) [97]. These techniques are trained on full normal data before actual start of detection anomalies and change-points which make them ineffective for dynamically changing environment, therefor, developing online model is necessary which can learn in online manner and capable to adapt to new norm of input data.

In a wider scope, this section is motivated by the growing need to gain further insights into the underlying processes of detecting anomalies using the modern recurrent neural networks in an online way for continuous stream of time series. Therefore, in this section we focused on the development of online temporal anomaly detection model based on Recurrent Neural Networks (RNNs), specifically, Long Short Memory Networks (LSTMs) for time series anomaly detection to address the challenges posed by sudden abrupt changes or permanent changes in normal behaviour. Some examples of online anomaly detection model based on artificial neural networks for streaming data based on ANN are Online evolving Spiking Neural Network for Unsupervised Anomaly Detection (OeSNN-UAD) [98], DeepAnT: Anomaly Detection Using Deep Learning for Time Series using CNN [29], Online Disturbance Detection in Satellite Images Time Series (LSTM-SITS) [99], Sparse Recurrent Neural Network based Anomaly Detection (SPREAD) [100], Online Anomaly Detection Using RNN (Online RNN-AD) [5]. Most of the online model either detect anomaly or handle change-points [15, 20]. In particular, here we propose and presented Online anomaly detection and change-point detection model using LSTM which is trained by LSTMs incrementally as new data stream arrives and detect the anomalies and change-point if data point shows any suspicious anomalous behaviour. The model is trained incrementally as new data becomes available and is capable of adapting to the changes in the data distribution and able to predict the values for multiple time steps using historical readings. The prediction errors are used to detect anomalies and appropriately adapt to changes. A novel method is being utilized for anomaly detection where LSTM learning of

LSTMs parameters are learned with BPTT while anomaly detector parameters are updated on basis of online learning of Mean and Covariance of error distribution.

2.6 Towards Model-based Unsupervised Anomaly Detection

It is typically assumed that anomalies are rare and different. Many conventional unsupervised anomaly detection techniques are evaluated in the presence of small anomalies in training set, where anomalies are usually point anomalies. This is not appropriate in many real-life applications such as system fault detection or intrusion detection, where anomalies may be seen in set or clusters [16]. Sometime the fraction of anomalies might be higher than point anomalies, but they still are in the minority (i.e., Abnormal ECG signal). Additionally, in numerous applications the streaming data needs to be evaluated as it is produced in real-time and annotation of data instances in real time is often expensive in this context. Furthermore, the normal behaviour of streaming data may change over time, so, an anomaly detection technique must be able to train a precise model even when the training data is contaminated with undetermined anomalies, which is the main functionality of detecting anomalies in an unsupervised manner.

A good anomaly and change-point detection should be model based to avoid storing and analysing a large volume of historical data that make it real or near real-time decisions regarding a new test instance. There are many model-free anomaly detection techniques proposed and studied in literature. Some examples of Kernel-based model-free anomaly detection techniques are Robust Kernel Density Estimation (RKDE) [101], Robust Kernel-Based Local Outlier Detection (RKOF) [102]. Some of available well-studied Clustering-Based model-free anomaly detection methods are Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [103], Lagrangian Relaxation Facility Location with Outliers (LRFLO) [16], Multi Observation Hidden Markov Model (MOHMM) [16]. Some examples of Distance and density-based model-free anomaly detection are Partition-based algorithm for mining outlier (PARTITION) [104], Hilbert Outlier (HilOut) [16], Bhattacharyya Distance Calculation for Anomaly Detection [105], Local Outlier Factor (LOF) [106].

All of above stated model-free techniques are unsupervised. The clustering-based methods with minor modifications are only model-free anomaly detection methods that can be used as model-based techniques. This is because the concise information from clusters can be saved as a model to get rid of the individual training set. Though, clustering methods are designed for other purpose rather than anomaly detection, and thus are not necessarily efficient for this understated task [15].

SmartSifter [107] is a parametric unsupervised model-based method which makes an assumption regarding the data distribution. Although this method is stated to be adaptive to non-stationary environments but its assumption involving the data distribution restricts its application for the data that comes from other distributions. Histogram-based outlier score (HBOS) [108], Balanced iterative reducing and clustering using hierarchies (BIRCH) [109], Outlier Removal Clustering (ORC) [110] , Isolation forest (iForest) [111], Replicator neural networks (RNN) [112] and Unsupervised autoencoder (UAE) [16] are some examples of nonparametric model-based unsupervised methods. HBOS [108] is histogram based and is not able to accurately model the relationships between features for data. BIRCH [109] and ORC [110] are clustering-based and model-based unsupervised techniques and like other clustering-based anomaly detection techniques are not amplified and optimised for anomaly detection in streaming non-stationary environment. Model-based non-parametric unsupervised method like UAE [16] is declared to be robust in modelling normal data when a training set is contaminated with an unseen anomaly, however, UAE [16] require a fairly large training set to train a good model. Model-based non-parametric unsupervised method like iForest [111] is a commonly used anomaly detection technique and has attractive properties, e.g., it does not require a distance or density-based divergence measure to be defined.

Hence, we assume that there are only a few model-based unsupervised non-parametric techniques, and this area requires further research. Although there are hypothetically well-founded model-based semi-supervised techniques, such as Support vector data description (SVDD) [16] and One-class support vector machine(OCSVM) [113] for anomaly detection and new techniques should be established for unsupervised model-based anomaly detection and change-point detection to address the learning challenges when dealing in non-stationary environments with unlabelled training datasets in fully online manner.

2.7 Toward Dynamic Estimation of Threshold

Anomaly detection can also be formulated as a prediction problem. Anomalies are unexpected events, which makes them hard to predict. If we build a system that can predict the value of the next measurement quite well, we can compare that prediction with the actual measurement. If there is a large difference between what is predicted and what is measured, an anomaly probably occurred. Most anomaly detectors need the specification of a threshold value that has a substantial impact on the resultant true positive and false positive rate. The effect on the TPR and FPR, on the other hand, is exactly the reverse. There are several methods available in literature for computing threshold. The threshold estimation method is divided into static and dynamic estimation, that utilises in anomaly detection models.

2.7.1 Statistical threshold estimation

There are many machine learning methods out there to predict values for time-series. Some of them even come with tools to estimate confidence boundaries like ARIMA or Gaussian Processes. The output is typically a Gaussian, which tells how likely it is that the actual measurement will fall between certain boundaries. A commonly used approach is the 3 Sigma rule. If measurement is more than three standard deviations away from the average, this measurement is considered an anomaly. But changing the accepted standard deviation is like setting the false-positive rate. It does not capture whether a measurement, in this case, a response time, is an anomaly or not [15, 114].

2.7.2 Anomaly score

Some approaches, such as Robust Random Cut Forest (RRCF), do not operate with Gaussian bounds [15]. RRCF is a tree-based approach for modelling data. When a new data point is added into the model, it checks to see if any adjustments are required to better fit the data. If the prediction was accurate, no changes are required, but if the prediction deviated significantly, the tree would need to be adapted to fit the model better. RRCF returns an anomaly score that measures the change the model had to do to fit the data. If the tree in the model has a size of 256 (the default), the score can range anywhere between 0 and 256. Small modifications to the model result in a low score, but if the entire tree is modified, it can approach 256. If a range of 0 to 100% is desired, just divide the result by the tree's size.

In general, a threshold can be set to, say, a 50% change in the tree. However, the problem of too many false positive predictions persisted, despite the fact that it was necessary to define a threshold for each statistic in order to get the optimal performance. The problem was determining the appropriate tree size for each of the criteria. More stable measures seldom used the complete tree, whereas noisy data saw many modifications just to mimic the predicted volatility in the data [20, 114].

2.7.3 Scaled minmax threshold estimation

To tackle the thresholding problem, this is a different approach. The idea is that the learning from past data what a good threshold looks like. If this reasoning is applied to the output of the anomaly score of RRCF, it can ignore the negative values in the examples that follow.

The anomaly detection model achieved best results in practice by setting the detection thresholds to 1.5 times the maximum (and minimum) deviation measured. The reason for this is that only significantly larger deviations than past deviations trigger an anomaly [114, 115].

2.7.4 Threshold estimation by maximizing $\boldsymbol{F_\beta - score}$

The anomaly detection method that utilises the modern RNN based prediction model. The prediction model learned on normal training data sequences is used to compute the error vectors for each point in the validation and test sequences. The error vectors are modelled to fit a multivariate Gaussian distribution $\mathcal{N} = \mathcal{N}(\mu\ \Sigma)$. The likelihood p of observing an error vector

e is given by the value of $\mathcal{N}$ at e. The error vectors for the points from normal validation sequence are used to estimate the parameters μ and Σ using Maximum Likelihood Estimation. An observation $x(t)$ is classified as 'anomalous' if $p(t) < \tau$, else the observation is classified as 'normal'. Some normal and anomalous validation sets are used to learn τ by maximizing $F_\beta - score$ (where anomalous points belong to positive class and normal points belong to negative class) [8, 26].

Most of the static threshold estimation are based in the 3-sigma rule but the exhibit the problem of higher false-positive rate. The anomaly detection method that utilises the modern RNN based prediction model estimates the threshold by maximizing $F_\beta - score$ using normal and anomalous data sequence. These estimations are publicised as outperformed in an offline manner but not suitable for online anomaly detection. The Anomaly score and min-max threshold are the dynamic threshold estimation methods but still exhibit the problem of large false positive predictions [8, 26].

The proposed anomaly detection and change-point detection technique is striving for fully on-line performance; therefore, our novel proposed online model is not relying on any user define parameters but automatically defined/initialized after burn-in period and updated by model itself at each time step. The method for initializing the threshold after burn-in period and that online updating at each time step gives the promising results given in table 4.5 and 5.5 specially in multi-step ahead predictions.

As the anomaly and change-point detection model is working online using unseen incoming data point x_t from time series at time t, the updation of all model's parameters including threshold is entirely depends on *normal tagging* of observation x_t, and the reason for not updating the model's parameter is that we are not intended to learn the anomaly by LSTM but only wants to detect the anomaly and adapt the changes either transitory or permanent in data distribution. In that way we prevent the model to lead to a drastic change but adapt to the latest norm.

2.8 Background: LSTM Recurrent Neural Network Learning Based on BPTT for Time Series Prediction

Long Short-Term Memory (LSTM) [19]; is an recurrent neural network architecture that elegantly addresses the vanishing gradient problem using “memory units”. These linear units have a self-connection of strength 1 and a pair of auxiliary “gating units” that control the flow of information from the unit. When the gating units are shut, the gradient can flow through the memory units without alteration for an indefinite amount of time, thus overcoming the vanishing gradient problem. While the gates never isolate the memory units in practice, this reasoning shows that the LSTM addresses the vanishing gradient problem in at least some situations, and indeed, the LSTM easily solve a number of synthetic problems with pathological long-range temporal dependencies that were previously believed to be unsolved by standard RNNs. LSTM has been refined and promoted by many researchers in their work [1, 8, 19] and is now widely used. As of 2016, major technology companies including Google, Apple, Microsoft, and Baidu are using LSTM networks as fundamental components in new products [116-119].

LSTMs are similar to standard RNN; LSTMs are chained together in a form of repeating modules, figure 2.2. Instead of having a single neural activation function like in standard RNN, LSTM repeating module have various functionalities, interacting in a very special way. Figure 2.2 shows the single cell model and 2.3 illustrations chain structure.

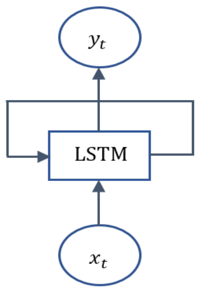


Figure 2.2: Depiction of LSTM Cell That Have Loop.

In figure 2.2, a single LSTM neural network is unfolded through time, “LSTM” looks at input x and outputs value y over different time periods. A feedback loop, as shown on the left-hand side of figure 2.3, when unfolded through time allows information to be passed from one time step t of the network to the next.

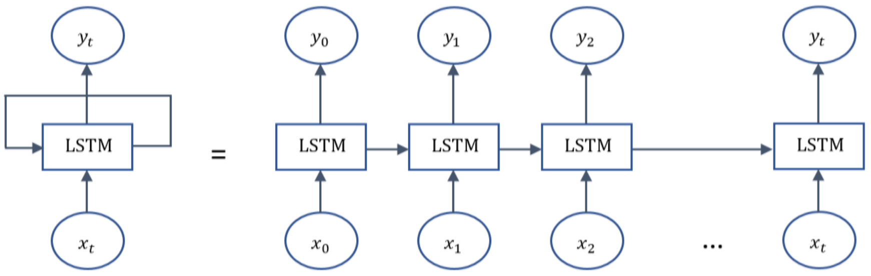


Figure 2.3: Depiction of an Unrolled LSTMs Cell with time.

Fundamentally, LSTMs are explicitly designed to be embedded into multi-layer structure neural network model involving at least one hidden layer. The principle unit in the hidden layer of LSTM network is the memory cell, which replaces the hidden unit in "traditional" RNN in figure 2.2.

The memory cell contains three adaptive, multiplicative gating units which gate input, forget and output to all cells. Each individual memory cell has at its core recurrent self-connected linear unit called the "Constant Error Carousel" (CECs), whose activation we call the cell state. The architecture of single LSTM unit is shown in figure 2.4.

The CECs resolve the vanishing gradient problem as in the absence of new input or error signal to the memory cell the CECs local error back flow remaining constant; neither growing nor decaying. The CECs is protected from both forward flowing activation and backward flowing error by the input and output gate respectively. When gates are closed (activation-output is zero), irrelevant input and noise do not enter the cell, and the cell state does not agitate the remainder of the network

Corresponding to all other RNN which trained with gradient descent, the LSTM also has two passes or phases: forward phase (activation of units) and learning phase (error calculations and weight updates). These phases would be explained in detail in the sections.

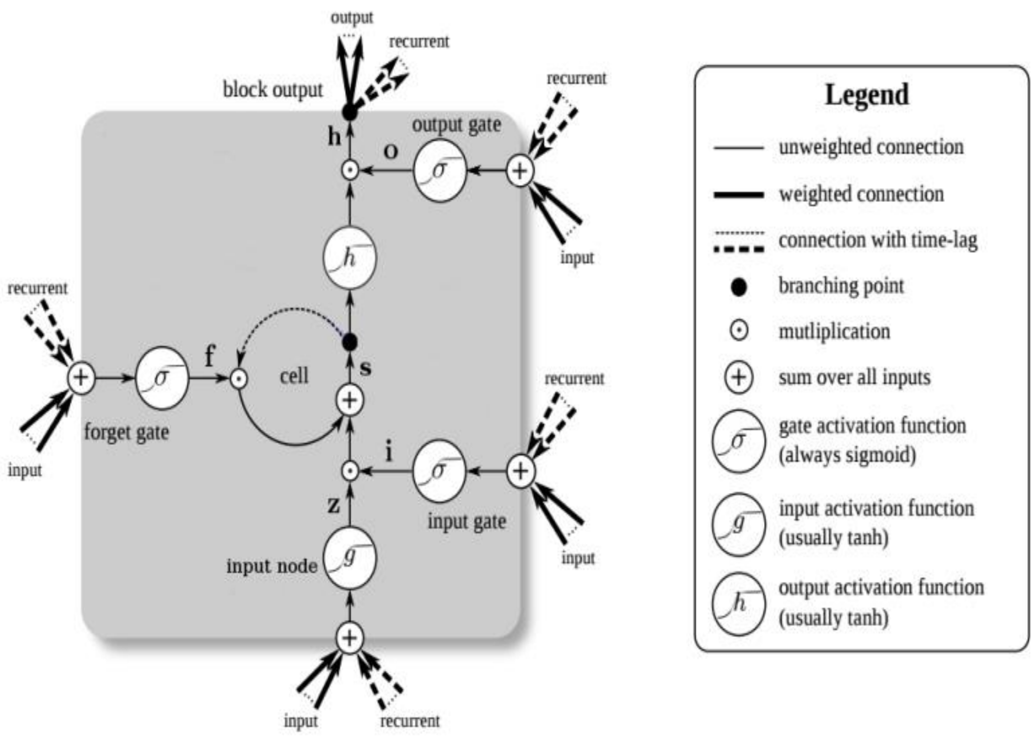


Figure 2.4: A LSTM unit: A schematic diagram of the LSTM unit as introduces in [1].

2.8.1 LSTM Forward Phase

Forward phase is also known as activation of units. Technically the figure 2.4 and 2.5 are same. Figure 2.5 describes the traditional view of forward phase of LSTM; which used by many researchers [120, 121] to explain the computational flow of their mathematical functions. Figure 2.5 shows that x_t is input, c_{t-1} and h_{t-1} are the cell state and hidden state at time $t-1$. The output of cell at time t is c_t and h_t. Further explanation of two phases of LSTM are given below

Let the input at time t in the LSTM cell be x_t, the cell state from time $t-1$ and t be c_{t-1} and c_t and the output for time $t-1$ and t be h_{t-1} and h_t . The initial value of c_t and h_t at t will be zero.

.

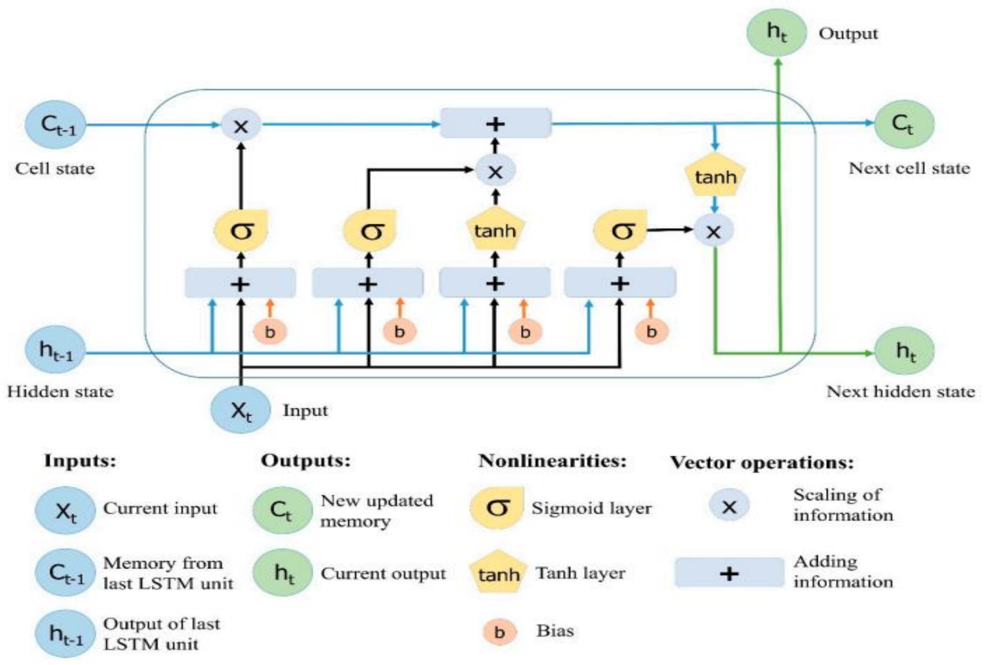


Figure 2.5: Structure of the LSTM unit. Image adapted from [4].

Step 1: Initialization of the weights.

Weights for input gate: w_{xi} , w_{xg} , b_i , w_{hi} , w_{hg} , b_g

Weights for forget gate: w_{xf} , w_{hf} , b_f

Weights for output gate: w_{xo} , w_{ho} , b_o

Step 2: Passing through different gates.

Inputs: x_t, h_{t-1} and c_{t-1} are given to the LSTM cell

Passing through input gate:

$$z_g = w_{xg} * x_t + w_{hg} * h_{t-1} + b_g \tag{2.1}$$

$$g = \tanh(z_g) \tag{2.2}$$

$$z_i = w_{xi} * x_t + w_{hi} * h_{t-1} + b_i \tag{2.3}$$

$$i = \text{sigmoid}(z_i) \tag{2.4}$$

$$input\ gating = g * i \tag{2.5}$$

Passing through forget gate:

$$z_f = w_{xf} * x_t + w_{hf} * h_{t-1} + b_f \tag{2.6}$$

$$f = sigmoid(z_f) \tag{2.7}$$

$$forget\ gating = f \tag{2.8}$$

Passing through the output gate:

$$z_o = w_{xo} * x_t + w_{ho} * h_{t-1} + b_o \tag{2.9}$$

$$o = sigmoid(z_o) \tag{2.10}$$

$$output\ gating = o \tag{2.11}$$

Step 3: Calculating the output h_t and current cell state c_t.

Calculating the current cell state $\boldsymbol{c_t}$:

$$c_t = (c_{t-1} * forgeting\ gating) + input\ gating \tag{2.12}$$

Calculating the output gate $\boldsymbol{h_t}$:

$$h_t = output\ gating * \tanh(c_t) \tag{2.13}$$

This concludes the forward phase of LSTM.

2.8.2 LSTM Learning/Backpropagation Phase:

Learning phase involve in calculating error and weights updation for LSTM network. Fundamentally, as in BPTT, once an error signal arrives at a memory cell ouput, it get scaled by the semilinear output function h; subequnelty, it enters the memory cell's linear CEC, where it can flow back. Once the error escape from the memory cell through an opening input gate; it get scaled once more by the aditional semilinear function g, and then serves to chage incoming weight befor being truncated.

Gradient is calculated through the back propagation through time at time stamp t using the chain rule.

Let the gradient pass down by the above cell be given in figure 2.5:

$$\Delta E = \frac{dE}{dh_t} \tag{2.14}$$

If we are using MSE (mean square error) for error then,

$$\Delta E = (y_t - h_t) \tag{2.15}$$

Here y_t is the original value and h_t is the predicted value at time t.

Gradient with respect to output gate

$$\frac{dE}{do} = \frac{dE}{dh_t} * \frac{dh_t}{do} = \Delta E * \frac{dh_t}{do} \tag{2.16}$$

$$\frac{dE}{do} = \Delta E * \tanh(c_t) \tag{2.17}$$

Gradient with respect to c_t

$$\frac{dE}{dc_t} = \frac{dE}{dh_t} * \frac{dh_t}{dc_t} = \Delta E * \frac{dh_t}{dc_t} \tag{2.18}$$

$$\frac{dE}{dc_t} = \Delta E * o * (1 - \tanh^2(c_t)) \tag{2.19}$$

Gradient with respect to input gate $\frac{dE}{di}$, $\frac{dE}{dg}$

$$\frac{dE}{di} = \frac{dE}{dc_t} * \frac{dc_t}{di} \tag{2.20}$$

$$\frac{dE}{di} = \Delta E * o * (1 - \tanh^2(c_t)) * g \tag{2.21}$$

Similarly,

$$\frac{dE}{dg} = \Delta E * o * (1 - \tanh^2(c_t)) * i \tag{2.22}$$

Gradient with respect to forget gate

$$\frac{dE}{df} = \Delta E * \frac{dE}{dc_t} * \frac{dc_t}{dt} \tag{2.23}$$

$$\frac{dE}{df} = \Delta E * o * (1 - \tanh^2(c_t)) * c_{t-1} \tag{2.24}$$

Gradient with respect to c_{t-1}

$$\frac{dE}{dc_{t-1}} = \Delta E * \frac{dE}{dc_t} * \frac{dc_t}{dc_{t-1}} \tag{2.25}$$

$$\frac{dE}{dc_{t-1}} = \Delta E * o * (1 - \tanh^2(c_t)) * f \tag{2.26}$$

Gradient with respect to output gate weights:

$$\frac{dE}{dw_{xo}} = \frac{dE}{d0} * \frac{do}{dw_{xo}} = \Delta E * \tanh(c_t) * sigmoid(z_o) * (1 - sigmoid(z_o)) * x_t \tag{2.27}$$

$$\frac{dE}{dw_{ho}} = \frac{dE}{d0} * \frac{do}{dw_{ho}} = \Delta E * \tanh(c_t) * sigmoid(z_o) * (1 - sigmoid(z_o)) * h_{t-1} \tag{2.28}$$

$$\frac{dE}{db_o} = \frac{dE}{d0} * \frac{do}{db_o} = \Delta E * \tanh(c_t) * sigmoid(z_o) * (1 - sigmoid(z_o)) \tag{2.29}$$

Gradient with respect to forget gate weights:

$$\frac{dE}{dw_{xf}} = \frac{dE}{df} * \frac{df}{dw_{xf}} \tag{2.30}$$

$$\frac{dE}{dw_{xf}} = \Delta E * o * (1 - tanh^2(c_t)) * c_{t-1} * sigmoid\left(z_f\right) * (1 - sigmoid(z_f)) * x_t \tag{2.31}$$

$$\frac{dE}{dw_{hf}} = \frac{dE}{df} * \frac{df}{dw_{hf}} \tag{2.32}$$

$$\frac{dE}{dw_{hf}} = \Delta E * o * \left(1 - tanh^2(c_t)\right) * c_{t-1} * sigmoid\left(z_f\right) * (1 - igmoid(z_f)) * h_{t-1} \tag{2.33}$$

$$\frac{dE}{db_0} = \frac{dE}{df} * \frac{df}{db_0} \tag{2.34}$$

$$\frac{dE}{db_0} = \Delta E * o * (1 - tanh^2(c_t)) * c_{t-1} * sigmoid\left(z_f\right) * 1 - sigmoid(z_f)) \tag{2.35}$$

Gradient with respect to input gate weights:

$$\frac{dE}{dw_{xi}} = \frac{dE}{di} * \frac{di}{dw_{xi}} \tag{2.36}$$

$$\frac{dE}{dw_{xi}} = \Delta E * o * (1 - tanh^2(c_t)) * g * sigmoid(z_i) * (1 - sigmoid(z_i)) * x_t \tag{2.37}$$

$$\frac{dE}{dw_{hi}} = \frac{dE}{di} * \frac{di}{dw_{hi}} \tag{2.38}$$

$$\frac{dE}{dw_{hi}} = \Delta E * o * (1 - tanh^2(c_t)) * g * sigmoid(z_i) * (1 - sigmoid(z_i)) * h_{t-1} \tag{2.39}$$

$$\frac{dE}{db_i} = \frac{dE}{di} * \frac{di}{db_i} \tag{2.40}$$

$$\frac{dE}{db_i} = \Delta E * o * (1 - tanh^2(c_t)) * g * sigmoid(z_i) * 1 - sigmoid(z_i)) \tag{2.41}$$

$$\frac{dE}{dw_{xg}} = \frac{dE}{dg} * \frac{dg}{dw_{xg}} \tag{2.42}$$

$$\frac{dE}{dw_{xg}} = \Delta E * o * \left(1 - tanh^2(c_t)\right) * i * \left(1 - tanh^2\left(z_g\right)\right) * x_t \tag{2.43}$$

$$\frac{dE}{dw_{hg}} = \frac{dE}{dg} * \frac{dg}{dw_{hg}} \tag{2.44}$$

$$\frac{dE}{dw_{hg}} = \Delta E * o * \left(1 - tanh^2(c_t)\right) * i * \left(1 - tanh^2\left(z_g\right)\right) * h_{t-1} \tag{2.45}$$

$$\frac{dE}{db_g} = \frac{dE}{dg} * \frac{dg}{db_g} \tag{2.46}$$

$$\frac{dE}{db_g} = \Delta E * o * (\left(1 - tanh^2(c_t)\right) * i * \left(1 - tanh^2\left(z_g\right)\right) \tag{2.47}$$

Subsequently that all weights connections of LSTM model are updated and modified using the gradient values derived in equations (2.16-6.47) using (6.21):

$$\vec{\mathrm{w}}\ (\text{updated}) = \mathrm{w(old)} + \text{gradient}(w) \tag{2.48}$$

The above described steps of training is summarised in generalized algorithm for in algorithm 2.2.

Algorithm 2.2: Long Short-Term Memory (LSTM)

Input: Data instances from Time Series

Output: Single/Multi-step Predictions

```
begin
Step 1: Initialise input data weight and bias: $w_{xg}, w_{xi}, b_g, b_i, w_{xf}, b_f, w_{xo}, b_o$
Step 2: Initialise recurrent weight: $w_{hg}, w_{hi}, w_{hf}, w_{ho}$
Step 3: Set initial LSTM cell state $c_{t-1}$ and cell output $h_{t-1}$
Step 4: At time $t$ the $x_t, h_{t-1}$ and $c_{t-1}$ are given to the LSTM cell
// Forward phase of LSTM
Step 5: Input and input gate computation
        $input: \; g = tanh(w_{xg} * x_t + w_{hg} * h_{t-1} + b_g)$
        $input\ gate \; i = \sigma(w_{xi} * x_t + w_{hi} * h_{t-1} + b_i)$
Step 6: Forget gate computation
        $forget\ gate: \; f = \sigma(w_{xf} * x_t + w_{hf} * h_{t-1} + b_f)$
Step 7: Output gate computation
        $output\ gate: \; o_t = \sigma(w_{xo} * x_t + w_{ho} * h_{t-1} + b_o)$
Step 8: Computing cell state
        $cell\ sate: \; c_t = (c_{t-1} \odot f) + (g \odot i)$
Step 9: Computing output
        $output: \; h_t = tanh(c_t) \odot o$
//End of forward Phase
//Start of backpropagation phase of LSTM using BPTT
Step 10: Gradient with respect to output gate using equation 2.17
Step 11: Gradient with respect to $\boldsymbol{c_t}$ and o $\boldsymbol{c_{t-1}}$ using equation 2.19 and 2.26
Step 12: Gradient with respect to input gate using equation 2.21 and 2.22
Step 13: Gradient with respect to forget gate using equation 2.24
Step 14: Gradient with respect to output gate, forget gate, input gate using equations 2.27-2.47
Step 15: Update weights using equation 2.48
//End of backpropagation phase of LSTM using BPTT
End
```

2.9 Summary

In this chapter, we reviewed anomaly detection techniques from the perspective of being offline and online anomaly detection that are model-based or model-free. Although there have been several studies on semi-supervised model-based methods and unsupervised model-free, but there is less focus on unsupervised model-based anomaly detection. However, growing requirement for online processing of data as it is generated in real-time in dynamically changing environments, makes a demand for effective and efficient model-based anomaly detection. Afterward a section is dedicated to exploring the online anomaly detection method. Consequently, we devote the subsequent chapters of this thesis to developing online model-based unsupervised anomaly detection technique for unlabelled datasets in non-stationary environments. Need of dynamic threshold estimation of model is discussed in subsequent sections that follows the detailed LSTM training using BPTT. We investigate the effectiveness of proposed novel methods for anomaly detection in this thesis.

Chapter 3

Online Anomaly Detection and Change Detection for Univariate Time Series Using LSTM: A Methodology

The approach for anomaly and change-point detection is developed in the presented chapter which will be tested for Transitory Anomalies/Outlier detection in Chapter 4 and for Permanent Anomalies/Change-Point detection in Chapter 5. The section 3.1 gives introduction to the chapter followed by the approach which is discussed in step-by-step detail in section 3.2. Performance evaluation metrics will be discussed in section 3.3 and followed by the chapter summary which is given in section 3.4.

3.1 Introduction

Mankind is on the quest of digital world, and we are encountered with tremendous increase in the accessibility of streaming, time-series data which is essentially driven by the upsurge of interconnected real-time data sources. This derived streaming data presents technical challenges and opportunities. Though, the real streaming data is habitually complicated with anomalies and change points, which can lead the training and detecting models deviating from the underlying patterns of the time series, particularly when the model trains in the context of online learning mode.

A significant problem for streaming analytics is an *anomaly* detection which identifies the points when the underlying distribution of a time series abruptly changes and deviate from normal behaviour, either transitory (point anomaly) or permanent (change-point). Therefore, the fundamental capability of streaming data analysis is to learn and model streaming data in an unsupervised fashion and detect bizarre, anomalous behaviours in real-time. The communal approaches of training one model in an offline way using historic data are probable to fail under non-stationary and dynamically changing environments where the definition of normal behaviour of streaming data changes over time and making the model irrelevant and ineffective to *streaming* especially for very *imbalance class* and *rare-class* time series [16].

Nevertheless, several shortcomings confine to the use of some existing anomaly and change-point detection methods in real-world applications. First, several anomaly detection techniques work in offline manner, where the entire time series needs to be stored before anomaly detection can be performed [9, 26]. These approaches of training one model in an offline way using historical data are probable to fail under non-stationary and dynamically changing environments where the definition of normal behaviour of streaming data changes over time making the model irrelevant and ineffective to *streaming* especially for very *imbalance class* and *rare-class* time series [16]. Second, the development made in for anomaly detection has been mostly based on supervised machine learning algorithms that require labelled datasets. Though, collecting and annotating labelled streaming datasets is difficult, time-consuming, and even too costly, while it requires domain knowledge from experts. Therefore, anomaly detection has been great challenge for researchers and constrained by the availability of labelled datasets [16]. Finally, most anomaly detection techniques require user-defined parameters (e.g., threshold) that need to be chosen based on the observed data, which limits their applicability to new unseen data [16]. To address these issues, we propose an online Anomaly Detection and change detection system for univariate time series based on LSTM [1] that uses an on-line estimate of the error distribution for anomaly detection that does not require prior distributional knowledge for detecting point anomalies/outliers and change points and effectively handles un-labelled univariate time series.

The proposed online model is trained incrementally as new data stream becomes available and is capable of adapting to the changes in the data distribution. LSTM is used to make single or multi-step predictions of the time series, and the prediction errors are used to detect anomalies and change points as well as update the model. Large prediction error is used to indicate anomalous behaviour. Further, the prediction errors are used to update the proposed model in such a way that anomalies and change points do not lead to a drastic change in the model but rapidly adapts to the new norm of input data distribution. The idea of using prediction error for anomaly detection comes from [8, 26]. But publicised approach in [8, 26] produces the challenge for streaming data as firstly it is an off-line approach to be effective in off-line mode and not dealing with the streaming data in an online manner and secondly, it worked only in supervised setting such that all the parameters (e.g., threshold) selection and tweaking was pre-defined and done manually on the bases of labelled time series that explicitly showing normal behaviour, finally, this off-line approach only works for short term anomalies and not able to handle the permanent changes in data stream.

To actually detecting the anomalies and change-point and initializing the anomaly detecting parameters, model must be train for burn-in period using the LSTM. Whereas it must be made sure that during the burn-in period the model must learn with only the normal data for that reason label are used. So, we can say that anomaly detection model does not assume any labels in the data stream learning except during evaluation and burn-in learning period of LSTM. After burn-in period of training with only normal data model starts actual anomaly detection. We assume that our algorithms get the correct answer either normal or anomalous but only after model make the prediction. It is online because we see what actually happened and we see what we predicted would happen, and if they are sufficiently different, we call it an anomaly. We don't know what should have happened. So, it is online. The proposed anomaly detection and change-point detection technique is striving for fully on-line performance; therefore, our novel proposed online model is not relying on any user define parameters but automatically defined and updated by model itself.

This chapter presented the proposed novel approach, which consists of two fundamental phases: *LSTM learning phase* that train in online way to generate prediction error and *anomaly detection phase* that use statistics of error to detect different anomalies either short-term or long-term (section 3.2). We validate the effectiveness of the proposed online novel approach via experiments on publicly available three real-world and four synthetic benchmark datasets [6, 27] and compare the results in term of Area Under Curve (AUC) with state of-the-art algorithm available in literature [5] and are demonstrated in chapter 4 and 5.

3.2 The Proposed Novel Online Approach for Anomaly Detection and Change-point Detection in Time Series

We assume that the LSTM model for the time series is that model which is able to predict the next few steps i.e., 1 or 5 or 10 timesteps into the future based on history. Specifically, if a time series forecasting model is able to predict the values at next few steps in future well, it has effectively learned to capture the significant temporal characteristics in the time series.

Presuming for stationary time series, once trained offline or in batch, such a model is likely to outperform [8, 26] and predict the time series delineating normal behaviour but perform inadequately on the time series prediction task corresponding to anomalous behaviour. This idea has been effectively applied to several domains where a temporal model for normal behaviour is learned based on LSTM, as LSTM performs well on temporal data due to its parameter efficiency and capability to extract long-term trends in the encountered univariate time series [19].

In the online way, such a model of normal behaviour needs to be updated in an incremental manner as the true value becomes available, the error is backpropagated through the network to improve the prediction. At same time, we use the most recent prediction error(s) to serve following primary purposes in an online manner:

i) The prediction error vector is modelled to compute the statistics of error (i.e., mean μ , and covariance V).
ii) The multi-dimensional prediction error(s) are modelled to fit a multivariate Gaussian distribution $\mathcal{N} = \mathcal{N}(\mu\, V)$. which is then used to estimate the likelihood of anomalous behaviour at a time instant.
iii) Multi-dimensional normal model of the prediction error is updated.

The dimensions of error model are determined by the prediction horizon i.e. 1-d in case of one step ahead prediction, 5-d or 10-d in case of 5 or 10 step ahead predictions of time series. When the size of the prediction error is large, the probability of the observed error will be very small, indicating an anomaly. Significant properties of a good online anomaly detection model based on prediction errors are following:

- If there is a small chunk of anomalous values for example point anomalies, the model should not be updated. Rather the updation step for model's parameters should be restrained.
- If there is a change in streaming values being observed over a significant period of time, the model should be able to quickly adapt to the latest norm by updating the parameters.

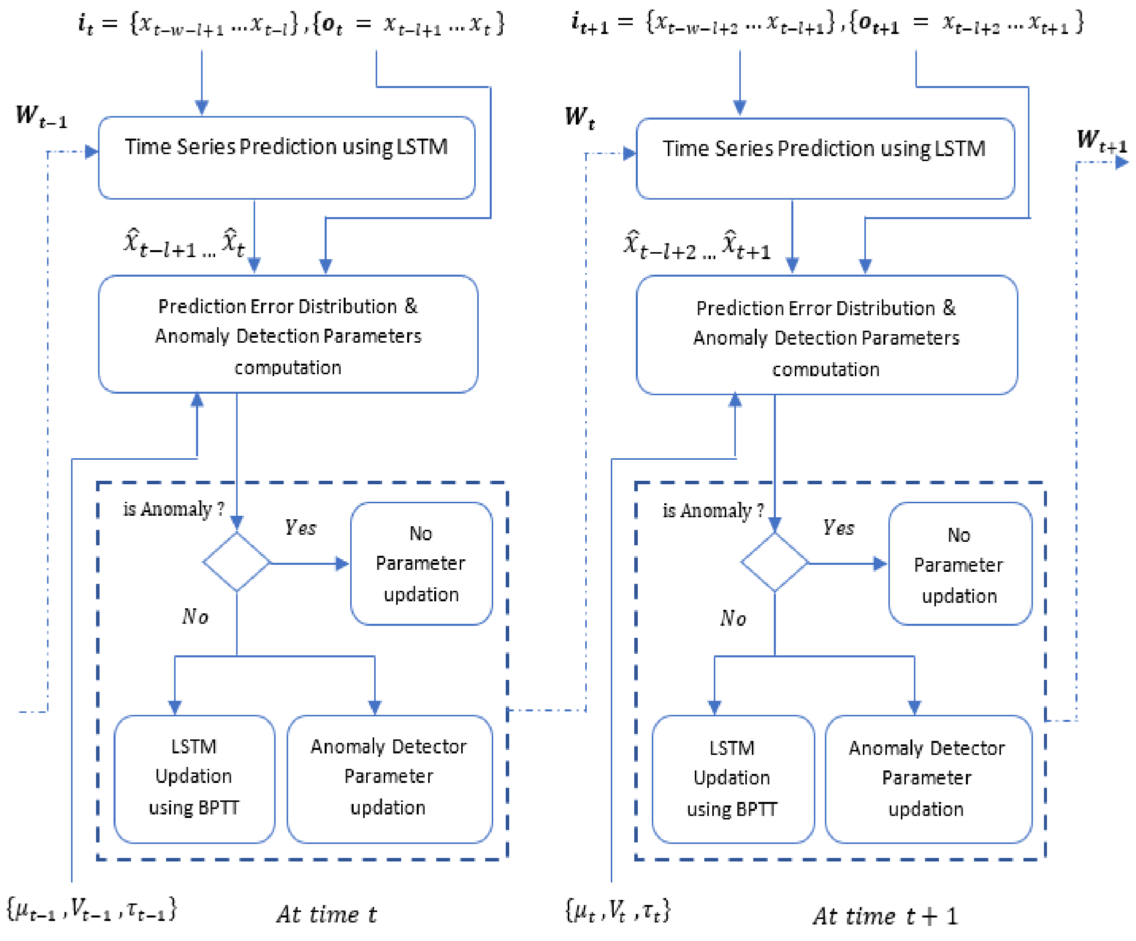

$i_t = \{x_{t-w-l+1} \dots x_{t-l}\}, \{o_t = x_{t-l+1} \dots x_t\}$
$i_{t+1} = \{x_{t-w-l+2} \dots x_{t-l+1}\}, \{o_{t+1} = x_{t-l+2} \dots x_{t+1}\}$
W_{t-1}
W_t
W_{t+1}
Time Series Prediction using LSTM
$\hat{x}_{t-l+1} \dots \hat{x}_t$
$\hat{x}_{t-l+2} \dots \hat{x}_{t+1}$
Prediction Error Distribution & Anomaly Detection Parameters computation
is Anomaly ?
Yes
No
No Parameter updation
LSTM Updation using BPTT
Anomaly Detector Parameter updation
$\{\mu_{t-1}, V_{t-1}, \tau_{t-1}\}$
At time t
$\{\mu_t, V_t, \tau_t\}$
At time $t+1$

Figure 3.1: Steps of Online Anomaly Detection and Change-Point Detection for Proposed Univariate Time Series (OLACD)

Overall, the proposed online anomaly and change-point detection model consists of two modules. The first module, ***Time Series Predictor*** that predicts time stamps for a given prediction horizon and the second module, ***Anomaly Detector*** is responsible for tagging the given time series data points as anomalous or normal.

We next describe our approach and show how multi-step predictions are obtained using LSTMs and then used for anomaly detection and as well as incremental model updation. Overall steps of the proposed model are illustrated in algorithm 3.1. The input and output of the model are portrayed in figure 3.1.

3.2.1. Time Series Predictor

The predictor module of proposed online anomaly and change-point detection model is based on LSTMs. LSTM has been employed in a wide range of streaming applications in a range of different capacities due to its parameter efficiency and primarily because of its capability to automatically discover complex features without having any explicit domain knowledge. This automatic feature learning capability makes the LSTMs a good candidate for online time series anomaly detection. Therefore, proposed model employs LSTMs as multi-step ahead predictor for time series. Like other artificial neural networks, a LSTMs uses training data to adapt and learn its parameters (weights and biases) to perform the anticipated desired task. In proposed online model, the parameters of the network are learned using *BPTT* and optimized using *Adam*. The idea of learning or training of a recurrent neural network is to reduce the error.

In this forecasting module, the cost function computes the difference between the network's predictions and the desired prediction. In the learning process, that difference is minimized during training by adapting the weights and biases of the network. In order to leverage LSTM for forecasting, time series data need to be changed in a compatible format to operate on it. Input time series data are transformed into several sequences of overlapping windows of size w. This window size defines the number of time steps in history, which are taken into account and referred as a *history window*. The number of time steps required to be predicted is referred to as *prediction window* or *prediction horizon* of size l. In some studies, prediction window is also known as *Forecasting Horizon* [122].

Consider a univariate time series $X = \{x_1, x_2, \dots, x_t\}$, where each point $x_i \in R$ in time series represents the data recording at single time instance. At time t, we use a history window of length w as raw input time series i_t:

$$i_t = x_{t-w-l+1} \quad \dots \quad x_{t-l} \tag{3.1}$$

The LSTM model obtained at time $t-1$ with weight parameters W_{t-1} is used to obtain the l steps ahead forecast $\hat{x}_{t-l+1} \dots \hat{x}_t$ corresponding to last l raw observed values o_t :

$$o_t = x_{t-l+1} \quad \dots \quad x_t \tag{3.2}$$

The raw input i_t and raw target output o_t are appropriately scaled in online manner (every data point of input stream is scaled immediately in incremental manner but only when it appears to the model) if needed before feeding it to the latest LSTMs network represented with weight parameters W_{t-1}. The prediction error in forecasting output o_t by processing input i_t using the LSTM weights W_{t-1} are used to compute the anomaly detection parameter $\mu_{t-1}, V_{t-1}, \tau_{t-1}$ and update the LSTM to W_t and anomaly detector parameters to μ_t, V_t, τ_t. We next describe these steps in detail:

A) Online Input Scaling:

Deep learning recurrent neural network models learn a mapping from input and to an output variable. As a result, the scale and distribution of the domain data may change for each variable. Input variables may have distinct units (for example, feet, kilometres, and hours), implying that the variables have different scales.

Differences in scaling among input variables may exacerbate the difficulty of the simulated problem. Substantial input values (for example, a spread of hundreds or thousands of units) can result in a model that learns large weight values. A model with large weight values is frequently unstable, which means it may perform poorly during learning and be sensitive to input values, resulting in larger generalisation error. When employing recurrent neural network models, scaling input variables is crucial.

The input scaling is an important for numerical stability of training the LSTM that standardize the range of data else the time series can take unbounded values. We transform the values of numeric variables of time series so that the transformed data points have specific supportive properties. We are only transforming the range of time series data but not changing the shape of the distribution of data. The scaling of time series is done in an online way; each data point of time series is scaled only when it is required and available for training and anomaly detection. Let suppose x is real time series and y is normalized time series, then linear scaling/normalization of each data point of time series is done in online way using equation 3.3 in such a way that $xMax$ scales to $yMax$ and $xMin$ scales to $yMin$.

$$y = mx + b \tag{3.3}$$

where

$$m = (yMax - yMin)/(xMax - xMin)$$

$$b = \frac{yMax + yMin}{2\, m\, xMax + xMin} = \left[\frac{yMax + yMin}{2(xMax + xMin)}\right]\left[\frac{xMax - xMin}{yMax - yMin}\right]$$

Such that $xMax$ & $xMin$ are maximum and minimum points estimated from original time series; the range possibly consist of only normal data points, while $yMax$ & $yMin$ are desired maximum and minimum range of time series. Each point in the raw input i_t as well as the raw output o_t is transformed using scaling parameter m and b.

B) Multi-Step Ahead Prediction for Time Series:

For univariate time series forecasting there is one linear unit in the input layer and l linear units in the output layer such that there is one unit for each of the l future predictions horizon. The recurrent units in a hidden layer are fully connected through recurrent connections above it through feedforward connections. A sample RNN-LSTM architecture is presented in figure 3.2. The LSTM based RNN can be considered to be a function with weight parameters W_{t-1}

that takes scaled input i_t and provides estimates for output o_t after going through the set of computations in Equation 2.1 with a linear output layer at the top, so as to provide estimates $\hat{o}_t$ corresponding to observed values o_t:

$$\hat{o}_t = f_{LSTM}\,(i_t, W_{t-1}) \tag{3.4}$$

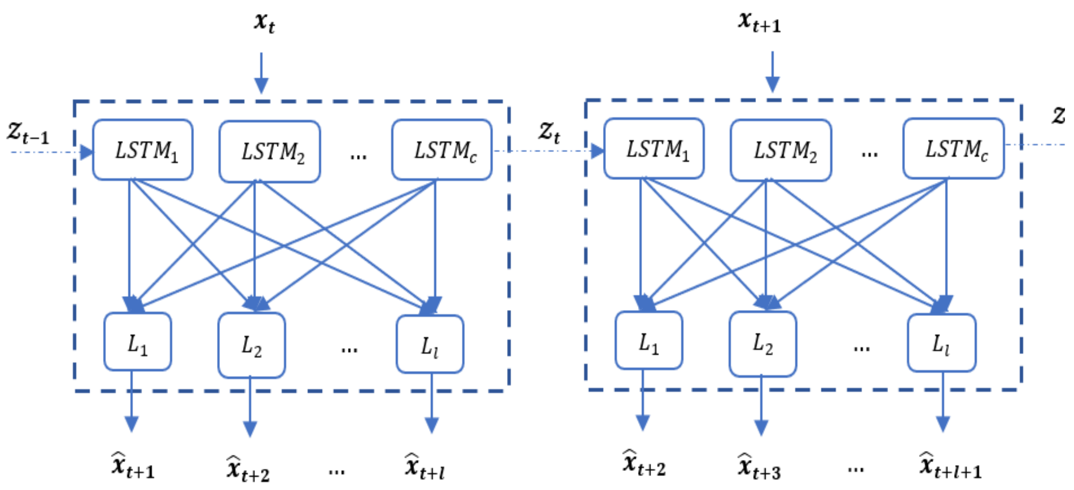


Figure 3.2: A LSTM delineating l-step ahead predictions with one hidden layer with *c* LSTM cells and a linear output layer with l neurons representing with *L*

C) Prediction Error Estimation

Given an estimated value $\hat{x}_t^k$ for x_t^k using equation 3.5, the prediction error vector estimate e_t^k for point x_t is $e_t = [e_t^1, e_t^2, \dots, e_t^l]$, where e_t^k is the error difference between x_t and its predicted value $\hat{x}_t$ at time t for each l prediction where $k = 1,2,..l$. Error for every l prediction is given by equation 3.5:

$$e_t^k = \frac{|x_t^k - \hat{x}_t^k|}{x_t^k} \quad for\ k = t - l + 1 \dots t \tag{3.5}$$

The dimensions of error vector e_t are determined by the prediction horizon l, i.e., *1-d* in case of $l = 1$ which is one step ahead prediction, *5-d* or *10-d* in case of $l = 5$,10 which is 5 or 10 step ahead predictions of time series.

The prediction error e_t in forecasting output o_t by processing input i_t using the LSTM W_{t-1} is used to compute the anomaly detection parameter $\mu_{t-1}, V_{t-1}, \tau_{t-1}$ (mean, covariance and threshold). The p_t is the probability of an error vector e_t after applying Multivariate Gaussian Distribution $\mathcal{N} = \mathcal{N}(\mu\, V)$. When the size of the prediction error e_t is large, the probability p_t of the observed error e_t at time t will be very small, indicating an anomaly. If the system indicate an anomaly then the parameter $W_{t-1}, \mu_{t-1}, V_{t-1}, \tau_{t-1}$ would be restrained while if the system tag x_t as normal at t time stamp than the LSTM parameter W_{t-1}update to W_t and anomaly detector parameters update to μ_t, V_t, τ_t. We next describe the steps followed by anomaly detector after estimation of error vector in detail.

3.2.2 Anomaly Detector

Once the prediction $\hat{x}_t$ at time stamp t is made by the *Time Series Predictor* and error vector e_t is estimated that is passed to this module than this module detects anomalies in a given time series data points in an incremental manner by following some define steps. First, anomaly detector parameters i.e., $\mu_t(mean), V_t(covariance\ ), \tau_t(threshold)$ are computed using e_t. The likelihood p_t of an error vector e_t is given by applying Multivariate Gaussian Distribution $\mathcal{N} = \mathcal{N}(\mu\, V)$. The probability p_t of the observed error e_t at time t will be very small, indicating an anomaly, when the size of the prediction error e_t is large. The threshold τ ,which is fundamental requirement in most of the anomaly detection algorithm, unlike other online anomaly detector method [15, 26], the threshold τ , is not predefined but only initialized automatically by model at once and then updated at every time stamp of time series in an online way along with other statistics, mean μ and covariance V. Overall steps followed by anomaly detector are described in detail:

A) Online Learning of Anomaly Detection Parameter

The prediction error vector e_t estimated from LSTM based time series predictor is modelled to compute the statistics of error (i.e., mean μ , covariance V& threshold τ), these statistics are served as parameters for proposed anomaly detection model. The dimensions of error vector

e_t are determined by the prediction horizon l. For example, if prediction horizon $l = 1$ which is one step ahead prediction, then dimension of e_t is 1 i.e., *1-d*. Whereas if prediction horizon $l = 5$ which is 5 steps ahead prediction of time series in future than dimension of e_t is 5 that is *5-d* and so on. We learned the anomaly detection parameter for one dimension with some specified formulas given in equation 3.6-3.8 that utilize $1 - dimensional\ e_t$. But AD-parameters for multiple dimensions are computed separately with equations 3.9-3.15 that utilize $l - dimensional\ \ e_t$, where l is prediction horizon. The idea of updating the statistics of error for $l - dimensional$ error vector is come from J.L. Shapiro in 2019 [30].

Online Learning of Anomaly Detection Parameters for one step prediction:

Let $e_1, \dots, e_t, \dots$ be a set of real-valued error calculated from LSTM training, with a corresponding time series of size t, the online mean μ and variance σ^2 are be calculated as following:

$$\mu_1 = e_1;$$
$$\sigma_1^2 = 0;$$

$$\mu_{t+1} = \frac{t}{t+1}\mu_t + \frac{1}{t+1}x_{t+1} \tag{3.6}$$

$$\sigma_{t+1}^2 = \frac{t}{t+1}\sigma_t^2 + \frac{(x_{t+1} - \mu_{t+1})^2}{t+1} - (\mu_t - \mu_{t+1})^2 \tag{3.7}$$

$$\tau_{t+1} = \frac{2\,\pi^{1/2}}{\Gamma(1/2)} * [\chi_1^2(\alpha)]^{1/2} * |\sigma_t^2|^{1/2} \tag{3.8}$$

Where $\Gamma(.)$ is the gamma function, $\chi_1^2(\alpha)$ is chi-square probability with 1 degrees of freedom which is equal to number of predictions horizon $l = 1$ while $|\sigma^2|$ is determinant of variance. The value α (*alpha*) is the significance level for chi-square test of independence, which is the probability of rejecting the null hypothesis when it is true. For example, a significance level of 0.01 indicates a 1% chance of concluding that x_t is anomalous at time t and 99% chance of x_t being concluded as normal data point. The value of α is automatically selected by model before starting to detect the anomalies in real streaming and it will than remain fixed for rest of time series enacted for anomaly detection.

On-line learning of AD Parameters for multi-step prediction:

Let $e_1, \ldots, e_k, \ldots$ be an unending sequence of *l*-dimensional real-valued error vectors calculated from LSTM training. Each of the *l* components of the vectors will be denoted with superscript. So, e_k^i is the i^{th} component of the vector X_k .The online mean μ is calculated with following:

$$\mu_{t+1} = \frac{t}{t+1}\mu_t + \frac{\mathbf{e}_{t+1}}{t+1} \tag{3.9}$$

And for all i components,

$$\mu_{t+1}^i = \frac{t}{t+1}\mu_t^i + \frac{e_{t+1}^i}{t+1} \tag{3.10}$$

Covariance $\mathbf{V}$ is calculated as following:

$$\mathbf{V}_t = \frac{1}{t}\sum_{k=1}^{t}(\mathbf{e}_k - \mu_t)(\mathbf{e}_k - \mu_t)^{\mathrm{T}};\ l * l \text{ matrix, with componets} \tag{3.11}$$

$$\mathrm{V}_t^{ij} = \frac{1}{t}\sum_{k=1}^{t}\left(e_k^i - \mu_t^i\right)\left(e_k^j - \mu_t^j\right) \tag{3.12}$$

$$\mathbf{V}_{t+1} = \frac{t}{t+1}\mathbf{V}_t + \frac{(\mathbf{e}_{t+1} - \mu_t)(\mathbf{e}_{t+1} - \mu_t)^{\mathrm{T}}}{t+1} - (\mu_t - \mu_{t+1})(\mu_t - \mu_{t+1})^{\mathrm{T}} \tag{3.13}$$

$(\mathbf{e}_k - \mu_t)^{\mathrm{T}}$ is transpose of $(e_k - \mu_t)$, $(\mathbf{e}_{t+1} - \mu_t)^{\mathrm{T}}$ is transpose of $(\mathbf{e}_{t+1} - \mu_t)$ while $(\mu_t - \mu_{t+1})^{\mathrm{T}}$ is transpose of $(\mu_t - \mu_{t+1})$

For all $i$and$j$ components,

$$\mathrm{V}_{t+1}^{ij} = \frac{t}{t+1}\mathrm{V}_t^{ij} + \frac{\left(e_{t+1}^i - \mu_t^i\right)\left(e_{t+1}^j - \mu_t^j\right)}{t+1} - \left(\mu_t^i - \mu_{t+1}^i\right)\left(\mu_t^j - \mu_{t+1}^j\right) \tag{3.14}$$

Threshold τ is updated using following:

$$\tau_{t+1} = \frac{2\,\pi^{l/2}}{l\,\Gamma(l/2)} * \left[\chi_l^2(\alpha)\right]^{l/2} * |V_t|^{1/2} \tag{3.15}$$

Where l is number of predictions, $\Gamma(.)$ is the gamma function, $\chi_l^2(\alpha)$ is chi-square probability with l degrees of freedom which is equal to number of predictions and $|V|$ is determinant of covariance matrix. The α (*alpha*) is the significance level for chi-square test of independence which is selected automatically by model after sufficient training of time series for some time steps before initiating the anomaly detector module and remain same for rest of stream that encountered for anomaly detection.

B) Initialization of Anomaly Detection Parameters

The assumptions we are using is that the first few hundred data points are supposed to be as normal data points and that does not contain any anomalies and we carefully ensure that for testing purpose by manually looking at the time series data set. We first learn the LSTM prediction model for few hundred incoming normal data point from time series and compute the prediction error for each trained data point in an incremental fashion. After training of LSTM for few hundred normal time series data points we initiate our anomaly detector model by initializing the anomaly detector parameters.

Initialization of Mean:

After training of LSTM for few hundred normal time series data we get an error vector for trained data points. Let $\mathrm{e}_1, \dots, \mathrm{e}_\mathrm{k}$ be a sequence of *l*-dimensional real-valued error vectors calculated from LSTM training. We initiate the *Mean* μ_1 by assigning the last error vector as first vector of *Mean* as in equation 3.16:

$$\mu_1 = e_\mathrm{k}; \tag{3.16}$$

Initialization of Covariance:

Let $e_1, \dots, \mathrm{e}_\mathrm{j}, \dots, \mathrm{e}_\mathrm{k}$ be a sequence of *l*-dimensional real-valued error vectors calculated from LSTM training for few hundred normal time series data points. *d*-dimension correspond to the number of predictions and each of the *l*-components of the vectors will be denoted with superscript. So, x_k^i is the i^{th} component of the vector X_k where $i = 1 \dots l$. So, for initialising the *Covariance Matrix*, we first calculate the standard deviation for each *l*-dimensional real-valued error vectors by skipping the few very initial error vector values using equation 3.17; as in the beginning the error would be high and reduced/converged with the proceeding of LSTM model learning.

$$Std^i = Std.Dev^i(e_j^i, \dots, e_k^i) ; \tag{3.17}$$

The equation gives us vector of standard deviation containing l components; such that one standard deviation is calculated for each l-dimension.

Here if we have l number of predictions we need to initialize the *Covariance Matrix* of size $l * l$. So, we initialize the zero valued $l * l$ *Covariance Matrix* $\boldsymbol{V}_1$ and assign each value from standard deviation vector Std to the diagonal that corresponding to l.

$$\mathbf{V}_1 = \begin{bmatrix} Std^1 & 0 & 0 & 0 \\ 0 & Std^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Std^l \end{bmatrix} ;\ l * l \text{ matrix}$$

Initialization of Threshold:

Considering the equation 3.15, π is constant, Γ is gamma function, l is number of predictions, $\boldsymbol{V}_1$ is first initialized $l * l$ covariance matrix, χ_l^2 is Chai square statistics with l degree of freedom. So, there is only value of α which need to be initialized for the initialization of threshold value. For this purpose we first find the standard deviations for each l-dimensional real-valued error vectors given as $Std = [Std^1 ,\dots, Std^l]$ by skipping the few very initial error vector values as per equation 3.17, and taken the mean of that standard deviations, i.e. $mean(Std)$ which gives only one value of standard deviation. The reason for skipping the few very initial error vectors is that, at start of LSTM training and for very few trainings of time series data points the value of error would be high as and we need to wait for sufficient training to be procced until the error vectors would be stabilised around its convergence to nearly zero. Afterward, α *(alpha)* is decided as per equation 3.18 and threshold τ is initialized using equation 3.19 we initialize the threshold, the formula for initialisation and updation of τ is depicted and derived from [123].

$$\alpha = [mean(Std\,)]^{l} \tag{3.18}$$

$$\tau_1 = \frac{2\,\pi^{l/2}}{l\,\Gamma(l/2)} * \,[\chi_l^2(\alpha)]^{l/2} * \,|V_1|^{1/2} \tag{3.19}$$

C) Anomaly Detection Using Prediction Error Distribution

The assumptions we are using is that we first learn the prediction model for each incoming data point from time series using LSTMs in an online way, and then estimate the error vector that pass on to anomaly detector module to compute the prediction error distribution for that trained data point. For the incremental prediction error distribution for every data point of time series at each time stamp, the online estimation of mean, covariance and threshold should be done using their respective recursive set of equations 3.6-3.15, which is then used to detect the anomaly. We are assuming that the initial standard deviation is selected from the normal error signal and that is the maximum value of normal range of error signal that achieve after some sufficient LSTM model learning i.e., time series prediction model learns in continuous regime for few streaming data points until the error value become stable.

Let suppose l predictions are predicted for the input x_t, the error vector e_t for data point x_t is $e_t^l = [e_t^1\,, \dots, e_t^l]$. The Multivariate Gaussian Distribution $MV\mathcal{N}(\mu\,V)$ is fitted to the error vector, where μ is mean and V is covariance matrix.

$$p_t = \frac{1}{\sqrt{|V\,|(2\pi)^d}} exp^{-\frac{1}{2}(e_t-\mu)'\boldsymbol{V^{-1}}\,(e_t-\mu)} \tag{3.20}$$

The p_t is the probability of an error vector after applying Multivariate Gaussian Distribution using equation 3.20. An observation x_t is classified as '*anomalous'* if $p_t < \tau_t$ at time stamp t, else the observation is classified as '*normal*', where anomalous points belong to positive class and normal points belong to negative class).

D) Updating the Model Parameters Updation

Once the observation x_t is decided and tagged as *anomalous*, none of parameter including LSTM and anomaly detector are updated but restrained such that: $W_t = W_{t-1}$, $\mu_t = \mu_{t-1}$, $V_t = V_{t-1}$, $\tau_t = \tau_{t-1}$. Whereas if observation x_t at time t is decided and tagged as *normal* than then LSTM parameters (weights W) are updated using BPTT and anomaly detection parameters (mean μ , covariance V& threshold τ) are updated by values of parameter computed using 3.6-3.15. The updation of model's parameters is entirely depends on *normal tagging* of observation x_t, and the reason for not updating the model's parameter is that we are not intended to learn the anomaly by LSTM model but only wants to detect the anomaly and adapt the changes either transitory or permanent in data distribution.

It is to be noted that the $MV\mathcal{N}$-probability estimation of error controls the update of model parameters as when the size of the prediction error is large, the probability of the observed error will be very small, indicating an anomaly. Since the probability and threshold are computed for prediction horizon of l steps, a point anomaly or a very short-term anomaly will lead to very small probability estimated for observing large error only for a very short period of time. This ensures that no significant updates to the model parameters are made if probability is small that indicating an anomaly. On the other hand, a small probability estimation over a period of time due to change points leads to small probability estimation for a significantly large period of time leading to continuous changes and performing no updation. In this case we introduce a *change detection counter*, if the model continuously estimates small probabilities, then the counter value decides and consider it as change point and start updating till the model parameters are adapted to predict the new normal behaviour well and the estimated probabilities increases accordingly. It is important that the *change detection counter* must be at least greater than twice of prediction horizon l ; such that $change\ detection\ counter > 2 * l$. All the above-described steps for online anomaly change detection model are summarised and demonstrates in algorithm 3.1.

```
Algorithm 3.1: Online Anomaly Detection and Change-point Detection
Input: Single/Multi-step Predictions of LSTM for data instances from Time Series
Output: Data instance classify as normal or anomalous

Begin
Step 1: Set burn-in period and architecture of LSTM network such as input, hidden and output
        units. Where $l$ is equal to number of output units.
Step 2: Perform algorithms 2.2 for LSTM from Step 1-9
Step 2: $l$-dimensional Predictions of LSTM given to the AD model after burn-in period
Step 3: $l$-dimensional prediction error estimation after LSTM burn-in period using equation 3.5 for
        observation $x_t$
Step 4: Initialise AD parameters and $\alpha$ (alpha): $\mu_1$, $V_1$, $\tau_1$ after LSTM burn-in period using
        equations 3.16, 3.17, 3.18 and 3.19
Step 5: Learning of AD parameters at time step $t$
        For one-step prediction learn $\mu_t$, $\sigma_t^2$, $\tau_t$ using equations 3.6, 3.7 and 3.8
        For multi-step prediction learn $\mu_t$, $V_t$, $\tau_t$ using equations 3.9-3.15
Step 6: Multivariate gaussian distribution $MV\mathcal{N}(\mu\ V)$ is fitted to the $l$-dimensional prediction
        error vector, where $p_t$ is the probability of error vector
Step 7: if
        the observation $x_t$ is classified as anomalous if $p_t < \tau_t$ at time step t
        else
        the observation $x_t$ is classified as normal
Step 8: if
        $x_t$ is classified as anomalous.
        No updates of all parameters at time $t$ such that
        $W_t = W_{t-1}$ .   $\mu_t = \mu_{t-1}$, $V_t = V_{t-1}, \tau_t = \tau_{t-1}$
        else
        update AD parameter learned at current time step $t$ such that
        update all LSTM parameters using equations 2.16-2.48
End
```

3.3 Performance Evaluation Metrices for OLACD

The anomaly detection can be considered as a special case of binary classification problem on data with extremely unbalanced classes [16]. Therefore, the well-known metrics based on the confusion matrices are usable also in the context of anomaly detection and change-point detection.

Before discussing the specific performance metrics, the definition of the confusion matrix that is used in this thesis is presented. The confusion matrix is 2 × 2 matrix in the following form:

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}$$

where *TP* (*True Positives*) is number of anomalous data points correctly labelled by the detector, *FP* (*False Positives*) is the number of normal data points incorrectly labelled as anomalous, *FN* (*False Negatives*) is the number of anomalous data points incorrectly labelled as normal and *TN* (*True Negatives*) is the number of normal data points correctly labelled as normal.

Some of the useful performance metrics that we can employ to evaluate anomaly and change detection algorithms are summarized below. While these are described in the context of binary classification, they can each be extended to classification of a greater number of classes by providing the measures for each class independently or in combination [9, 10, 124].

Accuracy: It calculated as the ratio of correctly classified data points to total data points. This measure provides a high-level idea about the algorithm's performance [9, 10, 124].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Given the huge, expected imbalance between classes (anomalous and normal data points), some popular classification metrics such as *Accuracy* is irrelevant for anomaly detection. The companion to accuracy is Error Rate, which is computed as 1 - Accuracy. Accuracy and Error Rate do not provide insights on the source of the error or the distribution of error among the different classes. Even the trivial anomaly detector (null detector) where every data point labelled as normal (TP = 0) would have very high accuracy

$$Accuracy_{null} = \frac{TN}{TN + FN}$$

Because the number of anomalous instances is expected to be considerably lower than number of normal instances and therefore $TN + FN \;\cong\; TN \;\Rightarrow Accuracy_{null} \cong 1$. On the other side, the *true positive rate* (or *recall* or *sensitivity*) and *false positive rate* are very suitable metrics because they describe properties that are certainly expected from a good anomaly detection system:

The Accuracy is considered as ineffective for evaluating performance in a class-imbalanced dataset, which is typical for anomalies and change point detection, because they consider different types of classification errors as equally important. Sensitivity and G-mean are useful metrics to utilize in this case.

Sensitivity: Which also referred to as Recall or the true positive rate (TPR). This refers to the portion of an anomalous class of interest (Anomalies and Change Points) that was recognized correctly.

$$Senstivity = Recall = TPR = \frac{TP}{TP + FN}$$

Specificity: Which also referred to as the true negative rate (TNR). This refers to the portion of a normal class of interest that was recognized correctly.

$$Specificity = TNR = \frac{TN}{TN + FP}$$

G-mean: Anomaly and change point detection typically results in a learning problem with an imbalanced class distribution because the ratio of changes to total data is small. As a result, G-mean is commonly used as an indicator of model performance. This utilizes both Sensitivity and Specificity measures to assess the performance of the algorithm both in terms of the ratio of positive accuracy (Sensitivity) and the ratio of negative accuracy (Specificity) [9, 10, 124].

$$G-mean = \sqrt{Sensitivity \times Specificity}$$

Precision: This is calculated as the ratio of true positive data points (anomalies and change points) to total points classified as anomalous[9, 10, 124].

$$Precison = \frac{TP}{TP + FP}$$

F-score: This measure provides a way to combine Precision and Recall as a measure of the overall effectiveness of a anomaly detection and change-point detection algorithm. F-measure is calculated as a ratio of the weighted importance of Precision and Recall[9, 10, 124].

$$F-score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Receiver Operating Characteristics Curve (ROC): ROC-based assessment facilitates explicit analysis of the trade-off between true positive and false positive rates. This is done by plotting a two-dimensional graph with the false positive rate on the x axis and the true positive rates on the y axis. An anomaly detection and change-point detection algorithm produce a (TPR, FPR) pair that corresponds to a single point in the ROC space. One algorithm can generally be considered as superior to another if its point is closer to the (0,1) coordinate (the upper left corner) than the other. To assess the overall performance of an algorithm, we can look at the Area Under the ROC curve, or AUC [5, 9, 10, 20, 124].

Area Under Curve (AUC): Most anomaly detectors require definition of some threshold value [16] (here α (*alpha*) value for significance level for chi-square test is considered to calculate the threshold) that significantly influences the resulting true positive rate and false positive rate. However, the impact on the TPR and FPR is exactly the opposite. Two edge cases are recognized: α value that causes all data points to be labelled as anomalous $(\text{TPR} = 1,$ high FPR) and the opposite value that causes all data points to be labelled as normal $(\text{TPR} = 0,\ \text{FPR} = 0)$; given that the used dataset contains at least one normal and one anomalous data point. Important information about the properties of an anomaly detection system can be gained by observing the changes in its behaviour (relation between TPR and FPR) while modifying the α value by binary search. This observation can be formalized by plotting the *receiver operating curve (ROC)* and calculating the *area under curve (AUC).* The ROC can be obtained by simply plotting the 2-D points with coordinates given by $x \to FPR, y \to TPR$ for various α values and applying the trapezoidal rule. This definition implies that anomaly and change detectors with higher AUC are better (intuitively, higher TPR is gained with lower increase in FPR) [5, 9, 10, 20, 123.

3.4 Chapter Summary

We have introduced the online anomaly detection and change-point detection for univariate time series in section 3.1. We proposed a novel online anomaly detection method using LSTM. Our novel approach is divided into two modules: First, time series prediction using LSTM and second, anomaly detector. Both modules of proposed approach are discussed in detail in section 3.3. Every anomaly detection system needs some evaluation metrices to check the validity and efficiency of respective model, we are used TPR, FPR and AUC that discussed in section 3.3. In next chapters we demonstrate the experimental results based on proposed technique and will compare the results in term of AUC with one of existing online anomaly detection model. To ensure the validity and efficiency of our proposed novel method we use same the time series data set utilized by existing online method in literature with that we are comparing our experimental results.

Chapter 4

Online Point Anomaly Detection for Univariate Time Series

The approach for anomaly and change-point detection which was developed in Chapter 3 is presented for verified on *Transitory Anomalies* in the presented chapter. The section 4.1 gives introduction to the chapter. Data sets description will be given in section 4.2, followed by the experimental setup in 4.3. Experimental results for Online Anomaly and Change-Point Detection (OLACD) model are demonstrated in section 4.4 that followed by the discussion and conclusion in 4.5.

4.1 Introduction

This chapter presented the experimental design and implementation of proposed online novel anomaly detection and change-point detection technique that utilised to experimentally validate the model for univariate time series and targeted the transitory/short-term anomalies. The main objective of the presented section of research report is to investigate and hence understand and start to build a comprehensive knowledge for experimental design and implementation for proposed novel online anomaly detection model specifically for short-term anomaly detection. As described in chapter 3 the novel online anomaly and change-point detection model first learn the *prediction model* for each incoming data point from time series $\{x_1, x_2, \ldots, x_t\}$, using LSTMs in an online way and compute the error vectors. Second, the *anomaly detector* that estimate of the error vectors is pass on to this to compute the prediction error distribution for each trained data point. For the incremental prediction error distribution for every data point of time series at each time stamp t, the online estimation of mean (μ_t), covariance (V_t) and threshold (τ_t) should be done using their respective recursive set of equations 3.6-3.15, which is then used to detect the anomaly. The error vector e_t are modelled to fit a Multivariate

Gaussian distribution $MV\mathcal{N}(\mu_t\ V_t)$ to handle the error vector of size l (the prediction horizon) generated by the LSTM learning model trained for multi-step predictions. The likelihood p_t of observing an error vector e_t is given by the value of $MV\mathcal{N}$ at e_t. The dimension of error vector is determined by the prediction horizon. An observation x_t is classified as '*anomalous*' if $p_t < \tau_t$ at time stamp t, else the observation is classified as '*normal*', where anomalous points belong to positive class and normal points belong to negative class. Unlike other online anomaly detection method [15] our proposed model is not relying on user pre-defined parameter such as threshold τ, rather it adapts the threshold and updated with each time step in an incremental manner.

The series of experiments, which enabled the author to comprehend proposed novel online algorithms for unsupervised training and anomaly detection of time series in an online way. One of our goals and indeed a requirement of experiments are to develop the necessary mathematical knowledge to enable anomaly detection (AD) in an online manner for continuous stream of data. Once the required online anomaly detection model has been appropriately working for one-step ahead prediction, we extended the experiments to multi-step ahead predictions; as recently the LSTMs has become a norm for designing and implementing various computing machines [115] that efficiently handles the spatial and temporal dependencies in streaming data. The proposed AD model is then experimentally validated, and its results are comparing with the existing anomaly detection techniques.

We validate the efficiency of the proposed novel approach for point anomalies of univariate time series via experiments on publicly available and proprietary two real-world and one synthetic benchmark datasets [4] and compare the results in term of Area Under Curve (AUC) with state of-the-art online anomaly detection algorithm available in literature and the research article was publicised by Saurav *et. al.* in 2018 [5] with title 'Online Anomaly Detection with Concept Drift Adaptation using Recurrent Neural Networks' in 2018. Saurav *et. al.* use Gated Recurrent Unit (GRU), a modern RNN for incremental training that generate the error and then model that error to compute the *Anomaly Score* and afterward use the anomaly score to decide the anomalous behaviour of time series. Furthermore, the technique is utilizing local normalization for data pre-processing before presenting the incoming streaming data to anomaly detection model.

We consider several variants of non-stationary real-valued univariate time series where statistical components (i.e., mean or the standard deviation) of the time series values over a window change with time. In this chapter we are analysing the proposed online anomaly detection system performance for transitory changes i.e., point anomalies. These changes may happen suddenly, or continuously. For the purpose of comparing the model evaluation results in term of AUC we consider the same data set that are used in existed research [5]. We first describe the datasets considered for experimental evaluation, then describe anomaly detection model setup and hyperparameters selection process followed by results and observations.

4.2 Data Sets Description

We consider two real-world and one synthetic univariate time series which are taken from Yahoo Webscope3 [27] The datasets are considered to cover at least one of the scenarios as identified in for example point anomalies or outliers. Yahoo Webscope3 is anomaly detection benchmark which is a publicly available data set released by Yahoo Labs consists of 367 real and synthetic time series with point anomaly labels. Each time series consists of 1420 - 1680 instances. The benchmark is further divided into four sub-benchmarks namely *A*1 *Benchmark*, *A*2 *Benchmark*, *A*3 *Benchmark*, and *A*4 *Benchmark*. *A*1 *Benchmark* contains real Yahoo membership login data, which tracks the aggregate status of logins on Yahoo network [27], while, other three sub-benchmarks comprise of synthetic data. *A*2*Benchmark* and *A*3*Benchmark* contain only outliers or point anomalies which are present on random positions, while *A*4*Benchmark* also contains change-point anomalies. In each data file, there is a Boolean attribute '*label*' indicating if the value at the particular time stamp is tagged as anomalous or normal. In addition to value and label, *A*3*Benchmark* and *A*4*Benchmark* contain the additional attributes such as change-point, noise, trend, and seasonality. However, we are discarding all the additional fields and only using *value* attribute for all the experiments excepting for testing. The main reason for selecting this data set for evaluation is the availability of the point anomalies labels, which are not commonly available in publicly available streaming data sets.

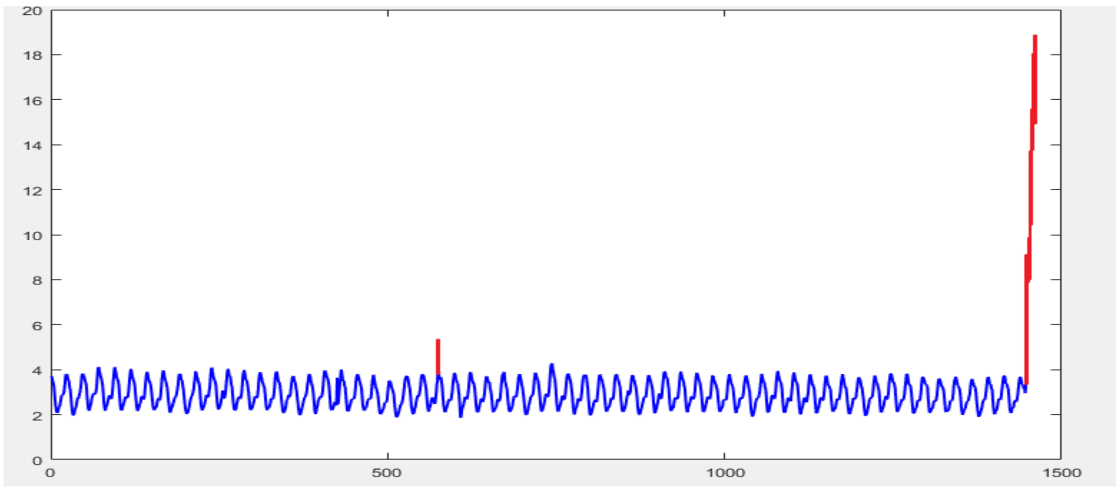
Secondly, a publicised online anomaly detection approach used same data set and shown their results in term of AUC, and we will compare our results with that state-of-the art available online algorithm for anomaly detection [5].

Let the input at time t in the LSTM cell be x_t, the cell state from time $t-1$ and t be c_{t-1} and c_t and the output for time $t-1$ and t be h_{t-1} and h_t . The initial value of c_t and h_t at t will be zero.

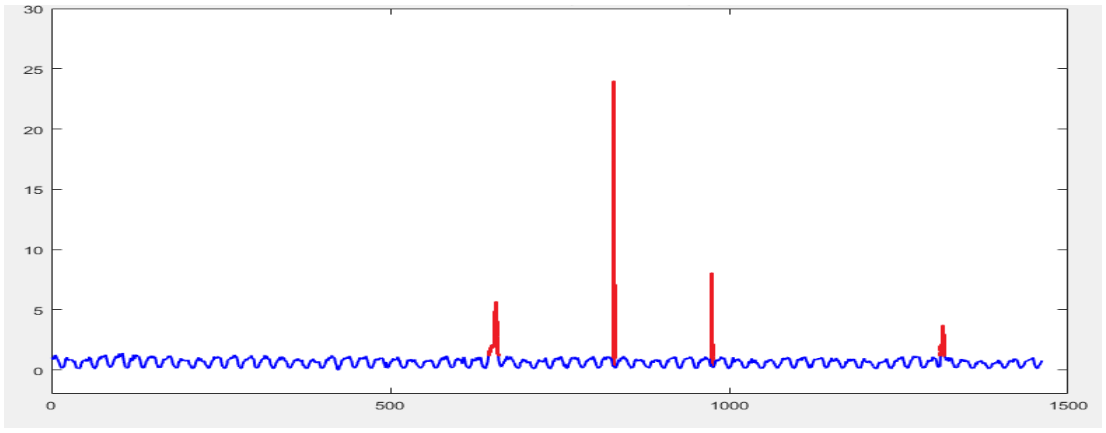
Description of all data sets that used for transitory anomaly detection are graphically demonstrated in figure 4.1 and summarised in table 4.1. Figure 4.1(a) shows Yahoo A1Benchmark/real 3.csv which is periodic time series having 1461 observation including 15 anomalies at 2 different locations. 4.1(b) is Yahoo A1 Benchmark/real 9.csv that is periodic time series having 1461 observations that includes 8 anomalies at 4 different locations and figure 4.1(c) is Yahoo A2 Benchmark/synthetic 13.csv which is periodic continuous time series having 4 anomalies at 2 different positions with 1421 total observations. Figure 4.2(b) shows the scaling for Yahoo A2 Benchmark/synthetic 13.csv times series. Blue region in sub-plots of figures 4.1 and 4.2 shows the normal time series recordings while red colour represents the anomalous region. Normal data points are labelled with zeros while all anomalous points have labelled with ones.

TABLE 4.1: SUMMARY OF ALL DATA SETS USED FOR TRANSITORY ANOMALY DETECTION

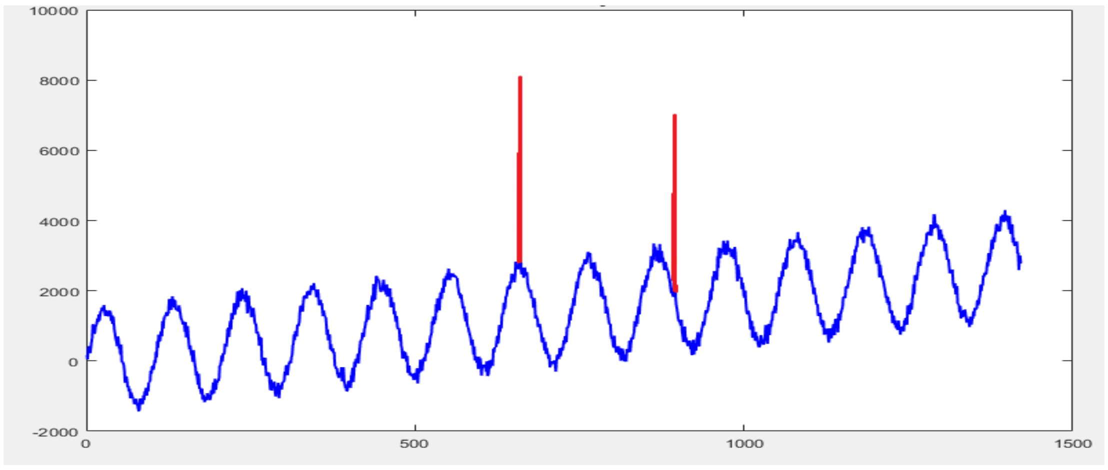
UNIVARIATE INPUT TIME SERIES	TOTAL OBSERVATIONS	NORMAL OBSERVATIONS	ANOMALOUS OBSERVATIONS
Yahoo A1Benchmark/real 3	1461	1446	15
Yahoo A1Benchmark/real 9	1461	1453	8
Yahoo A2Benchmark/synthetic 13	1421	1417	4



(a) Yahoo A1Benchmark/real 3



(b) Yahoo A1Benchmark/real 9



(c) Yahoo A2Benchmark/synthetic 13

Figure 4.1: Dataset/Time Series description used for online anomaly detection using LSTM.

4.1(a) Yahoo A1 Benchmark/real 3.csv is periodic time series having 15 anomalies at 2 different points 4.1 (b) Yahoo A1 Benchmark/real 9.csv is periodic time series having 8 anomalies at 4 different locations, 4.1(c) Yahoo A2 Benchmark/synthetic 13.csv is periodic continuous time series 4 anomalies at 2 different positions. Blue colour in all plots shows the normal time series recordings and red colour describe the anomalous recordings.

4.3 Experimental Setup

For each observed dataset, we train the initial RNN model in an incremental fashion (continuously updating the network parameters at each time step) over first 30% of time steps. For LSTM based model, the network with 1 hidden layer and various number of LSTMs in hidden layer is used for generating the prediction error. We used learning rate η = 0.01 for all cases. Further, we experimented with prediction horizon of l = 1, 5, 10. We are utilizing grid search to find the best architecture such as number of LSTM nodes in a hidden layer. We choose fixed RNN *history window* (h_w) of length 100 for all experiments.

Our online idea of using statistics of error computed by RNN is fetched from the offline and supervised techniques proposed by Chauhan *et al.* in 2015 [12] where they suppose a time series $X = \{x_1, x_2, \dots, x_t\}$, where x_t is arriving at time t and make one/multiple prediction and calculate an error vector e_t at time t for $x_{t-1}$from normal training set using RNN. The error vector e_t is supposed to be one dimensional in case of one step-a-head predictions and $l-dimentional$ version case of $l\ step\ a\ head$ prediction for point x_{t-1}. Afterward Multivariate Gaussian Distribution fitted to the error vectors on the normal validation set. p_t is the probability of an error vector e_t after applying Multivariate Gaussian Distribution $\mathcal{N} = \mathcal{N}(\mu\ V)$ and used the Maximum Likelihood Estimation to select the parameters μ and V from normal validation set; and used validation set that contain both normal and anomalous values to learn the threshold τ by maximizing $f - score$. All the observations in their test data are classified as '*anomalous*' if $p < \tau$, else classified as '*normal*'. The anomaly prediction model proposed by Chauhan *et al.* [12] is work purely in an offline manner and supervised controlled environment thus, not appropriate for online prediction and anomaly detection of streaming data.

Unlike the off-line techniques published by [8, 26] that model the error to detect anomalies which searched the best threshold for the labelled validation data sets, which allows them to set a threshold to separate normal data from anomalies. Rather, we are doing unsupervised learning and get an automatic initialization of threshold τ on every single time series straight after burning period of LSTM training for 30% data points of series. Once sufficient training is done for considered time series than proposed model starts the anomaly detection and update the threshold in such an online way that it will adapt the changes in time series. Generally, each time series has its own characteristics, and finding a generic value of significance level α (alpha) which works for all of the time series is not a straightforward task and may not work properly. Therefore, to keep every process in-line with the online methodology, the value of significance level α (alpha) is also automatically selected by the model which is described in section 3.2.2(B) of earlier chapter.

4.4 Experimental Results for Transitory Anomalies using Online Anomaly and Change Detection Model

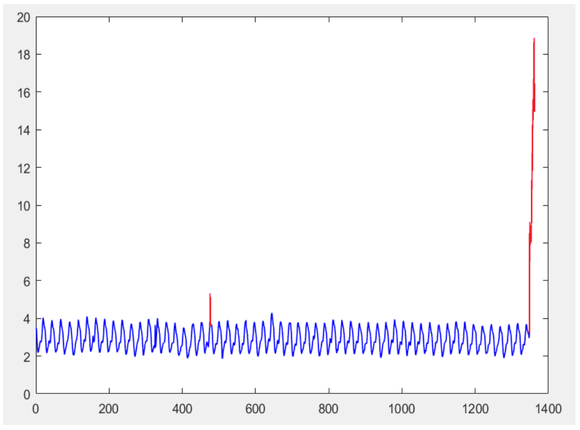
This section gives detailed results set produced by each experiment and provides analysis and conclusion directly related to the aggregation of analytics of each experiment. The results of implemented experiments for online proposed methodology are explored for analysis; thus, evaluated in order to produce an evaluation conclusion (4.5). The performance evaluation metrics described in chapter 3 are adopted for the experimental validation that aim to prove the effectiveness and efficiency of *online anomaly detection* algorithm.

4.4.1 Online LSTM Training and Prediction Error

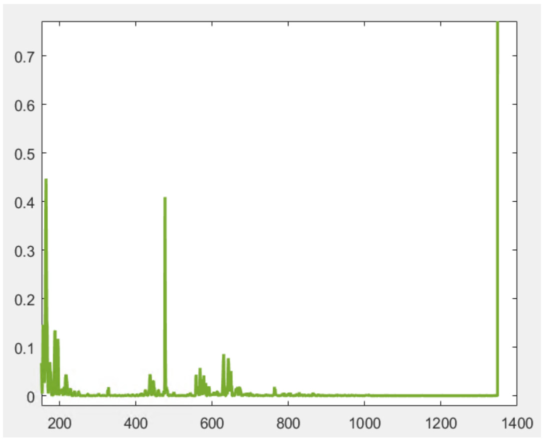
Considering the experimental setup discussed in section 4.3 and the parameter setting given the table 4.1. Figure 4.3 (b), 4.4 (b) and 4.5 (b) shows the observational values of time series. Figure 4.3 (c), 4.4 (c) and 4.5 (c) shows the prediction error for 1 step-ahead predictions for Yahoo A1Benchmark/real 3, Yahoo A1Benchmark/real 9 and for Yahoo A2Benchmark/synthetic 13 respectively. The figure 4.3 (d), 4.4 (d) and 4.5 (d) shows the prediction for 5 step-ahead predictions for Yahoo A1Benchmark/real 3, Yahoo A1Benchmark/real 9 and for Yahoo A2Benchmark/synthetic 13 respectively while figure 4.3 (e), 4.4 (e) and 4.5 (e) give the prediction error for 10 step-ahead predictions for Yahoo A1Benchmark/real 3, Yahoo A1Benchmark/real 9 and for Yahoo A2Benchmark/synthetic 13 respectively. The figures 4.3(c-e), 4.4(c-e) 4.5(c-e) shows that there is higher prediction error for anomalous region of time series as compare with the normal regions and this predictions error is further used in anomaly detection model in incremental manner.

Normal Time Series
Anomalous Time Series
Error for 1 Step-a-head Prediction
Error for 5 Step-a-head Predictions
Error for 10 Step-a-head Predictions

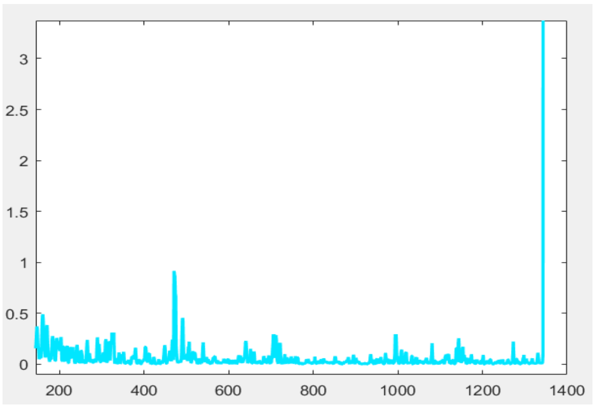
(a) Legends of subfigures (b)-(e)



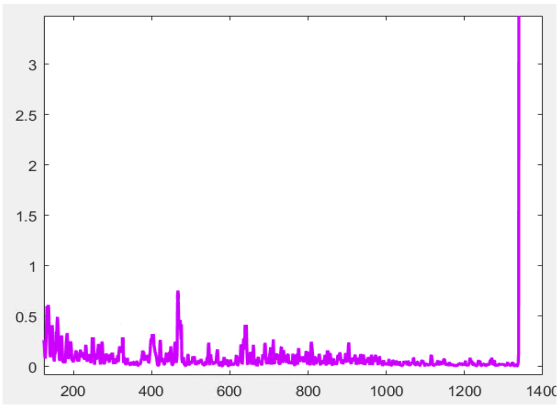
(b) Yahoo A1Benchmark/real 3



(c) 1 Step-a-head prediction error for Yahoo A1Benchmark/real 3



(d) 5 Step-a-head prediction error for Yahoo A1Benchmark/real 3



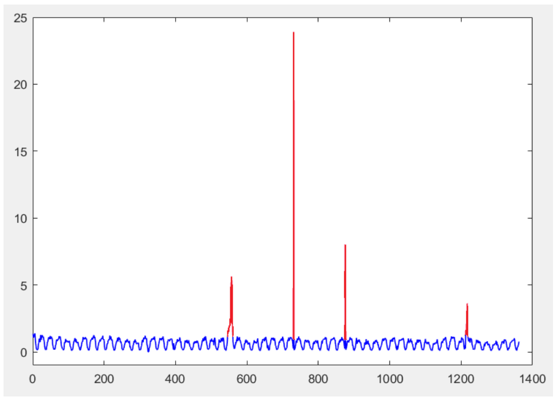
(e) 10 Step-a-head prediction error for Yahoo A1Benchmark/real 3

Figure 4.2: Results for Online LSTM Prediction for A1Benchmark/real 3 datasets.

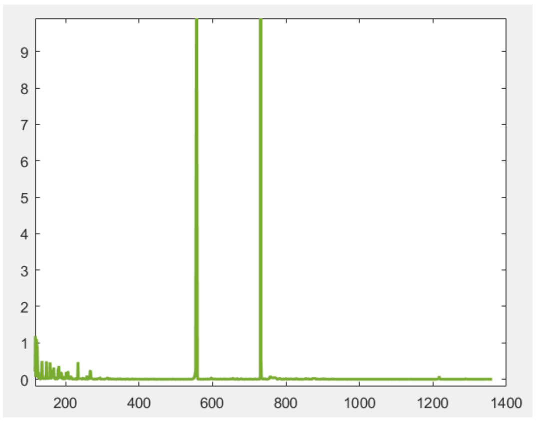
4.3 (a) give the legends of subfigures. In (b)-(e) time steps are given on x-axis. In (b)-(e) y-axis shows the observational values. Figure 4.3(a) present the time series, 4.3 (b) demonstrates the prediction error for 1-step ahead and shown in green colour, figure 4.3(c), shows the prediction error for 5-step ahead prediction and presented in cyan colour while 4.3(e) represents the prediction error for 10-step ahead which shown in magenta colour. Respected prediction error values are given in (b)-(e). Red colour in figure 4.3(a) represents the anomalous points in data stream and blue colures represents the normal points in time series. There are total fifteen anomalous points at two different locations for Yahoo A1Benchmark/real 3 data set, there is one anomalous point happen between 400 and 600, while there are set of 14 consecutive anomalous points occur between 1200 and 1400 and shown in sub-figure (b). Data instances of time series are given on x-axis, while prediction error values are given on y-axis in figure (c-e). The high peaks of respected *l* step-a-head prediction error in (c)-(e) exhibit the presence of anomalies/outlier in data stream given in (b).

Normal Time Series
Anomalous Time Series
Error for 1 Step-a-head Prediction
Error for 5 Step-a-head Predictions
Error for 10 Step-a-head Predictions

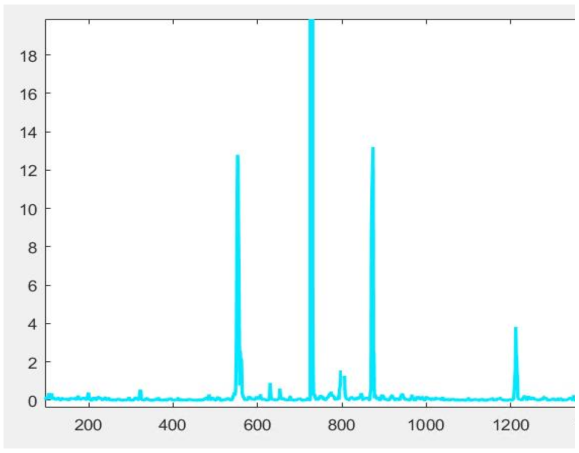
(a) Legends of subfigures (b)-(e)



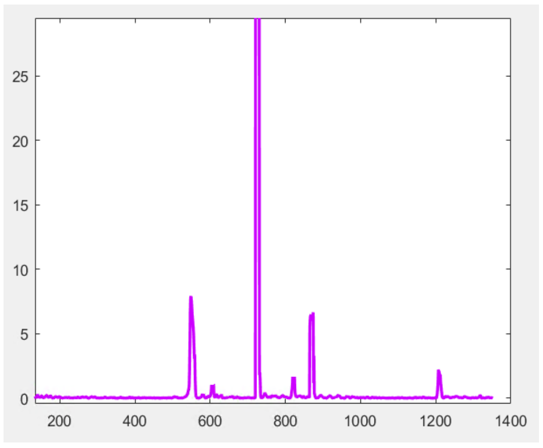
(b) Yahoo A1Benchmark/real 9



(c) 1 Step-a-head prediction error for Yahoo A1Benchmark/real 9



(d) 5 Step-a-head prediction error for Yahoo A1Benchmark/real 9



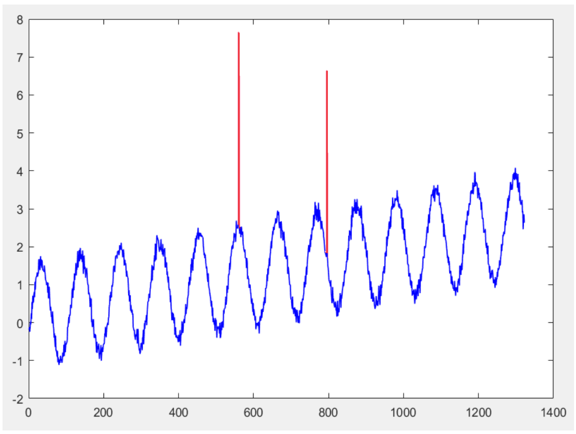
(e) 10 Step-a-head prediction error for Yahoo A1Benchmark/real 9

Figure 4.3: Results for Online LSTM Prediction for A1Benchmark/real 9 datasets.

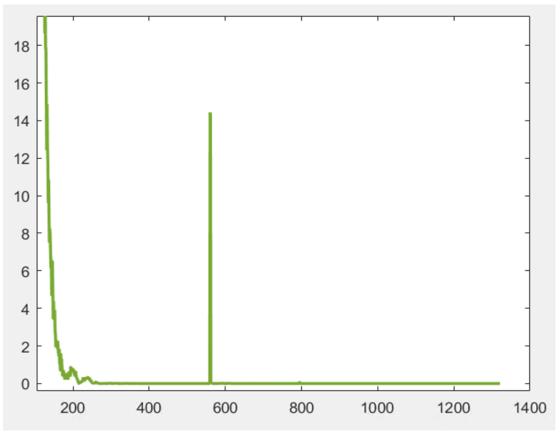
4.4(a) give the legends of subfigures. In (b)-(e) time steps are given on x-axis. In (b)-(e) y-axis shows the observational values. Figure 4.4(a) present the time series, 4.4 (b) demonstrates the prediction error for 1-step ahead and shown in green colour, figure 4.4(c), shows the prediction error for 5-step ahead prediction and presented in cyan colour while 4.4(e) represents the prediction error for 10-step ahead which shown in magenta colour. Respected prediction error values are given in (b)-(e). Peaks in (b)-(e) shows the higher values exhibit the presence of anomalies/outliers. Red colour in figure 4.4(a) represents the anomalous points in data stream and blue colures represents the normal points in time series. There are total eight anomalous points at four different locations for Yahoo A1Benchmark/real 9 data set, there are four anomalous point happen between 400 and 600, one anomalous points between 700-800, one outlier between 800-900 while there are two anomalous points occur between 1200 and 1300 and shown in sub-figure 4.4 (b). Data instances of time series are given on x-axis, while prediction error values are given on y-axis in figure (c-e). The high peaks of respected *l* step-a-head prediction error in 4.4(c)-(e) demonstrate the presence of anomalies/outlier in data stream given in 4.4(b).

Normal Time Series
Anomalous Time Series
Error for 1 Step-a-head Prediction
Error for 5 Step-a-head Predictions
Error for 10 Step-a-head Predictions

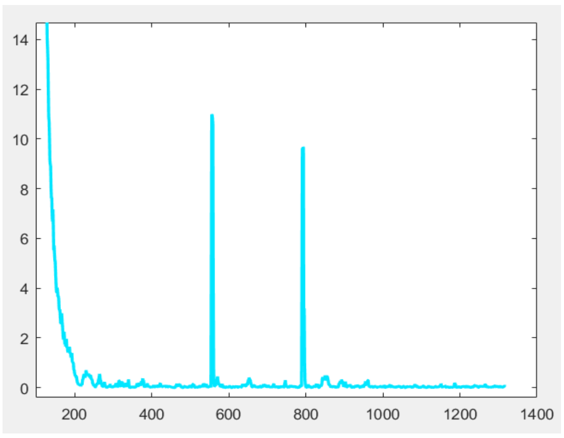
(a) Legends of subfigures (b)-(e)



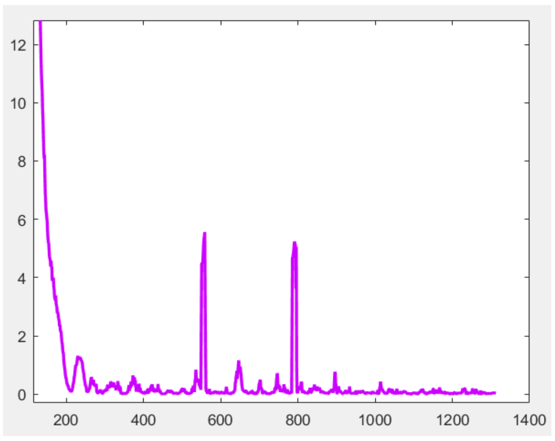
(b) Yahoo A2Benchmark/synthetic 13



(c) 1 Step-a-head prediction error for Yahoo A2Benchmark/synthetic 13



(d) 5 Step-a-head prediction error for Yahoo A2Benchmark/synthetic 13



(e) 10 Step-a-head prediction error for Yahoo A2Benchmark/synthetic 13

Figure 4.4: Results for Online LSTM Prediction for Yahoo A2Benchmark/synthetic 13 datasets.

4.5 (a) give the legends of subfigures. In (b)-(e) time steps are given on x-axis. In (b)-(e) y-axis shows the observational values. Figure 4.5(a) present the time series, 4.5(b) demonstrates the prediction error for 1-step ahead and shown in green colour, figure 4.5(c), shows the prediction error for 5-step ahead prediction and presented in cyan colour while 4.5(e) represents the prediction error for 10-step ahead which shown in magenta colour. Respected prediction error values are given in (b)-(e). Peaks in (b)-(e) shows the higher values exhibit the presence of anomalies/outliers. Red colour in figure 4.5(a) represents the anomalous points in data stream and blue colures represents the normal points in time series. There are total four anomalous points at two different locations for Yahoo A2Benchmark/synthetic 13 dataset, there are two anomalous point happen between 500 and 600, and two anomalous points between 750-850 and shown in sub-figure 4.5(b). Data instances of time series are given on x-axis, while prediction error values are given on y-axis in figure (c-e). The high peaks of respected *l* step-a-head prediction error in 4.5 (c)-(e) show the presence of anomalies/outlier in data stream given in 4.5(b).

4.4.2 Online OLACD Results at Model's Initialized α

The lists of experiments are described in table 4.1. Column three of table 4.1 gives the architecture for each experiment, i.e., 1-25-5 shows that there are three-layer architecture that includes one input, one hidden and one output layer; it also shows that here is one neuron/node at input layer, hidden layer consist of 25 nodes and there are 5 neurons/ nodes at output layer. Further, we experimented with prediction length l = 1, 5, 10 such that the different models trained for each value of l. We choose fixed window length w = 100 for all experiments.

TABLE 4.2: DEPICTION OFA LIST OF EXPERIMENTS IMPLEMENTED FOR POINT ANOMALY DETECTION USING OLACD

EXP#	MAPPING INPUT TIME SERIES	NETWORK STRUCTURE
1	Yahoo A1Benchmark/real 3	1-25-1
2		1-25-5
3		1-50-10
4	Yahoo A1Benchmark/real 9	1-25-1
5		1-25-5
6		1-40-10
7	Yahoo A2Benchmark/synthetic 13	1-25-1
8		1-25-5
9		1-25-10

Table 4.3 give the OLACD results produced by online Anomaly Detection model for each experiment where α (alpha) is initialized by model itself before starting the actual anomaly detection. The data points of each data set that considered for anomaly detection is shown in column two of table 4.3, AD-model's initialised alpha value, True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR) and False Negative Rate (FNR) for each experiment are given in subsequent column. The last column of table 4.3 gives the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) tagged by online anomaly detection model in each experiment for respective benchmark for univariate time series.

Anomaly detection results for all three data sets for each l-step ahead prediction are plotted in figures 4.6, 4.7 and 4.8, the datasets are collected from Yahoo Anomaly Benchmark [27], which are universally adapted to verify the performance of anomaly/change point detection models [5].

TABLE 4.3: OLACD-LSTM EXPERIMENTAL RESULTS AT MODEL'S INITIALIZED α FOR UNIVARIATE TIME SEREIS WITH TANSITORY ANOMALIES

	Performance Measures						
# of Predictions	data points	alpha	TPR	TNR	FPR	FNR	[TP,FN,FP,TN]
YahooA1 Benchmark Real-3							
1	962	0.0625	0.8667	0.9947	0.0053	0.1333	[13,2,5,942]
5	958	1.0380e-05	1	0.9905	0.0095	0	[15,0,9,934]
10	953	1.2112e-11	1	0.9872	0.0128	0	[15, 0,12,926]
YahooA1 Benchmark Real-9							
1	962	0.0083	0.5	0.9455	0.0545	0.5	[4,4,52,902]
5	958	2.5372e-07	1	0.9435	0.0565	0	[8,0,48,802]
10	953	6.79641e-20	1	0.9527	0.0473	0	[8,0,40,805]
YahooA2 Benchmark Synthetic-13							
1	922	0.0041	0.5	0.9989	0.0011	0.5	[2,2,1,917]
5	918	1.1291e-06	1	0.9836	0.0164	0	[4,0,15,899]
10	913	4.3403e-12	1	0.9813	0.0187	0	[4,0,17,892]

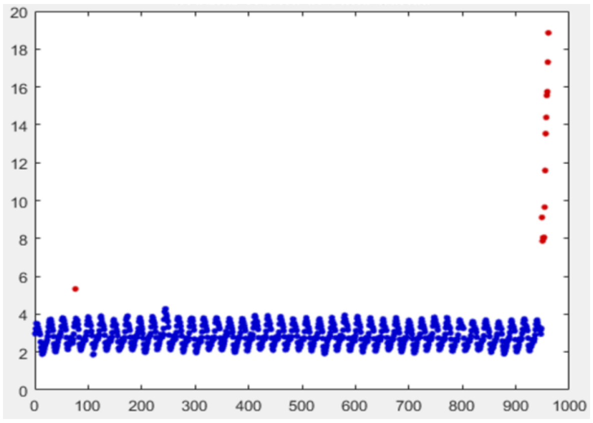
The model is able to detect 13 out 15 anomalous points for Yahoo A1Benchmark/real 3 in case of 1-step ahead prediction shown on 4.6(c), correctly detect 4 out of eight anomalous points in case of Yahoo A1Benchmark/real 3 shown in 4.7(c) and only detect 2 out of 4 anomalous points in case of Yahoo A2Benchmark/synthetic 13 demonstrated in figure 4.8(c). So, we could say that the 1-step ahead prediction is the case of late detection. Whereas in 5-steps ahead and 10 steps ahead prediction for all three datasets OLACD is successfully able to detect all anomalous points at model's initialize alpha (α). So, the True Positive Rate for all three datasets in case of multi-step prediction is 1 and very small False Positive Rate (FPR), which verify that proposed OLACD model is robust to outlier in multi-step ahead settings and capable to detect the outlier/point anomalies in streaming data. In all sub-figures of figure 4.6, 4.7 and 4.8, the blue colours dots represent the true normal predictions, red colour dots show the true anomalous prediction, yellow colour shows the false alarm/ false anomalies predictions, while green colour shows the false normal predictions.

OLACD is initializing alpha (α) using equation 3.18 which would remain fix during complete test run of one experiment. OLACD model only initialize the value of alpha (α) burning period of LSTM training. The propose OLACD model initialize 0.0625, 1.0380e-05 and 1.2112e-11 as alpha for 1,5,10 step ahead prediction for Yahoo A1Benchmark/real 3 dataset after utilizing 400 points for training. Similarly, 0.0083, 2.5372e-07 and 6.79641e-20 is initialized by model for Yahoo A1Benchmark/real 3 dataset for 1,5,10 step ahead prediction respectively after utilizing 400 points for training. In the same way 0.0041, 1.1291e-06 and 4.3403e-12 respectively alpha are initialized by model after utilizing 400 data points for training for Yahoo A2Benchmark/synthetic 13.

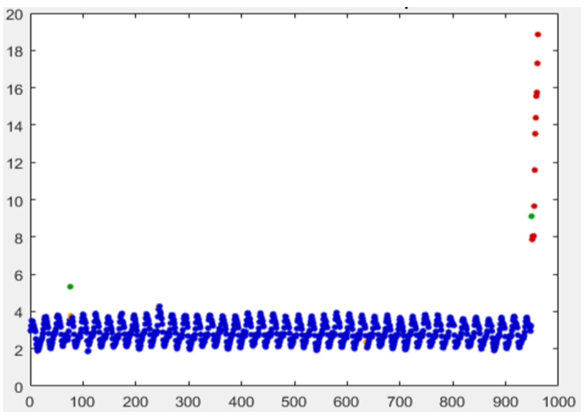
It is revealed that the OLACD predicts the number of false anomalies while detecting the true anomalies and it is shown in table 4.2, figure 4.6, 4.7 and 4.8 that algorithm is capable to handle the short-term anomalies in such as way these transitory anomalies do not lead to a radical change in the model. It is also demonstrated that the model learn with single step-a-head predictions is more reluctant and less responsive to anomalies as compared to multi step-a-head predictions. For instance, for all three datasets the model trained with multi-step-ahead predictions predicts all anomalous point while model trained with single step-a-head prediction is less responsive and reluctant to early detection of anomalous regions/points. Whereas in almost all case the model trained with multi-step-ahead predictions are producing more false alarm as compared to single step-a-head prediction.

TP: True Anomaly Predictions
FP: False Anomaly Predictions
TN: True Normal Predictions
FN: False Normal Predictions

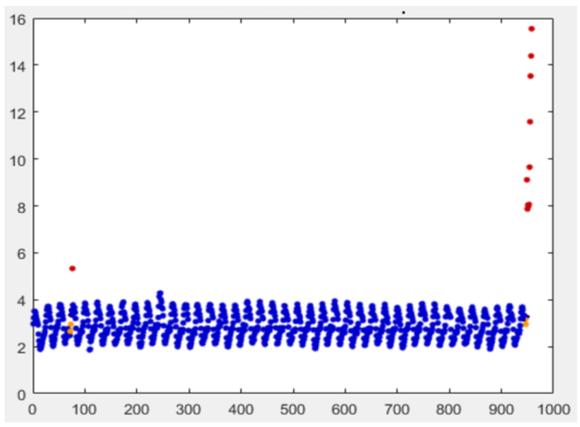
(a) Legends of subfigures (b)-(e)



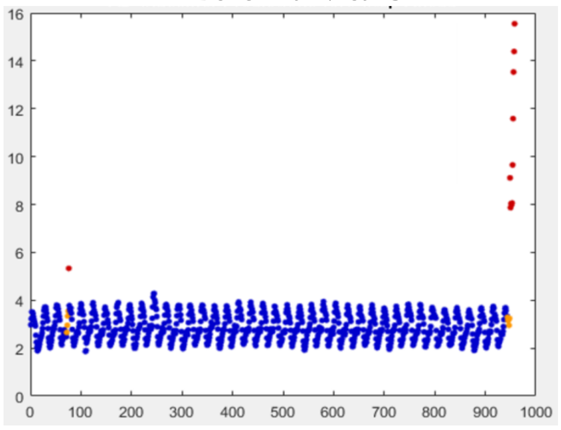
(b) Yahoo A1Benchmark/real 3: Ground Truth



(c) Anomalies/Change Point Detection for 1 Step-a-head Prediction for Yahoo A1Benchmark/real 3



(d) Anomalies/Change Point Detection for 5 Step-a-head Prediction for Yahoo A1Benchmark/real 3



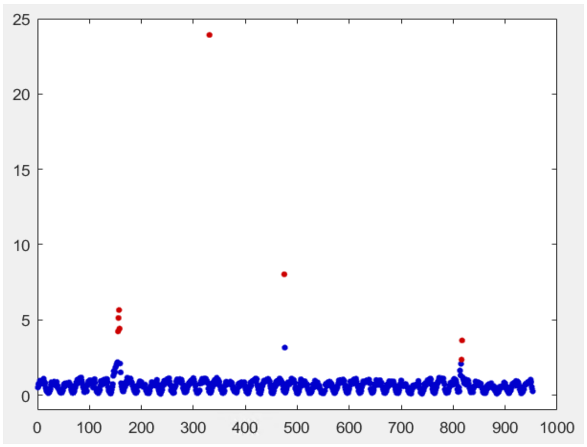
(e) Anomalies/Change Point Detection for 10 Step-a-head Prediction for Yahoo A1Benchmark/real 3

Figure 4.5: AD-LSTM Results for Yahoo A1Benchmark/real 3 Datasets at AD-Model's Initialized α.

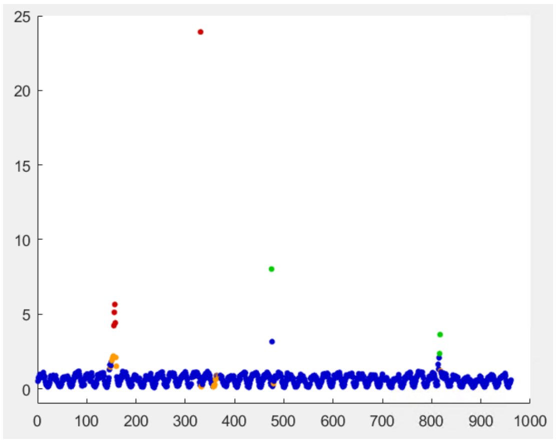
Figure 4.6 (a) gives the legends of subfigures. Ground truth observations are shown in 4.6 (b), where all blue colour represents the normal points and red colour represents the anomalous points/outliers. 4.6(e), gives the anomaly detection for 1-step ahead prediction, 4.6 (d) shows the anomaly detection for 5-step ahead prediction and 4.6(e) gives the anomaly detection for 10-step ahead prediction for Yahoo A1Benchmark/real 3 time series. The colour codes for true anomalous observations are red, true normal are blue, false anomalous are yellow and false normal observations are green. There are total fifteen anomalous points at two different locations for Yahoo A1Benchmark/real 3 dataset and shown in sub-figure 4.6(b). OLACD is able to detect the 13 anomalous points out of 15 in 1-step ahead prediction demonstrated in sub-plot 4.6(b) while able to detect all anomalous points in 5-steps ahead and 10-step ahead prediction represented in 4.6(d) and 4.6(e).

TP: True Anomaly Predictions
FP: False Anomaly Predictions
TN: True Normal Predictions
FN: False Normal Predictions

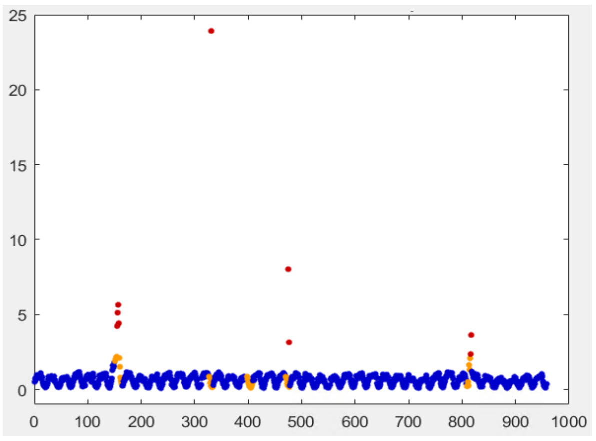
(a) Legends of subfigures (b)-(e)



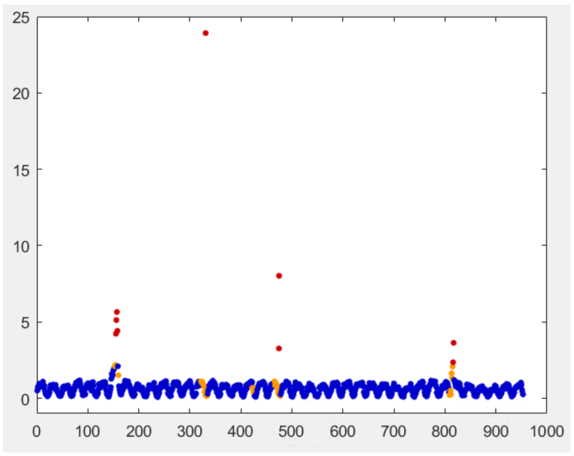
(b) Yahoo A1Benchmark/real 9: Ground Truth



(c) Anomalies/Change Point Detection for 1 Step-a-head Prediction for Yahoo A1Benchmark/real 9



(d) Anomalies/Change Point Detection for 5 Step-a-head Prediction for Yahoo A1Benchmark/real 9



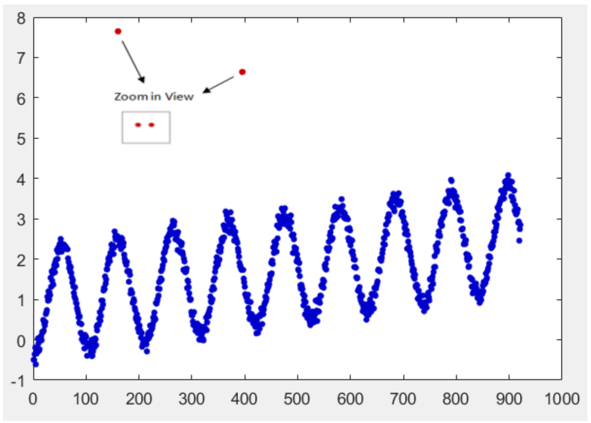
(e) Anomalies/Change Point Detection for 10 Step-a-head Prediction for Yahoo A1Benchmark/real 9

Figure 4.6: AD-LSTM Results for Yahoo A1Benchmark/real 9 Datasets at AD-Model's Initialized α.

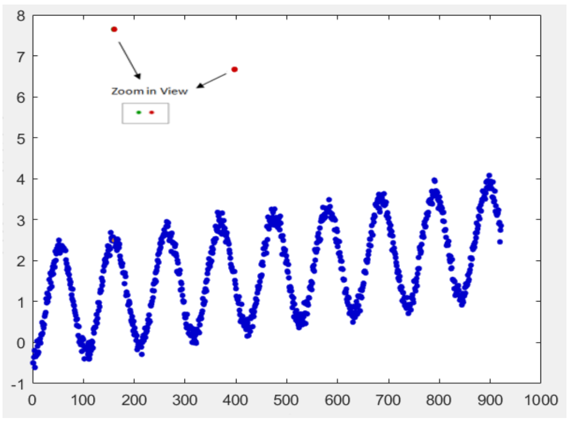
Figure 4.7 (a) gives the legends of subfigures. Ground truth observations are shown in 4.7 (b), where all blue colour represents the normal points and red colour represents the anomalous points/outliers. 4.7(e), gives the anomaly detection for 1-step ahead prediction, 4.7(d) shows the anomaly detection for 5-step ahead prediction and 4.7(e) gives the anomaly detection for 10-step ahead prediction for Yahoo A1Benchmark/real 3 time series. The colour codes for true anomalous observations are red, true normal are blue, false anomalous are yellow and false normal observations are green. There are total eight anomalous points at four different locations for Yahoo A1Benchmark/real 9 data set and shown in sub-figure 4.7 (b). OLACD is able to detect the 4 anomalous points out of 8 in 1-step ahead prediction demonstrated in sub-plot 4.7(b) while able to detect all anomalous points in 5-steps ahead and 10-step ahead prediction represented in 4.7(d) and 4.7(e).

TP: True Anomaly Predictions
FP: False Anomaly Predictions
TN: True Normal Predictions
FN: False Normal Predictions

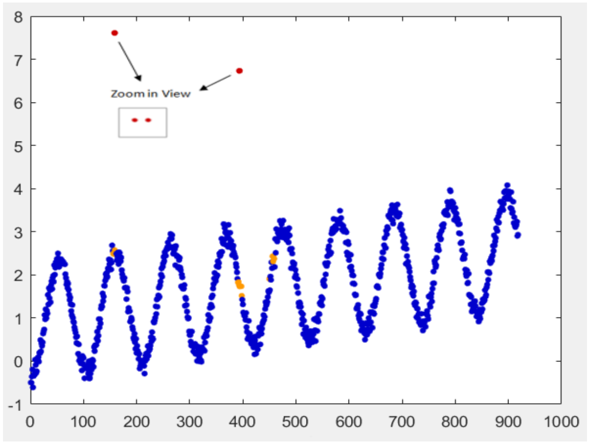
(a) Legends of subfigures (b)-(e)



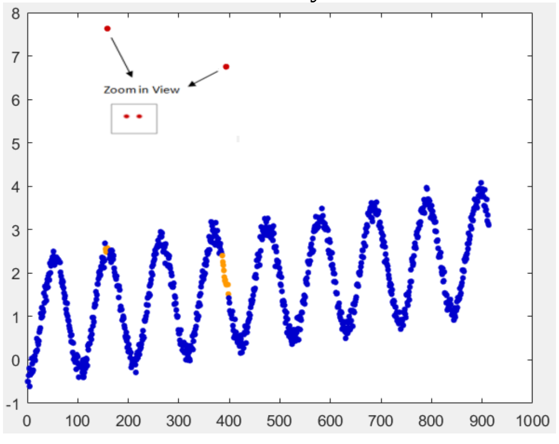
(b) Yahoo A2Benchmark/synthetic 13: Ground Truth



(c) Anomalies/Change Point Detection for 1 Step-a-head Prediction for Yahoo A2Benchmark/synthetic 13



(d) Anomalies/Change Point Detection for 5 Step-a-head Prediction for Yahoo A2Benchmark/synthetic 13



(e) Anomalies/Change Point Detection for 10 Step-a-head Prediction for Yahoo A2Benchmark/synthetic 13

Figure 4.7: AD-LSTM Results for Yahoo A2Benchmark/synthetic 13 Datasets at AD-Model's Initialized α.

Figure 4.8 (a) gives the legends of subfigures. Ground truth observations are shown in 4.8 (b), where all blue colour represents the normal points and red colour represents the anomalous points/outliers. 4.8(e), gives the anomaly detection for 1-step ahead prediction, 4.8(d) shows the anomaly detection for 5-step ahead prediction and 4.8(e) gives the anomaly detection for 10-step ahead prediction for Yahoo A1Benchmark/real 3 time series. The colour codes for true anomalous observations are red, true normal are blue, false anomalous are yellow and false normal observations are green. There are total four anomalous points at two different locations for Yahoo A2Benchmark/synthetic 13 dataset and shown in sub-figure 4.8(b). OLACD is able to detect the 2 anomalous points out of 4 in 1-step ahead prediction demonstrated in sub-plot 4.8(b) while able to detect all anomalous points in 5-steps ahead and 10-step ahead prediction represented in 4.8(d) and 4.8(e).

4.4.3 Online Anomaly Detection Model's Performance Metrics Results

To evaluate the anomaly detection methods on different time series, we use the standard metrics of precision, recall, F-score accuracy, G-mean and Area Under Curve (AUC). Table 4.4 summarise the performance metrics results of online anomaly detection model in three data sets.

TABLE 4.4: PERFORMANCE METRICES FOR UNIVARIATE TIME SEREIS WITH TANSITORY ANOMALIES

Performance Measures												
# of Pred.	**Data points**	**Alpha (α)**	**TP**	**FN**	**FP**	**TN**	**Precision**	**Recall**	**F-score**	**Accuracy**	**G-mean**	**AUC**
YahooA1 Benchmark Real-3												
1	962	0.0625	13	2	5	942	0.722	0.867	0.788	0.993	0.928	0.932
5	958	1.04E-05	15	0	9	934	0.625	1.000	0.769	0.991	0.995	0.997
10	953	1.21E-11	15	0	12	926	0.556	1.000	0.714	0.987	0.994	0.993
YahooA1 Benchmark Real-9												
1	962	0.0083	4	4	52	902	0.071	0.500	0.125	0.942	0.688	0.828
5	958	2.54E-07	8	0	48	802	0.143	1.000	0.250	0.944	0.971	0.985
10	953	6.80E-20	8	0	40	805	0.167	1.000	0.286	0.953	0.976	0.969
YahooA2 Benchmark Synthetic-13												
1	922	0.0041	2	2	1	917	0.667	0.500	0.571	0.997	0.707	0.748
5	918	1.13E-06	4	0	15	899	0.211	1.000	0.348	0.984	0.992	0.995
10	913	4.34E-12	4	0	17	892	0.190	1.000	0.320	0.981	0.991	0.987

As it shown in table 4.4 that accuracy is always high when model is able to detect the maximum number is anomalies regardless of the higher number of false positive detection. The measure recall is pretty straight forward as it is refers to the portion of an anomalous class of interest (anomalous points) that was recognized correctly. So, recall is always higher when model detect all the high number of anomalous points. Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances. Therefore, the model gives the high precision only when it able to predict all possible anomalous points and very less false predictions which is shown in table 4.4. The proposed anomaly detection model performance results given in table 4.4 demonstrates that F-score is always higher when there is high precision because F-score is the harmonic mean of a system's precision and recall values. As it shown in table 4.4 that accuracy is always high when model is able to detect the maximum number is anomalies regardless of the higher number of false positive detections. Therefore, due to the class imbalance, accuracy is not a good measure. Many researchers measure the G-mean which is commonly used as an indicator of model's performance in case with an imbalanced class distribution because the ratio of changes to total data is small. This utilizes both Sensitivity and Specificity measures to assess the performance of the algorithm both in terms of the ratio of positive accuracy (Sensitivity) and the ratio of negative accuracy (Specificity). It is shown in table 4.4 that the G-mean measure of the anomaly detection model is always higher when model is able to predict the maximum number of anomalies/true positive and values of G-mean is compromised when model predicts the false predictions. The performance in AUC is given all data set in table 4.4 which will further discussed in subsequent section.

4.4.4 Online Anomaly Detection Model's Performance Results in Term of AUC

To serve the purpose of validating our proposed novel online anomaly detection algorithm we are using same data sets and performance measures (e.g., AUC) used by existing online anomaly detection technique proposed by Saurav *et. al.* in 2018 [5] and comparing our results in term of Area Under ROC Curve (AUC), which is given in table 5.3. Although the value of α (*Alpha*) is chosen by AD model by itself but to generate the Receiver Operating Characteristic Curve (ROC, each experiment is repeated with different values of α (Alpha). Whereas different values of α will be find as per equation 4.1, including $\alpha_0 = 0$ and $\alpha_1 = 1$ for $n = 0$.

To finding the different value of significance level α *(Alpha)* we reduce the AD-Model's initialized α (*alpha)* using equation 5.1 and perform the experiments repeatedly until we find the no difference in True Positives and False Positives in two/more consecutive run using same parameter specified in table 5.1. Let n is experiments number which performed for particular α, so we repeatedly find the reduced value of $alpha$ as:

$$\alpha_{n+1} = \frac{1}{2^n} \tag{4.1}$$

Where

$$\alpha_1 = OLACPD\ Model'initialized\ Value$$

The method of initialization of α is described in chapter 3. Table 4.3 give the experimental results at model's initialized α alpha. Then Area under ROC is calculated using trapezoidal rules.

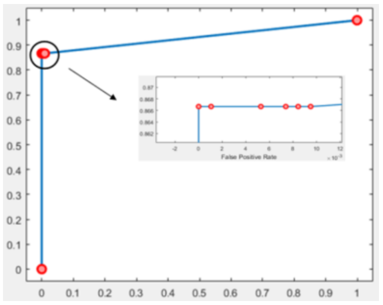
Performance for proposed OLACD is given by *Area under the Receiver Operating Curve (AUC-ROC)*. AUC - ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve which is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination *alpha* (α) value is varied, and AUC represents degree or measure of separability.

In a Receiver Operating Characteristic (ROC) curve the true positive rate (TPR) is plotted in function of the false positive rate (FPR) for different cut-off points. Each point on the ROC curve represents a TPR/FPR pair corresponding to a particular *alpha* (α). A test with perfect discrimination (no overlap in the two distributions) has a ROC curve that passes through the upper left corner (100% TPR, 100% FPR). Therefore, the closer the ROC curve is to the upper left corner, the higher the overall accuracy of the system [5].

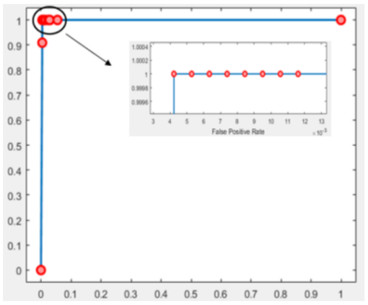
To find out the Area Under ROC Curve (AUC), each experiment is repeated with different values of α (Alpha) for same time series to produce the Receiver operating characteristic (ROC) curve. Whereas different values of α for each subsequent experiment is find as per equation 4.1, including $\alpha_1 = 1$ for $n = 0$. As per equation 4.1 α will very quickly be zero, so the experiment is done for $\alpha_0 = 0$ as well. Then Area under ROC is calculated using trapezoidal rules. The performance of proposed OLACD model is represented in terms of area under the receiver operating characteristic curve.

The lower left most corner of ROC in each subfigure of figure 4.9, shows that if α=0, every data point is considered by model as normal so at α=0, TPR=0 and FPR=0. But at α= 1, every data point is considered as anomaly, so at α=1, TPR=1 and FPR=1, which is shown in upper right corner of ROC.

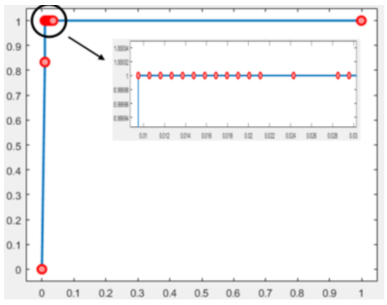
(a) Legend of Subfigures



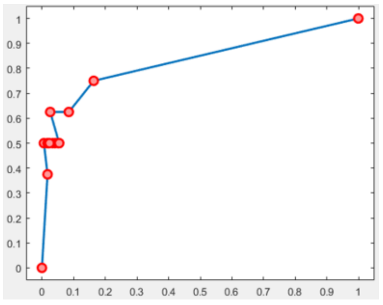
(b) AUC for Yahoo A1Benchmark/real 3: 1 step-a-head Predictions



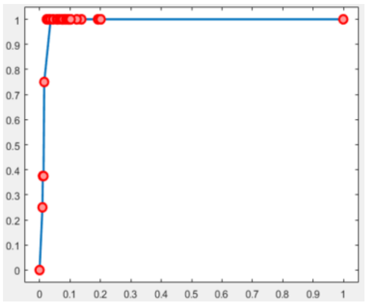
(c) AUC for Yahoo A1Benchmark/real 3: 5 step-a-head Predictions



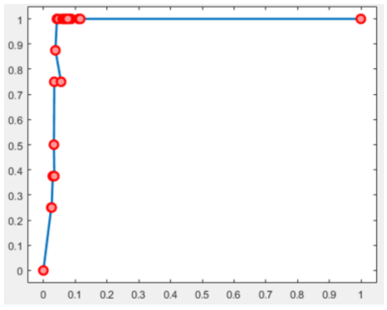
(d) AUC for Yahoo A1Benchmark/real 3: 10 step-a-head Predictions



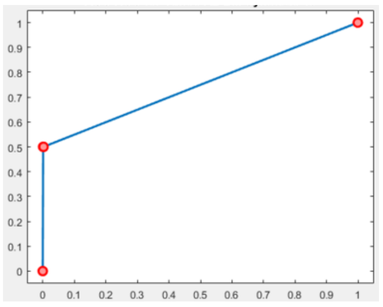
(e) AUC for Yahoo A1Benchmark/real 9: 1 step-a-head Predictions



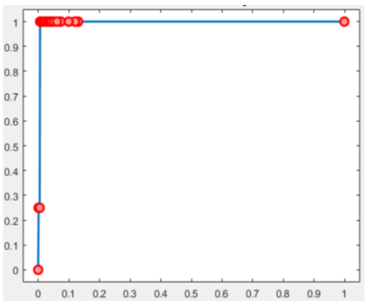
(f) AUC for Yahoo A1Benchmark/real 9: 5 step-a-head Predictions



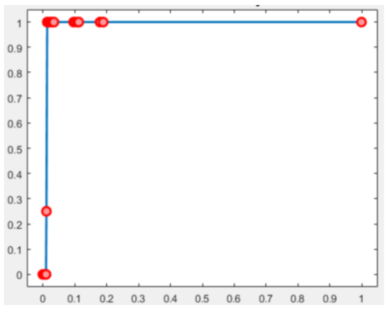
(g) AUC for Yahoo A1Benchmark/real 9: 10 step-a-head Predictions



(h) AUC for Yahoo A2Benchmark/synthetic 13: 1 step-a-head Predictions



(i) AUC for Yahoo A2Benchmark/synthetic 13: 5 step-a-head Predictions



(j) AUC for Yahoo A2Benchmark/synthetic 13: 10 step-a-head Predictions

Figure 4.8: AUC for Proposed Anomaly Detection Algorithm.

False positive rate (FPR) is given on x-axis and True positive rate (TPR) is given on y-axis of all subfigures.4.9 (b), (e) and (h) shows the AUC for 1-step Prediction for their respective datasets. 4.9 (c), (f) and (i) are AUC for 5-step Prediction while 4.9 (d), (g) and (j) gives the AUC for 10-step Prediction for their respective time series. Legends for each sub-figure given in 4.9 (a). AUC for Yahoo A1Benchmark/real 3 dataset is 0.932, 0.997, 0.993 respectively for 1, 5 and 10 step-a-head predictions shown in 4.9 (b), (c) and (d). AUC for Yahoo A1Benchmark/real 9 dataset is 0.828, 0.985, 0.969 respectively for 1, 5 and 10 step-a-head predictions shown in 4.9 (e), (f) and (g). AUC for Yahoo A2Benchmark/synthetic 13 dataset is 0.748, 0.995 and 0.987 respectively for 1, 5 and 10 step-a-head predictions shown in 4.9 (g), (i) and (j).

AUC for three of observed change point time series are demonstrated in figure 4.9. AUC for Yahoo A1Benchmark/real 3 dataset is 0.932, 0.997, 0.993 respectively for 1, 5 and 10 step-a-head predictions shown in 4.9 (b), (c) and (d). AUC for Yahoo A1Benchmark/real 9 dataset is 0.828, 0.985, 0.969 respectively for 1, 5 and 10 step-a-head predictions shown in 4.9 (e), (f) and (g). AUC for Yahoo A2Benchmark/synthetic 13 dataset is 0.748, 0.995 and 0.987 respectively for 1, 5 and 10 step-a-head predictions shown in 4.9 (g), (i) and (j).

4.5 Discussion and Conclusion

We have proposed a novel anomaly detection algorithm using LSTM (OLACD) that detects the anomalies based on the statistics of predictions error, details of model and methodology is given in chapter 3. In this chapter several experiments are undertaken as a means to fully comprehend the effectiveness and validity of OLACD algorithm. The formulated training and anomaly detection algorithm are experimentally analysed by implementing nine experiments using fully connected LSTMs model; train with backpropagation through time (BPTT), mapping univariate time series and these were implementing in MATLAB. Details of all experiments along with the parameter is given in table 4.2, where comparison of performance results in term of AUC is illustrated in table 4.5.

TABLE 4.5: COMPARISON OF RESULTS IN TERM OF AUC FOR PROPOSED ONLINE ANOMALY DETECTION AND CHANGE-POINT DETECTION MODEL USING LSTM FOR UNIVARIATE TIME SEREIS WITH Saurav *et. al.* [5]

DATA SETS	**PERFORMANCE RESULTS FOR TIME SERIES**					
	PREDICTION 1		**PREDICTION 5**		**PREDICTION 10**	
	AUC (**Saurav *et. al.*** [5])	**AUC** (**PROPOSED**)	**AUC** (**Saurav *et. al.*** [5])	**AUC** (**PROPOSED**)	**AUC** (**Saurav *et. al.*** [5])	**AUC** (**PROPOSED**)
YahooA1 Benchmark Real-3	0.985	0.932	0.959	0.997	0.967	0.993
YahooA1 Benchmark Real-9	0.879	0.828	0.779	0.985	0.792	0.969
YahooA2 Benchmark Synthetic-13	0.630	0.748	0.716	0.995	0.869	0.987

The anomaly detection techniques uses statistics of error for anomaly detection in offline way for a batch of data is proposed in [8, 14]. The techniques proposed in [8, 14] are not effective for streaming data as they only computes error statistics when full batch of data is trained on normal training data set and then perform anomaly detection on normal and anomalous test data sets, which make these techniques ineffective for time series data. One of the online versions of technique that computes the statistics of error for streaming data and use these statistics for either transitory (outlier) anomaly detection or permanent (change-point) anomaly detection is proposed by Saurav *et. al.* in 2018 [5], which computes anomaly score based on error statistics and update the model parameter accordingly. Saurav *et. al.* in 2018 [5] validates the results of their online model in term of Area Under ROC Curve (AUC).

As our proposed novel approach for anomaly detection is work in an online manner for streaming data and we also validate the results of our online approach in term of AUC, so we are comparing our results with online anomaly detection technique proposed by Saurav *et. al.* in 2018 [5]. For producing direct comparison, we are using same data sets and performance measures to validate our online anomaly detection model that use In [5].Hence, the performance of proposed novel online anomaly detection model is compared with the model by Saurav *et. al.* [5] in term of AUC and comparison is given in table 4.5. The results of investigations show that the proposed OLACD is working as good as the existing online model proposed in [5] in some cases, and novel OLACD is working better than existing model in most of case as shown in table 4.5. Proposed OLACD model seems to be robust to anomalies while working with multi-step ahead predictions as compare with the 1-step ahead prediction.

It is shown in table 4.3, 4.4 and 4.5 that OLACD model is less responsive to short-term anomalies in the 1-step ahead predictions and performing late detection rather than early detection, whereas the 1-step ahead predictions give advantage in term that it detect small number of false positves as compare to the multi-step ahead predictions. So, the AUC for 1-step ahead predictions for two data sets is lower than the model propsed in [5], for instance the AUC for YahooA1 Benchmark Real-3 data set for 1-step ahead prediction is 0.932, and for YahooA1 Benchmark Real-9 data set for 1-step ahead prediction is 0.828 which is lower than the online model of Saurav *et. al.* [5] that is 0.985 for YahooA1 Benchmark Real-3 data set and 0.828 for YahooA1 Benchmark Real-9 data set. In one case OLACD give the better results for 1-step ahead prediction for YahooA2 Benchmark Synthetic-13 that is 0.748 which is bigger than the model given in [5] that is 0.630.

Comparison results demonstrats that our propsed online anomaly detection algorithm is more robust to the outlier when it work in multi-steps ahead predictions and performing early anomaly that is one of the main characteristics of anomaly detection techniques. Therefore, the AUC for 5-step ahead predictions for all three compared datasets is higher than the model propsed in [5], for instance the AUC for YahooA1 Benchmark Real-3 data set for 5-step ahead prediction is 0.997, for YahooA1 Benchmark Real-9 dataset is 0.985 and for YahooA2 Benchmark Synthetic-13 dataset that is 0.995. Which are prety higher in most cases than the model given in [5] that is 0.959 for YahooA1 Benchmark Real-3 dataset, 0.779 for YahooA1 Benchmark Real-9 dataset and 0.716 for YahooA2 Benchmark Synthetic-13 dataset. Consequently, the AUC for 10-step ahead predictions for all three compared data sets is higher than the model propsed in [5], for illustrate the AUC for YahooA1 Benchmark Real-3 dataset for 10-step ahead prediction is 0.993, for YahooA1 Benchmark Real-9 dataset is 0.969 and for YahooA2 Benchmark Synthetic-13 dataset that is 0.987. Which are higher than the model given in [5] that is 0.967 for YahooA1 Benchmark Real-3 dataset, 0.792 for YahooA1 Benchmark Real-9 dataset and 0.869 for YahooA2 Benchmark Synthetic-13 dataset.

The AUC for 10-step ahead predictions for all three compared datasets is also higher than the model given in [5], for illustration the AUC for YahooA1 Benchmark Real-3 data set for 1-step ahead prediction is 0.993, for YahooA1 Benchmark Real-9 dataset is 0.969 and for YahooA2 Benchmark Synthetic-13 dataset that is 0.987. Which are higher than the model given in [5] that is 0.967 for YahooA1 Benchmark Real-3 dataset, 0.779 for YahooA1 Benchmark Real-9 dataset and 0.716 for YahooA2 Benchmark Synthetic-13 dataset. Consequently, Therefore, the AUC for 5-step ahead predictions for all three compared data sets is higher than the model propsed in [5], for instance the AUC for YahooA1 Benchmark Real-3 dataset for 1-step ahead prediction is 0.993, for YahooA1 Benchmark Real-9 dataset is 0.969 and for YahooA2 Benchmark Synthetic-13 dataset that is 0.987. Which are higher than the model given in [5] that is 0.967 for YahooA1 Benchmark Real-3 dataset, 0.792 for YahooA1 Benchmark Real-9 dataset and 0.869 for YahooA2 Benchmark Synthetic-13 dataset.

It is given in table 4. 2 that model trained with 1 -step ahead predictions produces less number of false positives as comapre to multi-step predictions (5-steps ahead or 10-steps ahead) so we can say that there is a trad off between early anomaly detection and prediction horizon, but OLACD model is more resposive to outlier in multi-step ahead predictions.

Chapter 5

Online Change Point Detection for Univariate Time Series

The methodology for anomaly and change-point detection which was developed in Chapter 3 is verified for *Permanent Anomalies/Change-Points* in this chapter. The section 5.1 delivers introduction to the chapter. Different types of change-points are discussed in 5.3 which is followed by datasets description in section 5.3. The experimental setup is determined in 5.4 while the experimental results for Online Anomaly and Change-Point Detection (OLACD) model for change-point data sets is demonstrated in section 5.5 that followed by the discussion and conclusion in 5.6.

5.1 Introduction

This chapter presented the experimental design and implementation of proposed novel anomaly detection technique *Online Anomaly Detector for Univariate Time Series* (OLACD) that utilised to experimentally validate the model for the change point detection of univariate time series. The main objective of the presented section of research report is to investigate and hence understand and start to build a comprehensive knowledge for experimental design and implementation of proposed OLACD model for change point detection in time series. As described in chapter 3 OLACD model is working in two parts. First, *The Univariate time series prediction using LSTM* which will first learn the prediction model for each incoming data point from time series $X = \{x_1, x_2, \dots, x_t\}$, using LSTMs in an online way and compute the error vectors. Second, *the anomaly detector* that estimate of the error vectors is pass on to this to compute the prediction error distribution for each trained data point. For the incremental prediction error distribution for every data point of time series at each time stamp t, the online estimation of mean (μ_t), covariance (V_t) and threshold (τ_t) should be done using their respective recursive set of equations 3.6-3.15, which is then used to detect the change point or set of change points. The error vector e_t are modelled to fit a multivariate Gaussian distribution

$MV\mathcal{N}(\mu_t\ V_t)$ as the error vector is multidimensional in case of multistep-a-head predictions. The likelihood p_t of observing an error vector e_t is given by the value of $MV\mathcal{N}$ at e_t. The dimension of error vector is determined by the prediction horizon. When d consecutively x_t observations are classified as '*anomalous'* if $p_t < \tau_t$ at their respective time stamp t then it considers as change point. where anomalous points belong to positive class and normal points belong to negative class. Unlike other online anomaly detection method [15] our proposed model is not relying on user pre-defined parameter such as threshold τ, rather it adapts the threshold and updated with each time step in an incremental manner.

The series of experiments, which enabled the author to comprehend OLACD algorithms for LSTM training and anomaly detection of time series in an online way. One of our goals and indeed a requirement of experiments are to develop the necessary mathematical knowledge to enable anomaly detection (AD) in an online way for continuous data. Once the required OLACD model has been appropriately working for one-step ahead prediction, we extended the experiments to multi-step ahead predictions; as recently the LSTMs has become a norm for designing and implementing various computing machines [19] that efficiently handles the spatial and temporal dependencies in streaming data. The proposed OLACD for change point detection is then experimentally validated and its results are comparing with the existing anomaly detection techniques.

We validate the efficiency of the proposed novel approach for change point detection of univariate time series via experiments on publicly available and proprietary three real-world [ref..] and two synthetic datasets [base paper] and compare the results in term of Area Under Curve (AUC) with algorithm proposed by Saurav *et. al.* in 2018 [5], where they found single/multi-dimensional error and compute anomaly score using error vector, afterward the RNN was updated based on anomaly score and pre-process the input data using local normalization [5].

We consider several variants of non-stationary real-valued univariate time series where statistical components (i.e., mean or the standard deviation) of the time series values over a window change with time. In this chapter we are analysing the proposed OLACD system performance for permanent changes i.e., change point, which either one change point or set of change points. These changes may happen suddenly, or continuously. We first describe the datasets considered for experimental evaluation, then describe change point detection model setup and hyperparameters selection process followed by results and observations.

5.2 Change point for Streaming Data

Humankind is on the expedition to digitalise the world and today's world is changing very fast and these changes occur in every aspect of life. Consequently, the capability to identify, adapt, and respond to the change play a significant role in all aspects of life. The real physical world is habitually represented in a variety of model or information system and the changes in the real physical world are indicated in terms of the changes in data or model built from that data. Thus, the nature of data is changing with time. The advancements in technology results in the data flood. The volume of data generated has been estimated to exceed the 6.7ZB in 2020 zettabytes in 2020 as reported in the IDC Survey [125]. The data deluge makes the traditional techniques including traditional parallel models and distributed framework inappropriate for analysing, processing, interpretation and storing these gigantic data sets. Data surges require a new generation of computing tools and methods that Jim Gray calls the 4th paradigm in scientific computing [126]. There have been some emerging computing paradigms that meet the requirements of big data such as Parallel batch processing model perhaps it only deals with the stationary massive data [9]. Nevertheless, the progressing data continuously arrives with high speed and, indeed, online data stream processing is the main approach to dealing with the problem of three characteristics of big data that includes big volume, big velocity, and big variety. The streaming data processing is a model of big data processing. Streaming data is temporal data in nature and in addition the streaming data may include spatial characteristics, such as, the geographic information systems can produce spatial-temporal data stream. Streaming data processing have been implementing in real-world systems such as InforSphere Streams (IBM) [127], Rapidminer Streams Plugin [128], StreamBase [129] and Massive online Analysis (MOA) [130]. To cope with the high-speed data streams, a hybrid model that combines the benefits of both streaming data processing model and parallel batch processing model is proposed, such as S4 [131], Storm [132], and Grok [133].

One of the challenges facing by streaming data processing is the changing nature of streaming data. The capability to identifying changes or patterns in the fundamental processes generating data contributes to the accomplishment of processing high-speed data streams. Thus, in dynamically changing and nonstationary environments, the data distribution can change over time, yielding the phenomenon of *change point* [9].

‘Change detection is the process of identifying differences in the state of an object or phenomenon by observing it at different times or different locations in space’. In the streaming context, ‘change detection is the process of segmenting a data stream into different segments by identifying the points where the stream dynamics changes’[9]. Change detection detects whether a change occurs and react to the presence of the change. The problem of detecting and locating the change has been investigated in the computational science and statistics in the problems of change point detection (CPD). Methods for CPD can be roughly categorized as a) offline vs. online, b) univariate vs. multivariate and c) model-based vs. nonparametric [9].

Traditionally in machine learning, previously collected data is processed in an *offline* mode. For example, predictive models are trained using historical data which is presented as an input data and output data. Models trained in such a manner can be afterwards utilised for predicting the output for latest/new unseen input data.

However, very often in real world the data comes in the form of streams. It is impractical and often infeasible to accommodating large volumes of streaming data in the machine’s main memory, therefore, only an *online* processing is appropriate. In case of online processing, predictive models can be trained incrementally by continuously updating data.

But computational efficiency is not the only issue in learning from data streams. Many data streams research deal with adaptive learning only to some extent, though the main focus remains on generating learning algorithms incremental and optimizing the stability and balance of computational resources and the predictive accuracy.

Learning algorithms are frequently employed in dynamic environments, which are changing unexpectedly over time and one of the advantageous and appropriate property of these algorithms is their ability of incorporating the new unseen data. If the data-generating process is not rigorously and precisely stationary that primarily applies to most of the real-world applications, the underlying changes, which we are predicting, may be changing over time. The ability to adapt to such *changes* can be seen as a natural and desired extension for the incremental learning systems [20, 55] that learn predictive model sample by sample. *Adaptive learning algorithms* can be considered as advanced and sophisticated incremental learning algorithms that are capable to adapt to progression of the data-generating process over time.

Adaptive learning refers to updating predictive models online during their operation to react to the *change point.* Over the last decade, research related to learning with *change detection* has been increasingly growing, and many change-aware adaptive learning algorithms have been developed [9].

Several research and reviews are limited to specific application. A set of requirements for complex adaptive systems to be used for defence [17 and for soft sensors [18]. The article [19] focuses on describing various ways in which data distribution can change over time and only covers adaptation technique in brief from the dataset shift community perspective but, mostly leaving out the works on change point detection and concept drift. In this chapter we identified that change occurs in time series using novel online change detection method that means showing a change point, or a set of change points in time series, while all forecasting process is work in online/incremental manner.

5.2.1 Types of Changes in Data Over Time

A data stream can be described as a hypothetically infinite sequence of one-dimensional values, $X = \{x_1, x_2, \dots, x_t\}$, where x_t is arriving at time t. Assume that these vectors are generated by a data source S_1, but that at some time, this data source S_1 is replaced with another source, S_2. The purpose is to detect from the data that the source has changed from S_1 to S_2. A univariate time series problem for streaming data is demonstrated in Figure 5.1, where significant change in the mean is observing and it assume that it would be sufficient to identify the change [3].

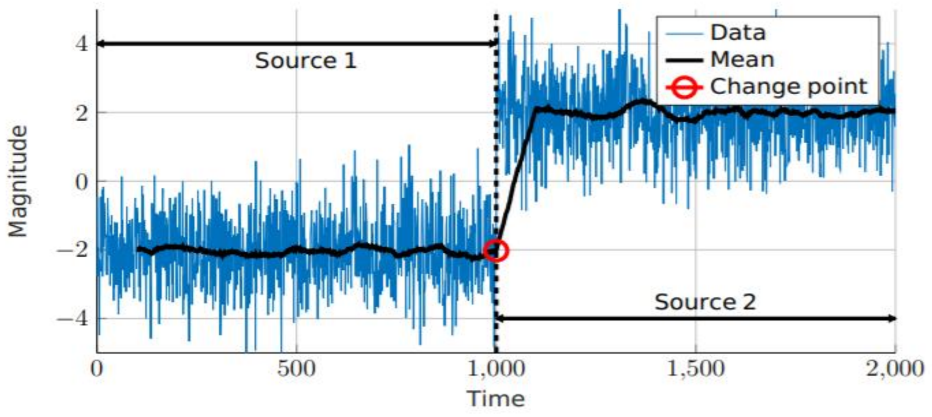
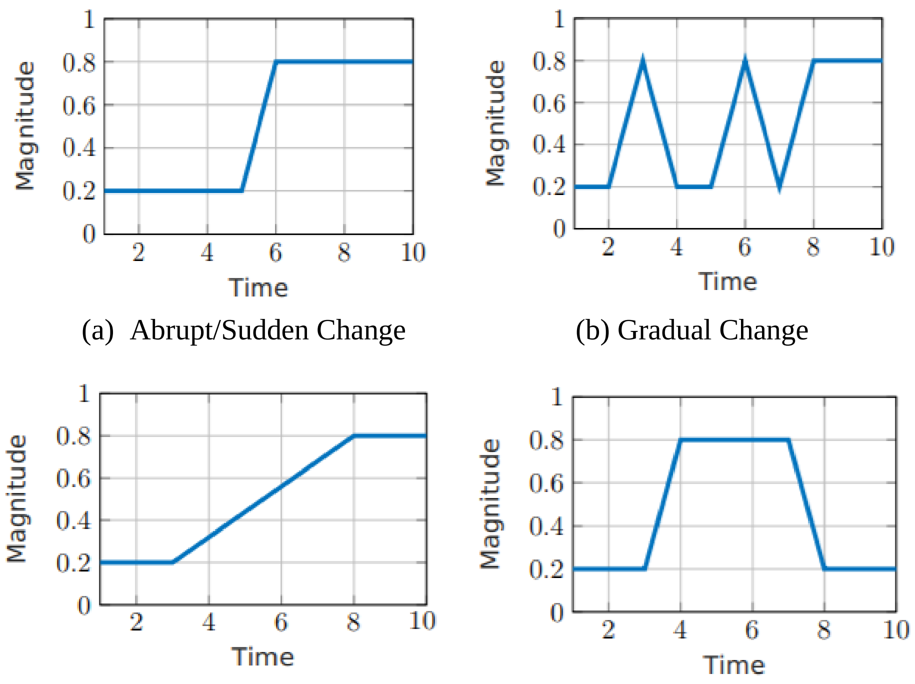


Figure 5.1: Sample streaming data with a change point at t= 1000 from Data Source 1 to Data Source 2.

’Change’ is a broad term, difficult to describe and measure, and extremely determined on the application at hand. Since change is context-dependent, it is useful to differentiate it so that we can reason about the type of change we want to detect.



(a) Abrupt/Sudden Change (b) Gradual Change

(c) Incremental Change (d) Re-occurring Concepts

Figure 5.2: Illustration of types of changes over time between two sources, S1 and S2 [3]

Gama [20] distinguishes the two dimensions of analysis of change in streaming data; the causes of change, and the rate of change and we primarily consider that how a single variable in a sequence might change over time. There are four frequent patterns of change over time which are frequently considered and examined in the literature [20], that are demonstrated in a single variable in figure 5.2. A change can be *abrupt/sudden* [3], change occurs where S_1 is replaced with S_2 at a single time point t shown in figure 5.2(a) at $t = 5$. An example of such a change might be a catastrophic sensor failure, or replacement of a sensor with another sensor that has a different calibration in a chemical plant.

When a change occurs over a range of time points than it is described in the literature as a gradual change where data points are sampled with growing probability over time from S_2 and with reducing probability from S_1, until the concept has changed completely to S_2. For instance, a person gets a product they have never bought before and buys it. Getting developed a buying preference, then in subsequent shopping trips they buy this product with increasing frequency until it become a primary choice. This category of change referred to [3] as *gradual change* and is illustrated in Figure 5.2(b).

A change can be *incremental* if it is defined by a slow progression from S_1 to S_2 via several mixed intermediate concepts, as demonstrated in Figure 5.2(c). An example could be sensor slowly wears off and becomes less accurate. The type of change is a *reoccurring concept*, where a previously encountered concept reappears periodically but unpredictably, shown in Figure 5.2(d). A fifth category from Gama [20], which has not been considered as change rather it is *outliers*. This is because change detection algorithms should ideally be robust to noise and outliers and one of the challenges for change point detection algorithms is not to mix the true change with an *outlier* [3] or noise, which refers to a once-off random deviation or anomaly. No adaptivity is needed in the case of *outlier*. Changes can be further characterized by severity, predictability, and frequency [20].

5.3 Data Sets Description

Much of the world's data is streaming, time-series data, where anomalies and changes in streaming data give significant information in critical situations; examples abound in domains such as finance, IT, security, medical, and energy. Detecting anomalies and change points in streaming data is a difficult task, requiring detectors/algorithms to process data in real-time and learn though simultaneously making predictions. One of the few benchmarks to adequately test and score the efficacy of real-time anomaly detectors, the Numenta Anomaly Benchmark (NAB) [6] attempts to provide a controlled and repeatable environment to test and measure anomaly detection algorithms on streaming and time series data which now a days consider as one of the benchmark datasets for evaluation and verification of novel anomaly detection techniques [5]. The ideal anomaly detector would possibly detect all anomalies and change points as soon as possible, generate no false alarms, working with real-world time-series data and automatically adapt to changing statistics of input stream.

It is often exorbitantly expensive to collect an accurately labelled anomalous data instances that covers all different categories of anomalous behaviour [29]. An important element of the NAB dataset is the enclosure of real-world data with anomalies and change points for known causes. NAB dataset was proposed as a quality collection of time-series data with labelled anomalies and change points. NAB dataset is considerably well suited to be a standard benchmark for streaming applications in research community. The NAB repository [6] is open source and publicly available and contains the data along with scripts which consist of the Numenta Anomaly Benchmark (NAB) datasets. NAB is a novel benchmark for evaluating algorithms for detecting anomalies and change points in streaming, real-time applications which is composed of over 50 labelled real-world and artificial timeseries data. For the purpose of comparing the model evaluation results in term of AUC we consider the same data set that are used in existed research [5].

We consider three real-world and two synthetic univariate time series which are taken or derived from Numenta Anomaly Benchmark (NAB) [6]. The source for the cases considered from NAB datasets are provided in Table 5.1. We consider change points in amplitude domain (i.e., change in the range of values taken) as well as in frequency domains (i.e., change in periodicity for periodic time series). More specifically, the datasets are considered to cover at least one of the following scenarios (as identified in [20]):

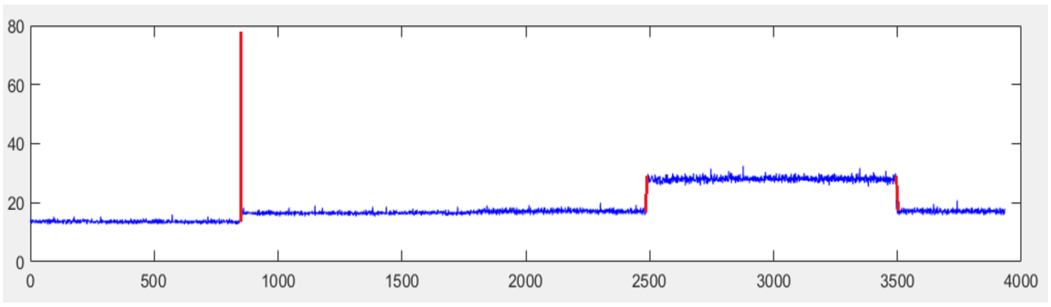
- Sudden change due to mean change (figures 5.3a).
- Continuous change (figures 5.3b): continuous change in the normal range of values.
- Incremental change in amplitude domain (figure 5.3c): containing various intermediate ranges of values while changing from one range of values to another over a significant time.

- Sudden change due to change in frequency for periodic time series (figure 5.3d).
- Sudden change due to amplitude change in periodic series (figure 5.3e).

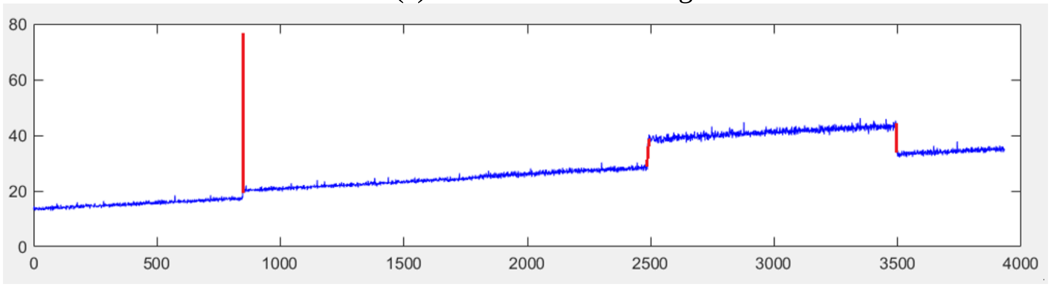
Each time series in NAB consists of several instances of data at different time stamps. In each data file that comprises of time series data instance and a Boolean attribute '*label*' indicating if the value at the particular time stamp is tagged as anomalous or normal. We are using *value* attribute as input time series for all the experiments. The main reason for selecting this data set for evaluation is the availability of the change point anomalies labels, which are not commonly available in publicly available streaming data sets. Secondly, a published online anomaly detection approach used same data set and shown their results in term of AUC, and we will compare our results with that available online anomaly detection algorithm [5].

Figure 5.3 illustrates the five time series data sets which we considered for experimental validation of proposed online change detection, time steps are displayed on x-axis and observation values are shown in y-axis, anomalies and change points are illustrated with red colour while blue colours is defined for normal observations.

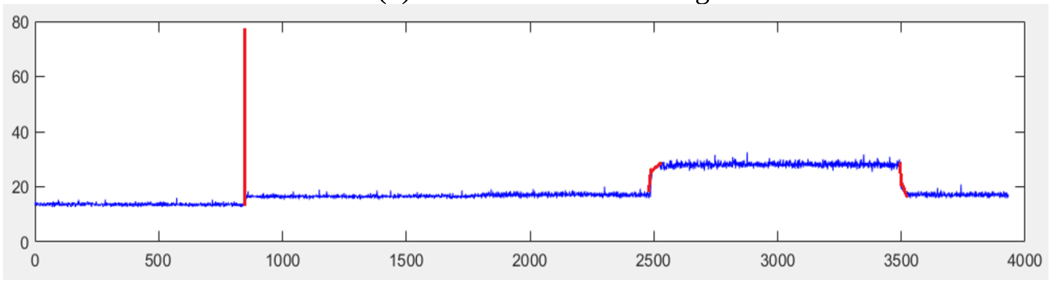
Figure 5.3(a) and (b) shows the dataset taken from NAB repository [6] which are univariate time series having 4032 observations including 4027 normal data points and 2 change point and 3 anomalies (3 anomalous data points are considers as outlier at first location in figure 5.3(a and b)). In both figures 5.3 (a and b) the 3 anomalous points (outlier) are shown in red colour and existed at time 947, 948 and 949 while two single change points are existed at time 2586 and 3594. The time series shown in 5.3(a) has sudden changes due to mean change while the time series given in 5.3(b) is showing continuous changes at two different locations of time and 3 anomalies (outlier). Figure 5.3(c) is also taken and derived from NAB repository [6] that comprises of 4032 observations 3966 normal observation and 1 anomaly (comprises of 3 values) and 2 change points that comprises the set of 63 observation. In figures 5.3 (c) the 3 anomalous points (outlier) are shown in red colour and existed at time 947, 948 and 949 while first set of change points are existed at time 2586-2622 and first set of change points are existed at time 3594-3621. This time series is showing incremental change in amplitude domain containing various intermediate ranges of values while changing from one range of values to another over a significant time.



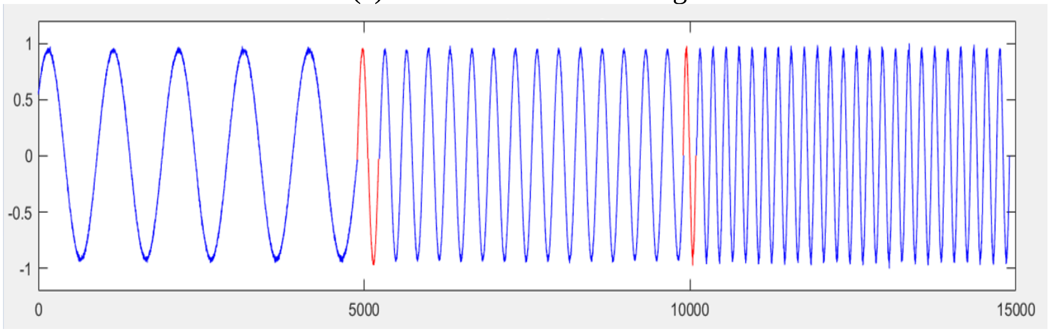
(a) NAB- Sudden Change



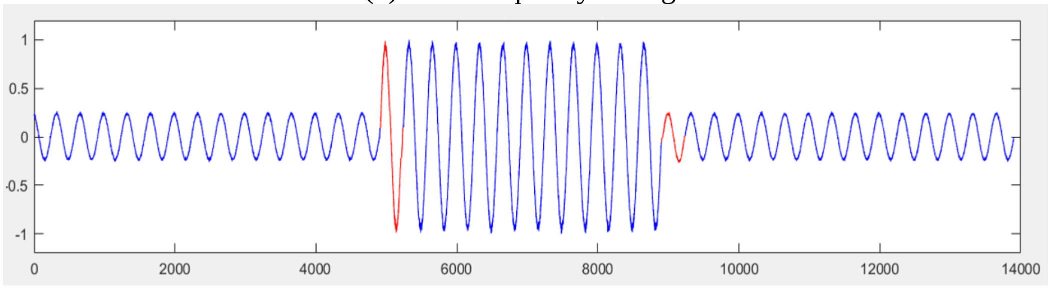
(b) NAB-Continuous Change



(c) NAB-Incremental Change



(d) Sine-Frequency Change



(e) Sine-Amplitude Change

Figure 5.3: Dataset/Time Series illustration used for change point detection using LSTM.

The two synthetic data sets are designed using sine wave which represents the multiple change point in frequency and amplitude domain. The figure 5.3(d) is demonstrated frequency change in sine wave dataset which is generated by combining three different frequencies and amplitude of 2 and noise with mean 1 and standard deviations of 0.01. The first sine wave is comprising of 5000 observations with 1000 observation per cycle shown in figure 5.3(d) at time 0 to 5000, second sine wave is comprising of 5000 observations and 333 observations per cycle shown in figure 5.3(d) at time 50001 to 10000 and finally the third sine signal is comprising of 5000 observations with 200 observations per cycle shown in figure 5.3(d) at time 10001 to 15000. Thus, there are altogether 15000 observations with 14466 normal data points and two sets of change points, such as 333 set of change point observation at 5001 to 5334 and 201 set of change point observation at 10001 to 10202 that is demonstrated in figure 5.3(d).

The figure 5.3(e) is illustrated amplitude change for sine wave dataset which is created by combining of three different amplitude and same frequency (333 observation per cycle) and noise with mean 1 and standard deviations of 0.01. The first sine signal is comprising of 5000 observations with 333 observation per cycle with amplitude of 0.5 starting at time 0 to 5000 figure 5.3(e), second sine signal is comprising of 4000 observations with 333 observations per cycle starting at time 5001 to 9000 with amplitude of 2 and third sine signal is comprising of 5000 observations with 333 observations per cycle with amplitude of 0.5, starting at time 9001 to 14000. Thus, there are altogether 14000 observations with 13334 normal data points and two sets of change points, such as first set of 333 change point observation starting at time 5001 to 5333 and second set of 333 point for change point starting at 9001 to 9333 that is demonstrated in figure 5.3 (e).

Blue region in plots shows the normal time series recordings while red colour represents the anomalous region. Normal data points are labelled with zeros while all anomalous points have labelled with ones. The labels for synthetic data set are generated by author while all labels for NAB datasets are taken from NAB repository [6] which is publicly available. Description of all data sets that used for change-point detection are summarised in table 5.1.

TABLE 5.1: DEPICTION OFA LIST OF EXPERIMENTS IMPLEMENTED FOR CHANGE-POINT DETECTION USING OLACDTABLE 5.2: SUMMARY OF ALL DATA SETS USED FOR ANOMALY AND CHANGE_POINT DETECTION

UNIVARIATE INPUT TIME SERIES	TOTAL OBSERVATIONS	NORMAL OBSERVATIONS	ANOMALOUS OBSERVATIONS	NUMBER OF CHANGE - POINTS	SET OF CHANGE -POINTS OBSERVATIONS
NAB Sudden Change	4032	4027	3	2	2
NAB Continuous Change	4032	4027	3	2	2
NAB Incremental Change	4032	3966	3	2	CP1=38 CP2 = 28 Total = 66
Sudden frequency change (Sine Wave)	15000	14466	0	2	CP1=333 CP2 = 201 Total = 534
Sudden amplitude change (Sine Wave)	14000	13334	0	2	CP1=333 CP2 = 333 Total = 666

5.4 Experimental Setup

For each observed dataset, we train the initial RNN model in an incremental fashion (continuously updating the network parameters at each time step) at least approximately first 10-15% of time steps. In case of periodic time series, we continuous the training that cover at least one complete cycle and sufficiently two-three cycles, i.e., in case of sine series we must continue the learning and train at least two complete non-anomalous sine wave cycles, so that LSTM gather sufficient information regarding series that help to correctly detect any significant change based on built historical training. For LSTM based model, the network with 1 hidden layer and various number of LSTMs in hidden layer is used for generating the prediction error. We used learning rate η = 0.01 for all cases. We are utilizing grid search to find the best architecture such as number of LSTM nodes in a hidden layer. Further, we experimented with prediction horizon of l = 1, 5, 10. We choose fixed RNN *history window* (h_w) of length 100 for all experiments.

Therefore, unlike supervised anomaly detection technique proposed by [8, 26] where they searched the best threshold for each sub-benchmark time series based on the validation data. Rather, in our proposed novel unsupervised online anomaly detection algorithm, we get an automatic initialization of threshold τ after burning period of LSTM training for 10-15% data points of normal time series. Once sufficient training is done for considered time series than proposed model starts the anomaly detection and update the threshold in an online way that so that it will adapt the changes in time series. As discuss in earlier chapter 3, the model evaluation, threshold initialization and its updates are directly related with the good choice of significance level α (alpha). Usually, each time series has its own characteristics, and finding a generic value of significance level α (alpha) which works for all of the time series is not an easy task and may not work appropriately. Thus, to keep every process in-line with the online methodology, the value of significance level α (alpha) is also automatically selected by the model which is described in section 3.2.2(B) of earlier chapter.

In considered case of change point detection we introduce a *change detection counter* in online anomaly detection methodology, whereas, if the model continuously estimates small probabilities, then the counter value decides and consider it as change point/set of change points and start updating till the model parameters are adapted to predict the new normal behaviour well and the estimated probabilities increases accordingly. It is important that the *change detection counter* must be at least greater than twice of prediction horizon l; such that *change detection counter* $> 2 * l$ and in case of periodic time series *change detection counter* must at least be the quarter of one period/cycle of series. For example, the sine signal shown in figure 5.3 (e) where the sine signal has 333 values per cycle/period, we set *change detection counter = 100.* To keep in line with online methodology the *change detection counter* is selected before starting the training and anomaly detection from the first 15-20% of data considered for training.

5.5 Experimental Results for Change Point Detection using OLACD

This section gives comprehensive results set produced by each experiment and provides analysis and conclusion directly related to the accumulation of each analysis. The results of implemented OLACD experiments are investigated for analysis; thus, evaluated in order to produce an evaluation conclusion (section 5.6). The performance evaluation metrics described in chapter 3 are adopted for the experimental validation that aim to prove the effectiveness and efficiency of OLACD algorithm.

5.5.1 Online LSTM Training and Prediction Error

Considering the experimental setup discussed in section 5.4 and the parameter setting given the table 5.1. Figure 5.4(b), 5.5 (b), 5.6 (b), 5.7(b) and 5.7(f) shows the observational values of time series; figure 5.4 (c), 5.5 (c), 5.6 (c), 5.7 (c) and 5.7 (g) shows the prediction error for 1 step-ahead predictions; figure 5.4 (d), 5.5 (d), 5.6 (d), 5.7 (d) and 5.7 (h) demonstrate the prediction error for 5 step-ahead predictions; figure 5.4 (e), 5.5 (e), 5.6 (e), 5.7 (e) and 5.7 (i) reveal the prediction error for 10 step-ahead predictions for NAB Sudden Change dataset, NAB Continuous Change dataset, NAB Incremental Change datasets, Sine Frequency Change and Sine Amplitude change data sets respectively.

The figures 5.4(c-e), 5.5(c-e) 5.6(c-e), 5.7(c-e) 5.7(g-i) shows that there is higher prediction error for anomalous region of time series as compare with the normal regions and this predictions error is further used in anomaly and change-point detection model in incremental manner.

5.5.2 Online Change Point Detection Results

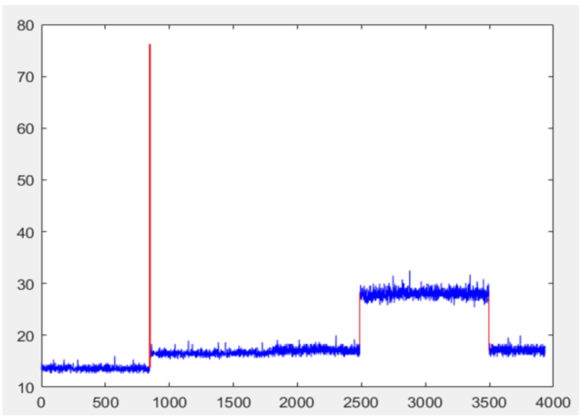
The lists of experiments are described in table 5.2. Column three of table 5.2 gives the architecture for each experiment, i.e., 1-25-5 shows that there are three-layer recurrent neural network architecture that includes one input, one hidden and one output layer; it also shows that here is one neuron/node at input layer, hidden layer consist of 25 neurons (e.g., LSTM Cells) and there are 5 neurons/ nodes at output layer. Further, we experimented with prediction length l = 1, 5, 10 such that the different models trained for each value of l. We choose fixed window length w = 100 for all experiments.

TABLE 5.2: DEPICTION OFA LIST OF EXPERIMENTS IMPLEMENTED FOR CHANGE-POINT DETECTION USING OLACD

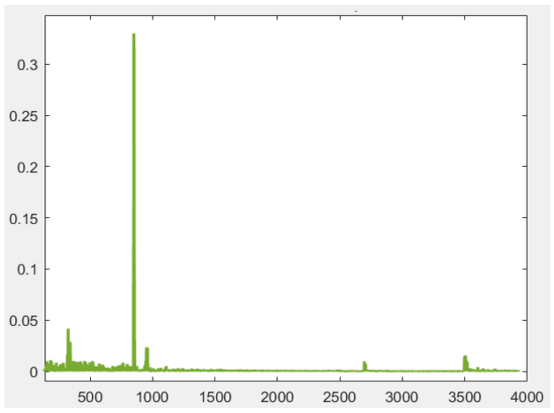
EXP#	MAPPING INPUT TIME SERIES	NETWORK STRUCTURE
1		1-25-1
2	Sudden Change	1-25-5
3		1-25-10
4		1-25-1
5	Continuous Change	1-25-5
6		1-25-10
4		1-25-1
5	Incremental Change	1-25-5
6		1-25-10
7		1-30-1
8	Sudden frequency change (Sine Wave)	1-25-5
9		1-40-10
10		1-30-1
11	Sudden amplitude change (Sine Wave)	1-25-5
12		1-25-10

Normal Time Series
Anomalous Time Series
Error for 1 Step-a-head Prediction
Error for 5 Step-a-head Predictions
Error for 10 Step-a-head Predictions

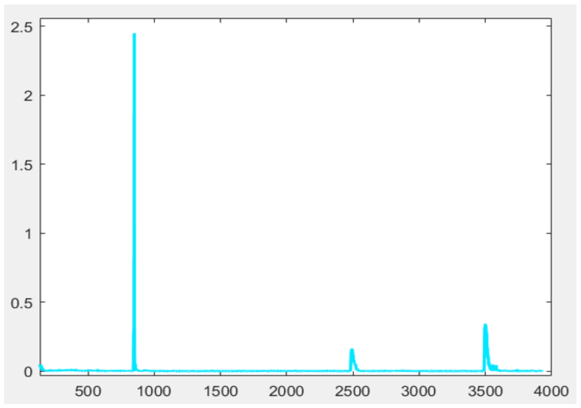
(a) Legends of subfigures (b)-(e)



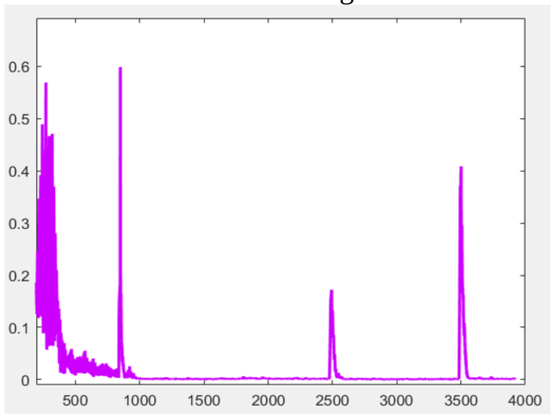
(b) NAB Sudden Change



(c) 1 Step-a-head prediction error for NAB Sudden Change



(d) 5 Step-a-head prediction error for NAB Sudden Change



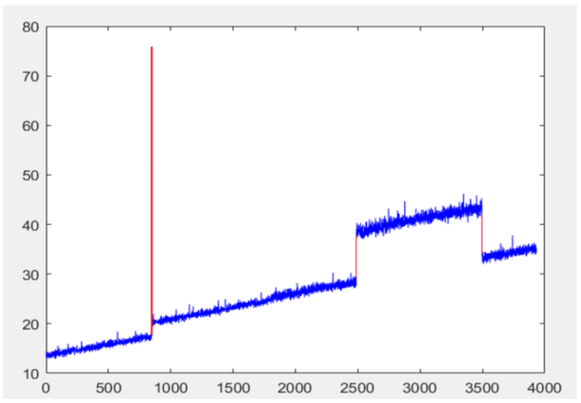
(e) 10 Step-a-head prediction error for NAB Sudden Change

Figure 5.4: Results for Online LSTM Prediction for NAB Sudden Change datasets.

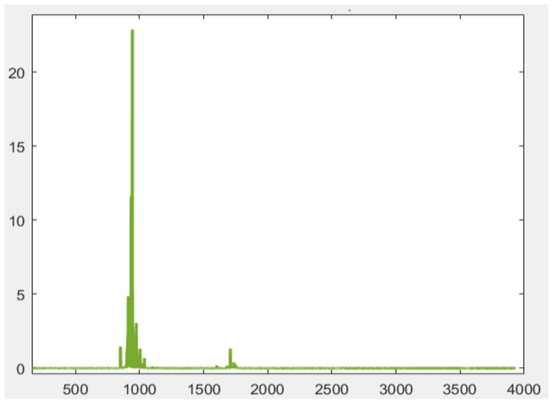
5.4 (a) give the legends of subfigures. In (b)-(e) time steps are given on x-axis. In (b)-(e) y-axis shows the observational values. Figure 5.4(a) present the time series, 5.4 (b) demonstrates the prediction error for 1-step ahead and shown in green colour, figure 5.4(c), shows the prediction error for 5-step ahead prediction and presented in cyan colour while 5.4(e) represents the prediction error for 10-step ahead which shown in magenta colour. Respected prediction error values are given in (b)-(e). Red colour in figure 5.4(a) represents the anomalous points in data stream and blue colures represents the normal points in time series. There are total 5 anomalous data points in NAB Sudden Change datasets at three different locations that includes 2 single change point and 3 short-term anomalies. 3 anomalous data points are considered as outlier at first location in figure 5.4(b) while subsequent single anomalous points are considered as change-point. Data instances of time series are given on x-axis, while prediction error values are given on y-axis in figure (c-e). The high peaks of respected *l* step-a-head prediction error in (c)-(e) exhibit the presence of anomalies/outlier in data stream given in (b).

Normal Time Series
Anomalous Time Series
Error for 1 Step-a-head Prediction
Error for 5 Step-a-head Predictions
Error for 10 Step-a-head Predictions

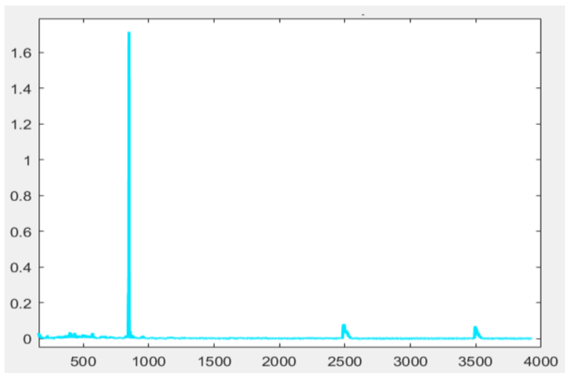
(a) Legends of subfigures (b)-(e)



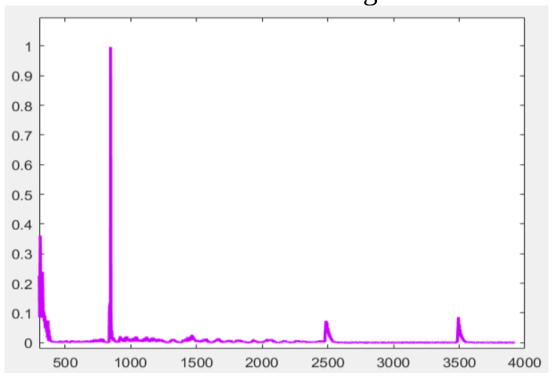
(b) NAB Continuous Change



(c) 1 Step-a-head prediction error for NAB Continuous Change



(d) 5 Step-a-head prediction error for NAB Continuous Change



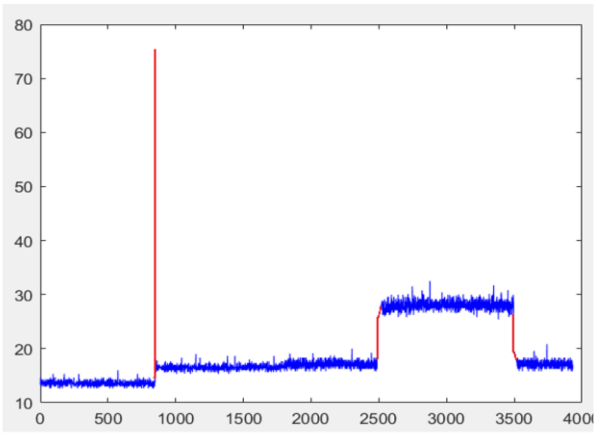
(e) 10 Step-a-head prediction error for NAB Continuous Change

Figure 5.5: Results for Online LSTM Prediction for NAB Continuous Change datasets.

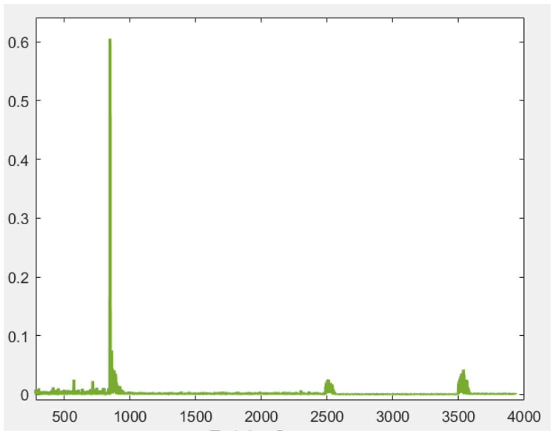
5.5 (a) give the legends of subfigures. In (b)-(e) time steps are given on x-axis. In (b)-(e) y-axis shows the observational values. Figure 5.5(a) present the time series, 5.5(b) demonstrates the prediction error for 1-step ahead and shown in green colour, figure 5.5(c), shows the prediction error for 5-step ahead prediction and presented in cyan colour while 5.5(e) represents the prediction error for 10-step ahead which shown in magenta colour. Respected prediction error values are given in (b)-(e). Red colour in figure 5.5(a) represents the anomalous points in data stream and blue colures represents the normal points in time series. There are total 5 anomalous data points in NAB continuous Change datasets at three different locations that includes 2 single change point and 3 short-term anomalies. 3 anomalous data points are considered as outlier at first location in figure 5.5(b) while subsequent single anomalous points are considered as change-point 5.5(b). Data instances of time series are given on x-axis, while prediction error values are given on y-axis in figure (c-e). The high peaks of respected *l* step-a-head prediction error in 5.5(c)-(e) exhibit the presence of anomalies/outlier in data stream given in 5.5(b).

Normal Time Series
Anomalous Time Series
Error for 1 Step-a-head Prediction
Error for 5 Step-a-head Predictions
Error for 10 Step-a-head Predictions

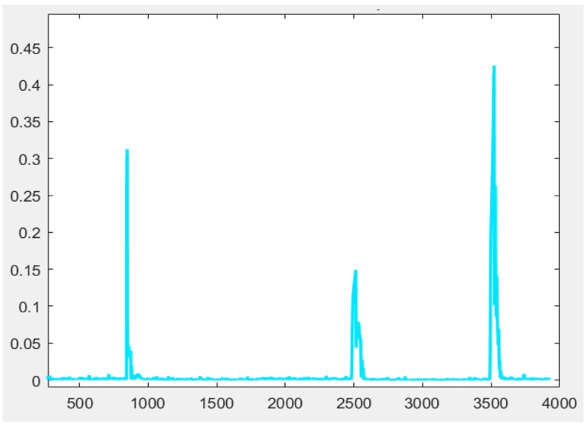
(a) Legends of subfigures (b)-(e)



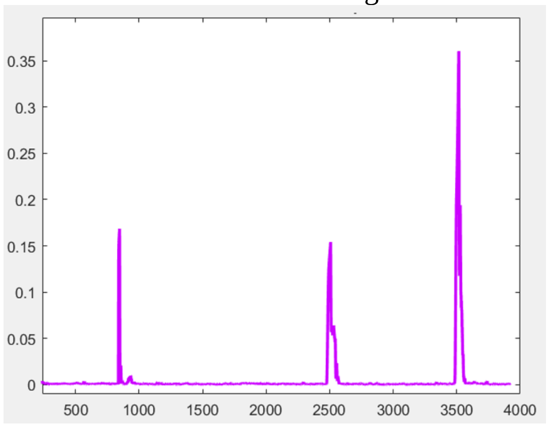
(b) NAB Incremental Change



(c) 1 Step-a-head prediction error for NAB Incremental Change



(d) 5 Step-a-head prediction error for NAB Incremental Change



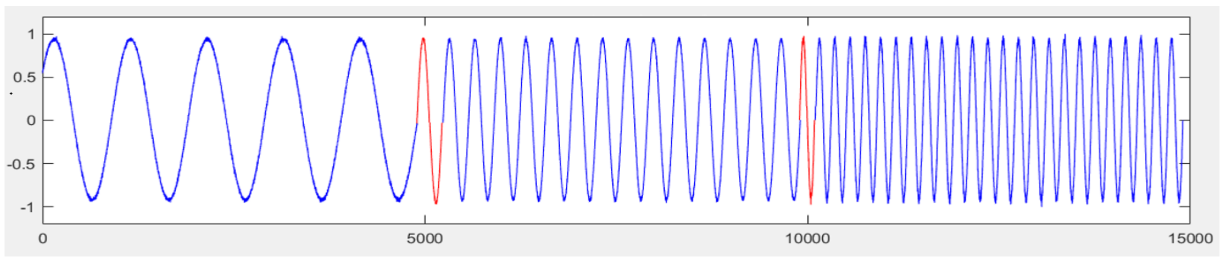
(e) 10 Step-a-head prediction error for NAB Incremental Change

Figure 5.6: Results for Online LSTM Prediction for NAB Incremental Change datasets.

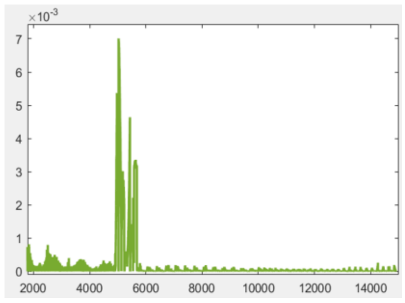
5.6(a) give the legends of subfigures. In (b)-(e) time steps are given on x-axis. In (b)-(e) y-axis shows the observational values. Figure 5.6(a) present the time series, 5.6(b) demonstrates the prediction error for 1-step ahead and shown in green colour, figure 5.6(c), shows the prediction error for 5-step ahead prediction and presented in cyan colour while 5.6(e) represents the prediction error for 10-step ahead which shown in magenta colour. Respected prediction error values are given in (b)-(e). Red colour in figure 5.6(a) represents the anomalous points in data stream and blue colures represents the normal points in time series. There are total 66 anomalous data points in NAB Incremental Change datasets at three different locations that includes 2 multiple sets of change point and 3 short-term anomalies. 3 anomalous data points are considered as outlier at first location in figure 5.6(b) while subsequent multiple anomalous points are considered as change-point. 5.6(b). Data instances of time series are given on x-axis in figure (c-e), while prediction error values are given on y-axis. The high peaks of respected *l* step-a-head prediction error in5.6(c)-(e) exhibit the presence of anomalies/outlier in data stream given in 5.6(b).

Normal Time Series
Anomalous Time Series
Error for 1 Step-a-head Prediction
Error for 5 Step-a-head Predictions
Error for 10 Step-a-head Predictions

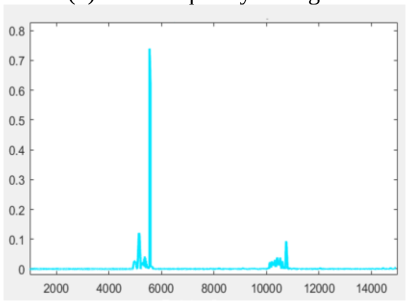
(a) Legends of subfigures (b)-(i)



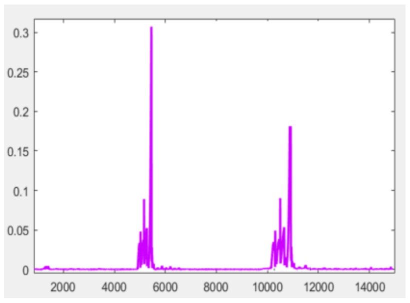
(b) Sine Frequency Change



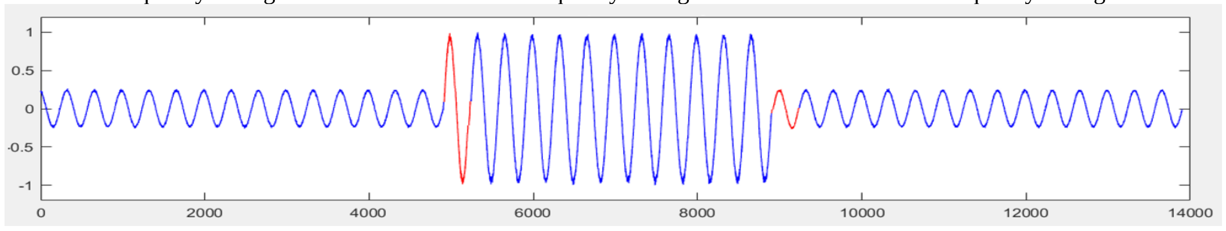
(c) 1 Step-a-head Prediction for Sine Frequency Change



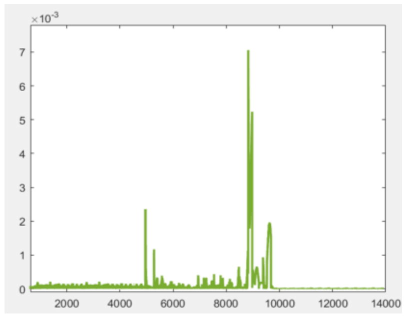
(d) 5 Step-a-head Prediction for Sine Frequency Change



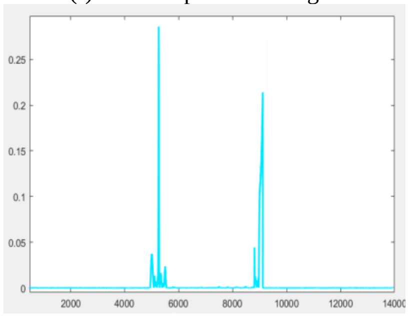
(e) 10 Step-a-head Prediction for S Frequency Change



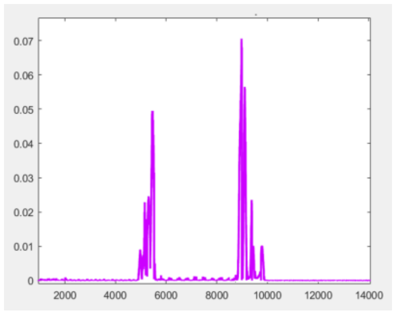
(f) Sine Amplitude Change



(g) 1 Step-a-head Prediction for Sine Amplitude Change



(h) 5 Step-a-head Prediction for Sine Amplitude Change



(i) 10 Step-a-head Prediction for S Amplitude Change

Figure 5.7: Results for Online LSTM Prediction for Synthetic datasets.

5.7 (a) give the legends of subfigures. In (b)-(i) time steps are given on x-axis. In (b) and(f) y-axis shows the observational values, where red colour represents the anomalous series and blue colures represents the normal time series. Respected prediction error values are given in (c), (d), (e), (g), (h) and (i). Peaks in (c)-(e) and (g)-(i) shows the higher values exhibit the presence of change. 5.5 (a) give the legends of subfigures. In (b)-(i) time steps are given on x-axis. In (b) and(f) y-axis shows the observational values, where red colour represents the anomalous series and blue colures represents the normal time series. Data instances of time series are given on x-axis, while prediction error values is given on y-axis in (c-e, g-i). Respected prediction error values are given in (c), (d), (e), (g), (h) and (i). Peaks in (c)-(e) and (g)-(i) shows the higher values exhibit the presence of change.

TABLE 5.3: OLACD-LSTM EXPERIMENTAL RESULTS AT MODEL'S INITIALIZED α FOR UNIVARIATE TIME SEREIS WITH ANOMALIES AND CHANGE-POINT

			Performance Measures				
# of Predictions	data points	alpha	TPR	TNR	FPR	FNR	[TP,FN,FP,TN]
Sudden Change							
1	3433	0.0025	0.4000	0.9912	0.0088	0.6000	[2,3,40,3388]
5	3429	8.3370e-12	1	0.9690	0.0310	0	[5,0,106,3318]
10	3424	1.4935e-13	1	0.9763	0.0237	0	[5,0,81,3338]
Continuous Change							
1	3433	0.0050	0.8667	0.9947	0.00528	0.1333	[2,3,30,3398]
5	3429	8.7639e-10	1	0.9871	0.0129	0	[5,0,44,3380]
10	3424	2.0864e-24	1	0.9728	0.0272	0	[5,0,93,3326]
Incremental Change							
1	3433	0.0025	0.8485	0.9709	0.0291	0.1515	[56,10,98,3269]
5	3429	3.6083e-14	0.9697	0.9741	0.0259	0.0303	[64,2,87,3276]
10	3424	1.0361e-28	0.9394	0.9797	0.0203	0.0606	[62,4,68,3290]
Sudden frequency change (Sine Wave)							
1	13900	3.1844e-05	0.4985	0.9481	0.0518	0.5015	[332,334,634,11600]
5	13896	1.8918e-15	0.7507	0.9217	0.0782	0.2492	[500,166,957,11273]
10	13891	1.9994e-25	0.9759	0.9169	0.0830	0.0240	[650,16,1015,11210]
Sudden amplitude change (Sine Wave)							
1	12900	7.5697e-05	0.5674	0.9622	0.0377	0.4325	[303,231,505,12861]
5	12896	2.5392e-16	0.9812	0.8643	0.1356	0.0187	[524,10,813,11549]
10	12891	1.9801e-34	0.9288	0.9645	0.0354	0.0711	[496,38,474,12883]

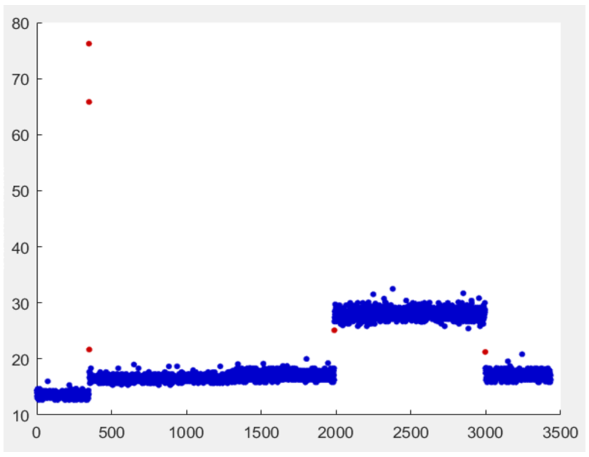
Table 5.3 give the OLACD results produced by online Anomaly Detection model for each experiment where alpha is initialized by model. The data points of each data set that considered for change detection is shown in column two of table 5.3. True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR) and False Negative Rate (FNR) for each experiment are given in subsequent column. The last column of table 5.3 gives the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) tagged by OLACD model in each experiment for respective benchmark univariate time series.

Anomaly and change-point results for all five data sets for each *l*-step ahead prediction are plotted in figures 5.8, 5.9 and 5.10 gives the results for data sets collected and derived from NAB [6], NAB is real benchmark data sets universally adapted to verify the performance of novel anomaly/change point detection models [6] while figure 5.11 and 5.12 give the results for synthetic sine wave dataset. The model is able to detect 2 out 5 anomalous points for NAB Sudden change in case of 1-step ahead prediction shown on 5.8(c), correctly detect 2 out of 5 anomalous points in case of NAB continuous change shown in 5.9(c) and detect 56 out of 66 anomalous points including change-points in case of NAB incremental change dataset. shown in 5.10(c). Therefore, we could say that the 1-step ahead prediction is the case of delayed detection. Whereas in 5-steps ahead and 10 steps ahead prediction for all five datasets, the proposed OLACD is successfully able to detect all anomalous points at model's initialize alpha (α). So, the True Positive Rate for four of datasets in case of multi-step prediction is 1 and small False Positive Rate (FPR), which verify that proposed OLACD model is robust to outlier and change-points in multi-step ahead settings and capable to detect the outlier and change-points in streaming data. In all sub-figures of figure 5.8, 5.9, 5.10, 5.11 and 5.12, the blue colours dots represent the true normal predictions, red colour dots show the true anomalous prediction, yellow colour shows the false alarm/ false anomalies predictions, while green colour shows the false normal predictions. OLACD is initializing alpha (α) using equation 3.18 which would remain fix during complete test run of one experiment. OLACD model only initialize the value of alpha (α) burning period of LSTM training. The propose OLACD model initialize and 0.0025, 8.3370e-12 and 1.4935e-13 as alpha for 1,5,10 step ahead prediction for NAB Sudden drift dataset. Similarly, 0.0050, 8.7639e-10 and 2.0864e-24 is initialized by model for NAB continuous change dataset for 1,5,10 step ahead prediction respectively. In the same way 0.0025, 3.6083e-14 and 1.0361e-28 respectively alpha are initialized by model NAB incremental change series. The alpha values 3.1844e-05, 1.8918e-15 and 1.9994e-25 are computed for Sine frequency change datasets while 7.5697e-05, 2.5392e-16 and 1.9801e-34 are computed for sine amplitude change dataset respectively.

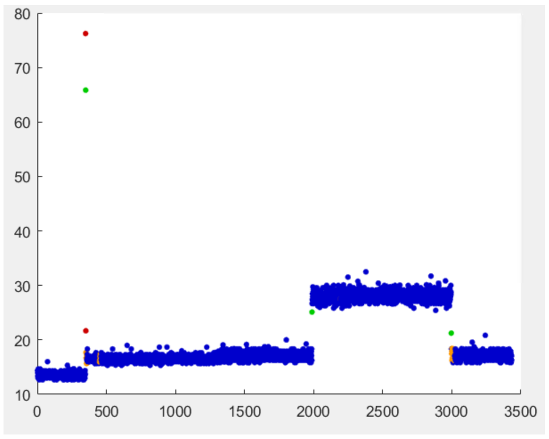
It is revealed that the OLACD predicts the number of false anomalies while detecting the true anomalies and it is shown in table 5.2 and figure 5.3 that algorithm is capable to adapt the short-term anomalies in such as way these transitory anomalies do not lead to a radical change in the model and model is capable to adapt the change quickly. It is also demonstrated that the model learn with single step-a-head predictions is more reluctant and less responsive to anomalies and change-points as compared to multi step-a-head predictions. For instance, for all five datasets the model trained with multi-step-ahead predictions predicts all anomalies and change-points while model trained with single step-a-head prediction is less responsive and reluctant to early detection of anomalous regions/points. Whereas in almost all case the model trained with multi-step-ahead predictions are producing more false alarm as compared to single step-a-head prediction.

TP: True Anomaly Predictions
FP: False Anomaly Predictions
TN: True Normal Predictions
FN: False Normal Predictions

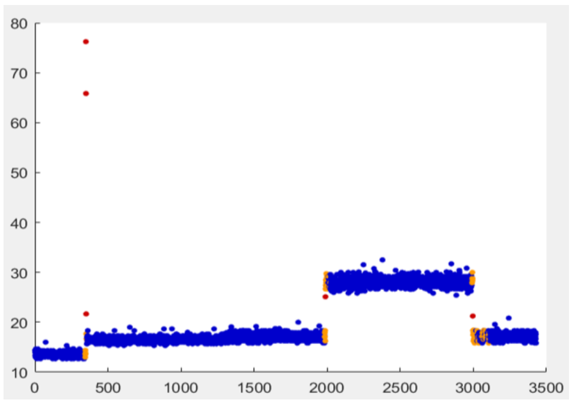
(a) Legends of subfigures (b)-(e)



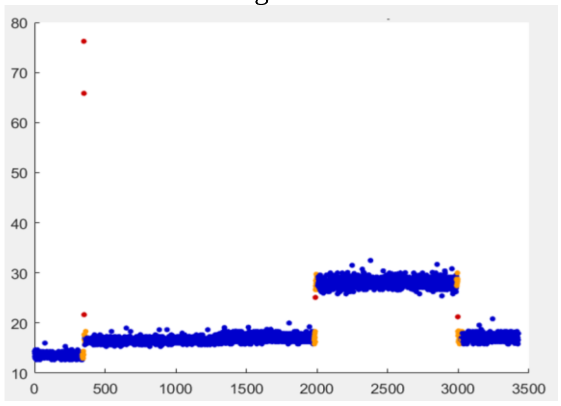
(b) NAB Sudden Change: Ground Truth



(c) Anomalies/Change Point Detection for 1 Step-a-head Prediction for NAB Sudden Change dataset



(d) Anomalies/Change Point Detection for 5 Step-a-head Prediction for NAB Sudden Change dataset



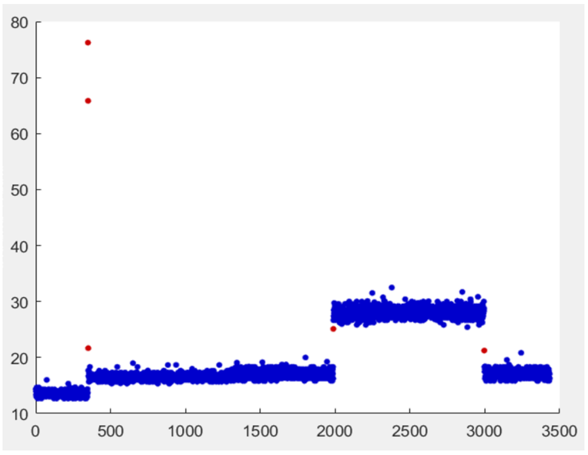
(e) Anomalies/Change Point Detection for 10 Step-a-head Prediction for NAB Sudden Change dataset

Figure 5.8: AD-LSTM Results for NAB Sudden Change dataset Datasets at AD-Model's Initialized α.

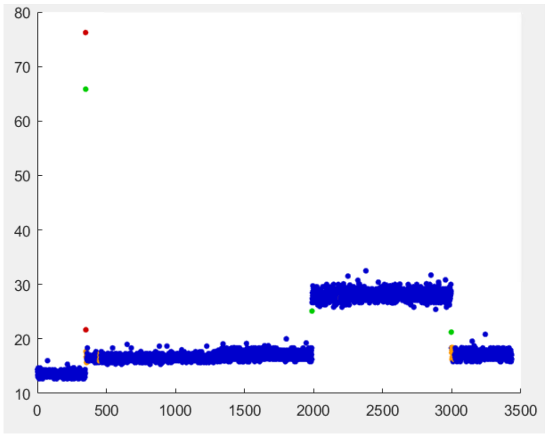
5.8(a) gives the legends of subfigures. Ground truth observations are shown in 5.8 (b), where all blue are normal point and red are anomalous/change points. 5.8 (c) gives the anomaly detection for 1-step ahead prediction, 5.8 (d) shows the anomaly detection for 5-step ahead prediction and 5.8 (e) gives the anomaly detection for 10-step ahead prediction. The colour codes for true anomalous observations are red, true normal are blue, false anomalous are yellow and false normal observations are green. There are total five anomalous points (3 outlier and 2 change-points) at three different locations for NAB Sudden Change dataset and shown in sub-figure 5.8(b). OLACD is able to detect the 2 out of 3 and will not be able to any detect change-points in 1-step ahead prediction demonstrated in sub-plot 5.8(b) while able to detect all anomalous points including outliers and change-points in 5-steps ahead and 10-step ahead prediction represented in 5.8(d) and 5.8(e).

TP: True Anomaly Predictions
FP: False Anomaly Predictions
TN: True Normal Predictions
FN: False Normal Predictions

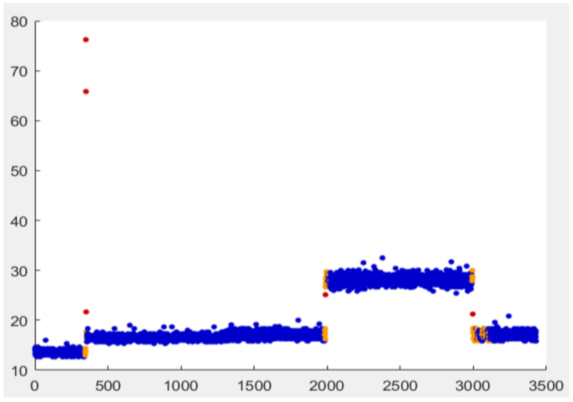
(b) Legends of subfigures (b)-(e)



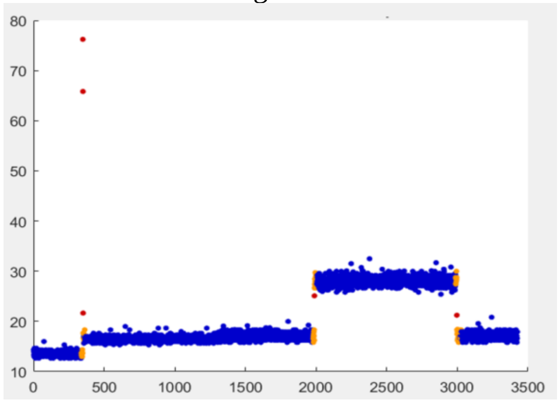
(b) NAB Sudden Change: Ground Truth



(c) Anomalies/Change Point Detection for 1 Step-a-head Prediction for NAB Sudden Change dataset



(d) Anomalies/Change Point Detection for 5 Step-a-head Prediction for NAB Sudden Change dataset



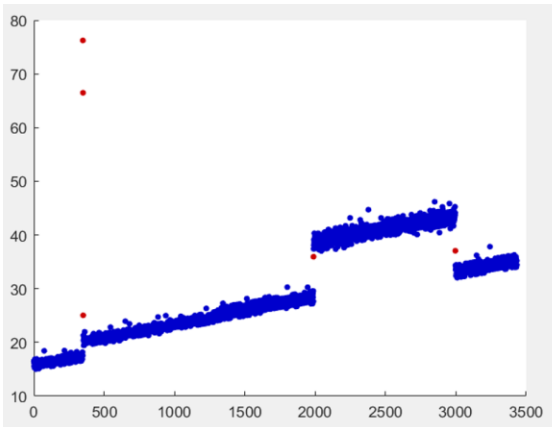
(e) Anomalies/Change Point Detection for 10 Step-a-head Prediction for NAB Sudden Change dataset

Figure 5.9: AD-LSTM Results for NAB Sudden Change dataset Datasets at AD-Model's Initialized α.

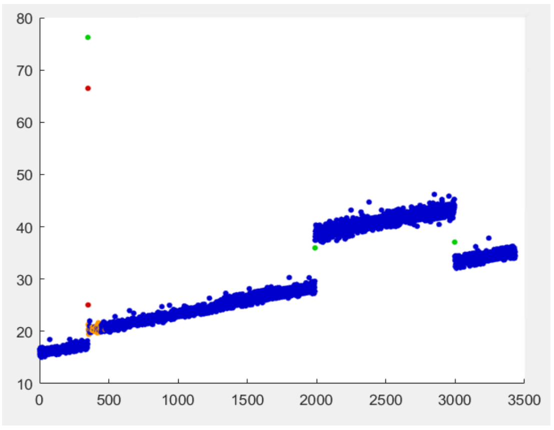
5.8(a) gives the legends of subfigures. Ground truth observations are shown in 5.8 (b), where all blue are normal point and red are anomalous/change points. 5.8 (c) gives the anomaly detection for 1-step ahead prediction, 5.8 (d) shows the anomaly detection for 5-step ahead prediction and 5.8 (e) gives the anomaly detection for 10-step ahead prediction. The colour codes for true anomalous observations are red, true normal are blue, false anomalous are yellow and false normal observations are green. There are total five anomalous points (3 outlier and 2 change-points) at three different locations for NAB Sudden Change dataset and shown in sub-figure 5.8(b). OLACD is able to detect the 2 out of 3 and will not be able to any detect change-points in 1-step ahead prediction demonstrated in sub-plot 5.8(b) while able to detect all anomalous points including outliers and change-points in 5-steps ahead and 10-step ahead prediction represented in 5.8(d) and 5.8(e).

TP: True Anomaly Predictions
FP: False Anomaly Predictions
TN: True Normal Predictions
FN: False Normal Predictions

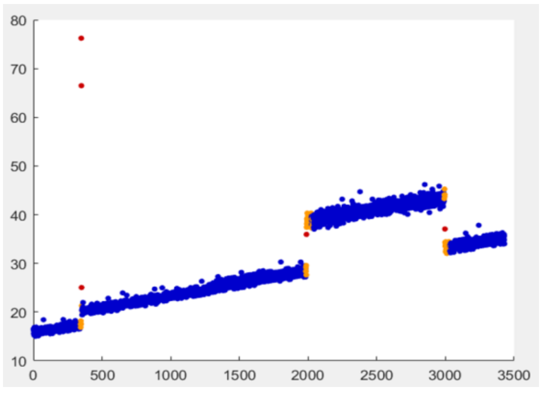
(a) Legends of subfigures (b)-(e)



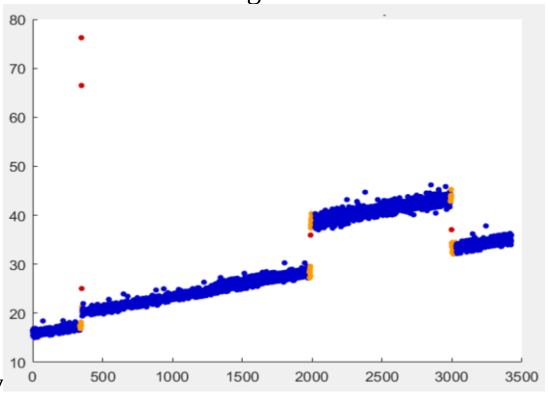
(b) NAB Continuous Chang: Ground Truth



(c) Anomalies/Change Point Detection for 1 Step-a-head Prediction for NAB Continuous Chang dataset



v



(d) Anomalies/Change Point Detection for 5 Step-a-head Prediction for NAB Continuous Chang dataset

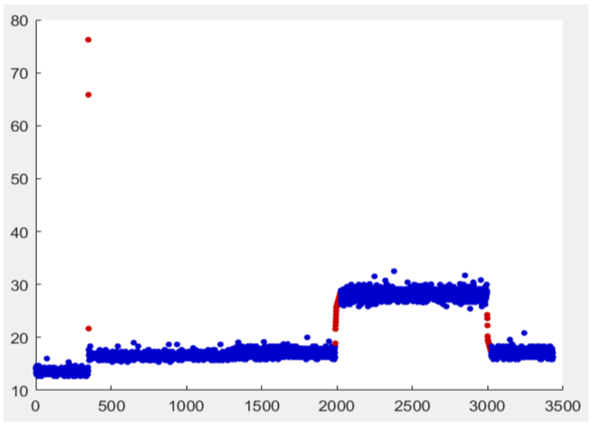
(e) Anomalies/Change Point Detection for 10 Step-a-head Prediction for NAB Continuous Chang dataset

Figure 5.10: AD-LSTM Results for NAB Continuous Chang Dataset at AD-Model's Initialized α.

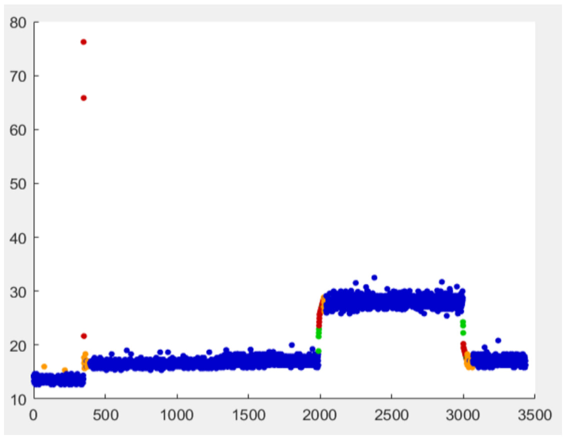
5.9(a) gives the legends of subfigures. Ground truth observations are shown in 5.9 (b), where all blue are normal point and red are anomalous/change points. 5.9 (c) gives the anomaly detection for 1-step ahead prediction, 5.9 (d) shows the anomaly detection for 5-step ahead prediction and 5.9 (e) gives the anomaly detection for 10-step ahead prediction. The colour codes for true anomalous observations are red, true normal are blue, false anomalous are yellow and false normal observations are green. There are total five anomalous points (3 outlier and 2 change-points) at three different locations for NAB Continuous Change dataset and shown in sub-figure 5.9(b). OLACD is able to detect the 2 out of 3 and will not be able to any detect change-points in 1-step ahead prediction demonstrated in sub-plot 5.9(b) while able to detect all anomalous points including outliers and change-points in 5-steps ahead and 10-step ahead prediction represented in 5.9(d) and 5.9(e).

TP: True Anomaly Predictions
FP: False Anomaly Predictions
TN: True Normal Predictions
FN: False Normal Predictions

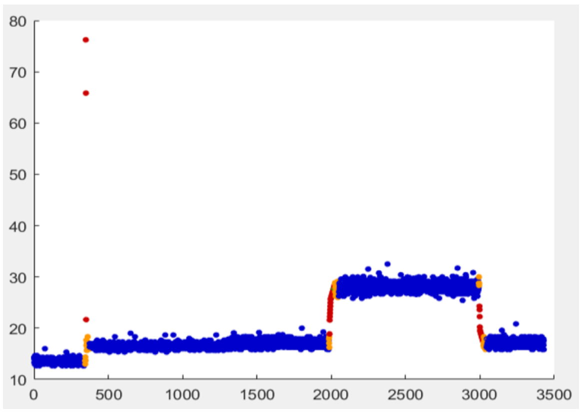
(a) Legends of subfigures (b)-(e)



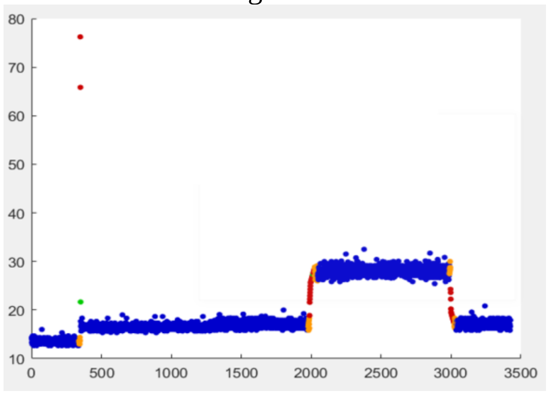
(b) NAB Incremental Change dataset: Ground Truth



(c) Anomalies/Change Point Detection for 1 Step-a-head Prediction for NAB Incremental Change dataset



(d) Anomalies/Change Point Detection for 5 Step-a-head Prediction for NAB Incremental Change dataset



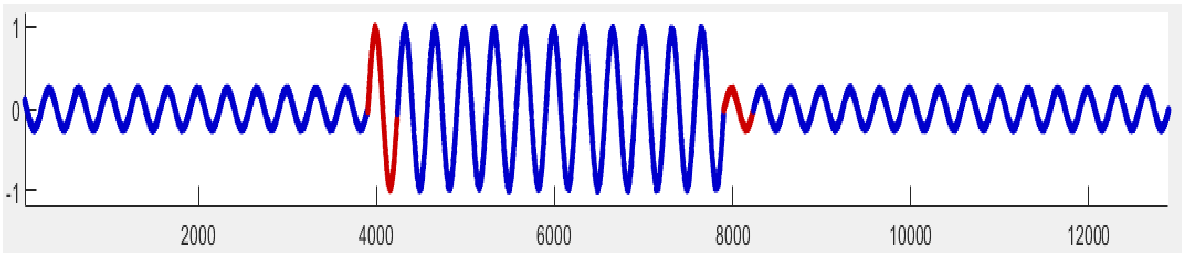
(e) Anomalies/Change Point Detection for 10 Step-a-head Prediction for NAB Incremental Change dataset

Figure 5.11: AD-LSTM Results NAB Incremental Change Datasets at AD-Model's Initialized α.

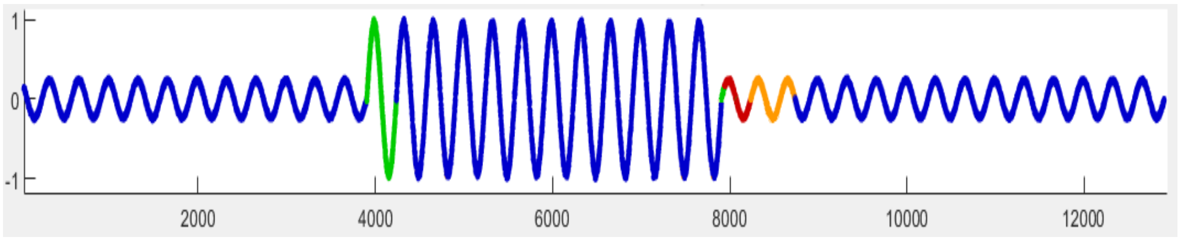
5.10(a) gives the legends of subfigures. Ground truth observations are shown in 5.10 (b), where all blue are normal point and red are anomalous/change points. 5.10 (c) gives the anomaly detection for 1-step ahead prediction, 5.10 (d) shows the anomaly detection for 5-step ahead prediction and 5.10 (e) gives the anomaly detection for 10-step ahead prediction. The colour codes for true anomalous observations are red, true normal are blue, false anomalous are yellow and false normal observations are green. There are total 66 anomalous points that includes 3 anomalies and 2 change points that comprises the 2 set of 63 observation (3 anomalous data points are considers as outlier at first location in figure 5.10(b). OLACD is able to detect the 56 out of 66 anomalies and change-points in 1-step ahead prediction demonstrated in sub-plot 5.10(b) while able to detect all anomalous points including outliers and change-points in 5-steps ahead and 10-step ahead prediction represented in 5.10(d) and 5.10(e).

TP: True Anomaly Predictions
FP: False Anomaly Predictions
TN: True Normal Predictions
FN: False Normal Predictions

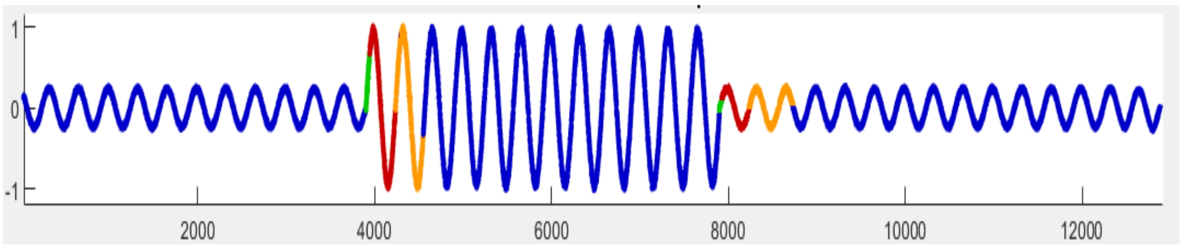
(a) Legends of subfigures (b)-(e)



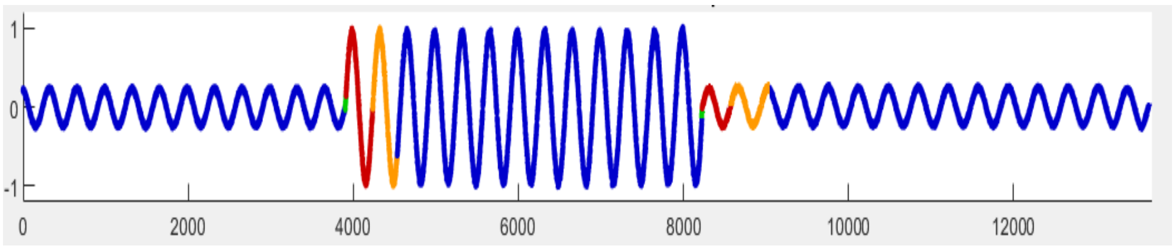
(b) Sine Amplitude Change: Ground Truth



(c) Anomalies/Change Point Detection for 1 Step-a-head Prediction for Sine Amplitude Change dataset



(d) Anomalies/Change Point Detection for 5 Step-a-head Prediction for Sine Amplitude Change dataset



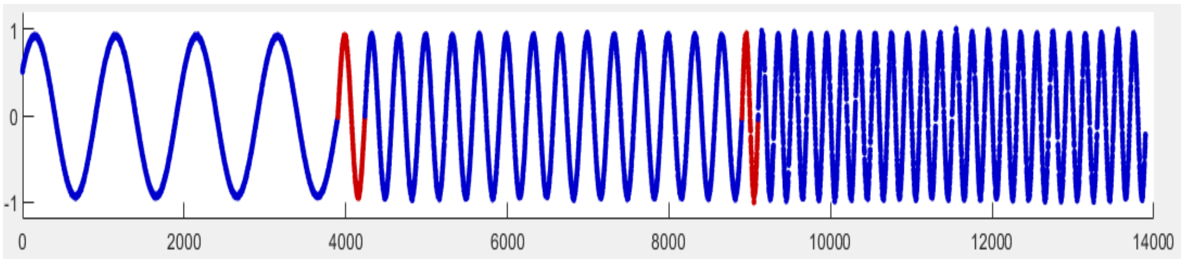
(e) Anomalies/Change Point Detection for 10 Step-a-head Prediction for Sine Amplitude Change dataset

Figure 5.12: AD-LSTM Results for Sudden Amplitude Change Sine Signal Datasets at AD-Model's Initialized α.

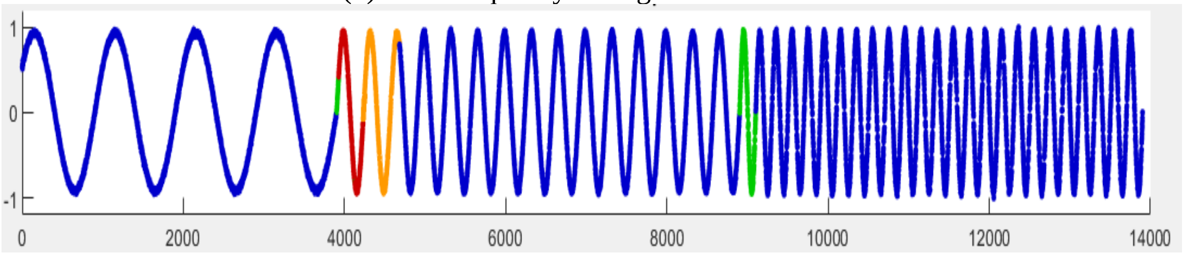
5.11 (a) gives the legends of subfigures. Ground truth observations are shown in 5.11 (b) where all blue are normal point and red are anomalous/change points. 5.11 (c) gives the change-points detection for 1-step ahead prediction, 5.11 (d) shows the change-points detection for 5-step ahead prediction and 5.11 (e) gives the change-points detection for 10-step ahead prediction. The colour codes for true anomalous observations are red, true normal are blue, false anomalous are yellow and false normal observations are green.

TP: True Anomaly Predictions
FP: False Anomaly Predictions
TN: True Normal Predictions
FN: False Normal Predictions

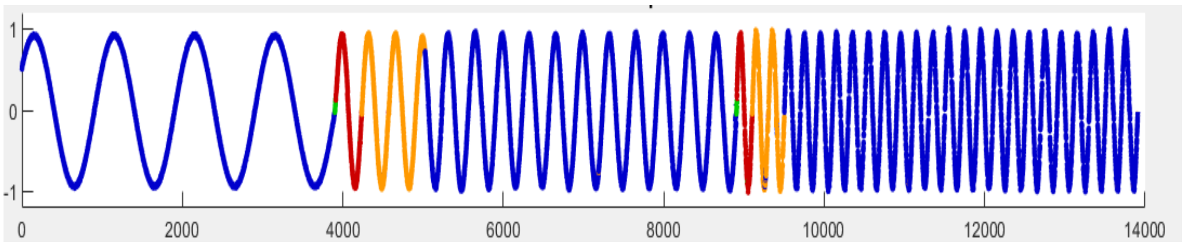
(a) Legends of subfigures (b)-(e)



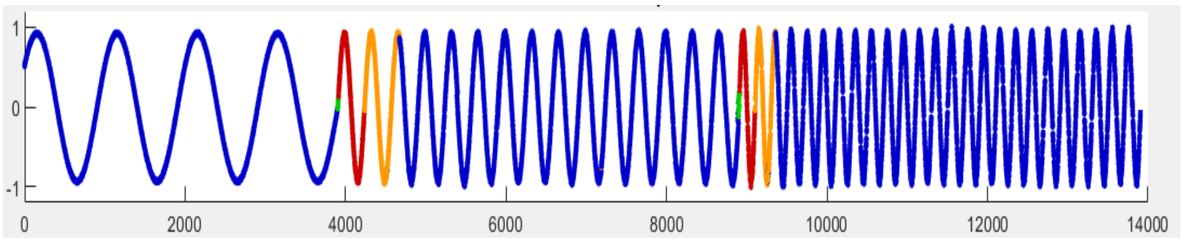
(b) Sine Frequency Change: Ground Truth



(c) Anomalies/Change Point Detection for 1 Step-a-head Prediction for Sine Frequency Change data



(d) Anomalies/Change Point Detection for 5 Step-a-head Prediction for Sine Frequency Change data



(e) Anomalies/Change Point Detection for 10 Step-a-head Prediction for Sine Frequency Change data

Figure 5.13: AD-LSTM Results for Sudden Frequency Change Sine Signal Datasets at AD-Model's Initialized α.

5.12 (a) gives the legends of subfigures. Ground truth observations are shown in 5.12 (b) where all blue are normal point and red are anomalous/change points. 5.12 (c) give the change-points detection for 1-step ahead prediction, 5.12 (d) shows the change-points detection for 5-step ahead prediction and 5.12 (e) gives the change-points detection for 10-step ahead prediction. The colour codes for true anomalous observations are red, true normal are blue, false anomalous are yellow and false normal observations are green.

5.5.3 Online Change Detection Model's Performance Metrics Results

To evaluate the change detection methods on different time series, we use the standard metrics of precision, recall, F-score accuracy, G-mean and Area Under Curve (AUC). Table 4.4 summarise the performance metrics results of online anomaly detection model in five data sets.

TABLE 5.4: PERFORMANCE METRICES FOR UNIVARIATE TIME SEREIS WITH TANSITORY ANOMALIES AND CHANGE-POINTS

Performance Measures												
# of Pred.	**Data points**	**Alpha (α)**	**TP**	**FN**	**FP**	**TN**	**Precision**	**Recall**	**F-score**	**Acc.**	**G-mean**	**AUC**
Sudden Change												
1	3433	0.0025	2	3	40	3388	0.048	0.400	0.085	0.987	0.629	0.931
5	3429	8.34E-12	5	0	106	3318	0.045	1.000	0.086	0.969	0.984	0.991
10	3424	1.49E-13	5	0	81	3338	0.058	1.000	0.110	0.976	0.988	0.988
Continuous Change												
1	3433	0.005	2	3	30	3398	0.063	0.400	0.108	0.990	0.630	0.890
5	3429	8.76E-10	5	0	44	3380	0.102	1.000	0.185	0.987	0.994	0.994
10	3424	2.09E-24	5	0	93	3326	0.051	1.000	0.097	0.973	0.986	0.992
Incremental Change												
1	3433	0.0025	56	10	98	3269	0.364	0.848	0.509	0.969	0.908	0.938
5	3429	3.61E-14	64	2	87	3276	0.424	0.970	0.590	0.974	0.972	0.989
10	3424	1.04E-28	62	4	68	3290	0.477	0.939	0.633	0.979	0.959	0.991
Sudden frequency change (Sine Wave)												
1	13900	3.18E-05	332	334	634	11600	0.344	0.498	0.407	0.925	0.688	0.765
5	13896	1.89E-15	500	166	957	11273	0.343	0.751	0.471	0.913	0.832	0.922
10	13891	2.00E-25	650	16	1015	11210	0.390	0.976	0.558	0.920	0.946	0.944
Sudden amplitude change (Sine Wave)												
1	12900	7.57E-05	303	231	505	12861	0.375	0.567	0.452	0.947	0.739	0.724
5	12896	2.54E-16	524	10	813	11549	0.392	0.981	0.560	0.936	0.957	0.838
10	12891	1.98E-34	496	38	474	12883	0.511	0.929	0.660	0.963	0.947	0.947

As demonstrated in table 5.4, accuracy is always high when the model detects the greatest number of anomalies, regardless of the greater number of false positive detections. The measure recall is straightforward, since it relates to the proportion of an anomalous class of interest (anomalous points) that was properly identified. As a result, recall is always higher when the model detects a large number of anomalous points. Precision (also known as positive predictive value) is the proportion of relevant instances among those retrieved. As a result, the model provides high precision only when it can anticipate all potential anomalous locations with extremely few false predictions, as shown in table 5.4.

Because F-score is the harmonic mean of a system's precision and recall values, the suggested anomaly detection model performance results in table 5.4 show that F-score is always greater when there is high precision. As demonstrated in table 5.4, accuracy is always high when the model detects the greatest number of anomalies, regardless of the greater number of false positive detections. As a result of the class imbalance, accuracy is not a good metric. Because the ratio of changes to total data is minimal, many studies measure the G-mean, which is often employed as an indication of model effectiveness in the situation of an imbalanced class distribution.

This employs both Sensitivity and Specificity measurements to evaluate the algorithm's effectiveness in terms of the ratio of positive accuracy (Sensitivity) and the ratio of negative accuracy (Specificity) (Specificity). The G-mean measure of the anomaly detection model is always greater when the model predicts the largest number of anomalies/true positives, as shown in table 5.4, and values of G-mean are compromised when the model makes incorrect predictions. The performance in AUC is reported for all data sets in table 5.4, which will be explored further in the next section.

5.5.4 Online Change Detection Model's Performance Results in Term of AUC

For validating our proposed novel anomaly and change-point detection algorithm we are using same data sets and performance measures (e.g., AUC) used by existing online anomaly detection technique proposed by Saurav et al. in 2018 [3] and comparing our results in term of Area Under ROC Curve (AUC), which is given in table 5.3. Although the value of α (*Alpha*) is chosen by AD model by itself but to generate the Receiver Operating Characteristic Curve (ROC, each experiment is repeated with different values of α (Alpha). Whereas different values of α will be find as per equation 4.1, including $\alpha_0 = 0$ and $\alpha_1 = 1$ for $n = 0$.

Finding the different value of significance level α *(Alpha)* is discussed in section 4.4.3 and we perform the experiments repeatedly until we find the no difference in True Positives and False Positives in two/more consecutive run using same parameter specified in table 5.1. Table 5.3

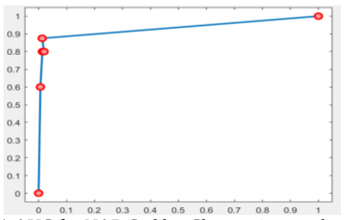
give the experimental results at model's initialized α alpha. Then Area under ROC is calculated using trapezoidal rules.

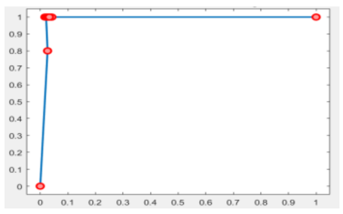
Performance for proposed OLACD is given by *Area under the Receiver Operating Curve (AUC-ROC)*. AUC for each experiment for five considered time series are presented in figure 5.13. Figure 5.13 (b), 5.13 (c), and 5.13 (d) give the AUC for NAB sudden change dataset for 1, 5 and 10 predictions respectively. AUC for NAB continuous change time series for 1, 5 and 10 predictions are given in figure 5.13 (e), 5.13 (f), and 5.13 (g) while AUC for NAB incremental change dataset for 1, 5 and 10 predictions are presented in figure 5.13 (h), 5.13 (i), and 5.13 (j). AUC for synthetic sine wave frequency change time series for 1, 5 and 10 predictions are given in figure 5.13 (k), 5.13 (l), and 5.13 (m) while AUC for sine wave amplitude change dataset for 1, 5 and 10 predictions are presented in figure 5.13 (n), 5.13 (o), and 5.13(p).

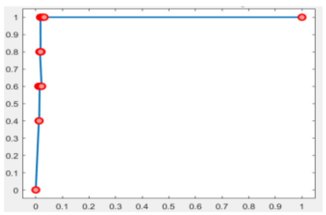
The lower left most corner of ROC in each subfigure of figure 5.13, shows that if α=0, every data point is considered as normal so at α=0, TPR=0 and FPR=0. If we consider *alpha* equal 1 then every data point is considered as anomaly, so at α=1, TPR=1 and FPR=1, which is shown in upper right corner of ROC.

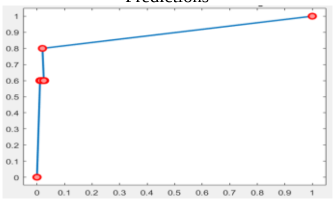
AUC for five of observed change point time series are presented in table 5.3 and demonstrated in figure 5.8. AUC for NAB sudden change dataset is 0.931, 0.991 and 0.988 respectively for 1, 5 and 10 step-a-head predictions shown in 5.13 (b), (c) and (d). AUC for NAB continuous change dataset is 0.890, 0.994 and 0.992 respectively for 1, 5 and 10 step-a-head predictions shown in 5.13 (e), (f) and (g). AUC for NAB incremental change dataset is 0.938, 0.989 and 0.991 respectively for 1, 5 and 10 step-a-head predictions shown in 5.13 (g), (i) and (j). AUC for sine frequency change dataset is 0.765, 0.992 and 0.944 respectively for 1, 5 and 10 step-a-head predictions shown in 5.13 (k), (l) and (m). While AUC for sine amplitude change dataset is 0.724, 0.838 and 0.947 respectively for 1, 5 and 10 step-a-head predictions shown in 5.13 (n), (o) and (p).

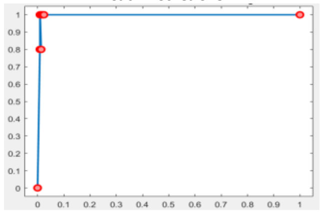
(b) Legend of Subfigures

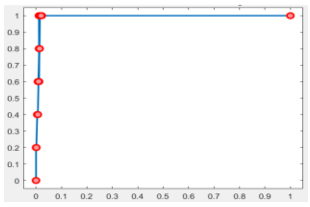

(b) AUC for NAB Sudden Change: 1 step-a-head Predictions

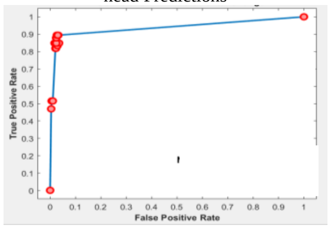

(c) AUC for NAB Sudden Change: 5 step-a-head Predictions


(d) AUC for NAB Sudden Change: 10 step-a-head Predictions

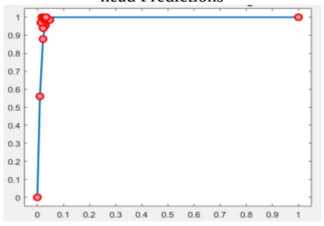

(e) AUC for NAB Continuous Change: 1 step-a-head Predictions

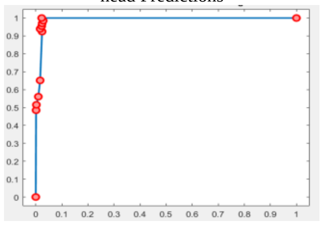

(f) AUC for NAB Continuous Change: 5 step-a-head Predictions

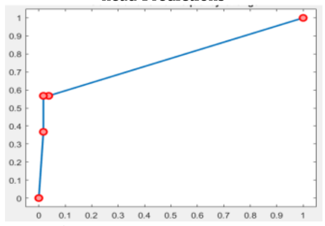

(g) AUC for NAB Continuous Change: 10 step-a-head Predictions

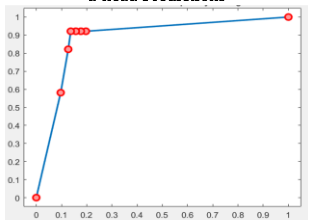


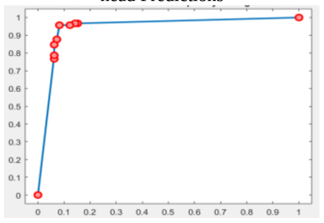
(h) AUC for NAB Incremental Change: 1 step-a-head Predictions

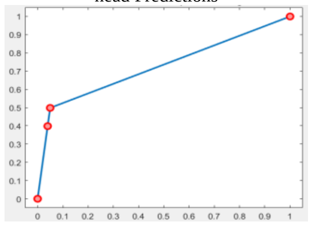

(i) AUC for NAB Incremental Change: 5 step-a-head Predictions

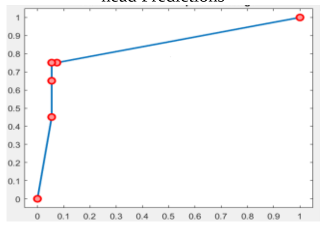

(j) AUC for NAB Incremental Change: 10 step-a-head Predictions


(k) AUC for Sine Frequency Change: 1 step-a-head Predictions


(l) AUC for Sine Frequency Change: 5 step-a-head Predictions


(m) AUC for Sine Frequency Change: 10 step-a-head Predictions


(n) AUC for Sine Amplitude Change: 1 step-a-head Predictions


(o) AUC for Sine Amplitude Change: 5 step-a-head Predictions

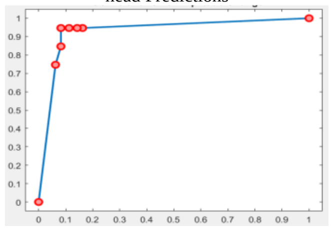

(p) AUC for Sine Amplitude Change: 10 step-a-head Predictions

Figure 5.14: AUC for Proposed Anomaly and Change-Point Detection Algorithm.

False positive rate (FPR) is given on x-axis and True positive rate (TPR) is given on y-axis of all subfigures.5.13(b), (e), (h), (k) and (n) shows the AUC for 1-step Prediction for their respective datasets. 5.13 (c), (f), (i), (l) and (o) are AUC for 5-step Prediction while 5.13 (d), (g), (j), (m) and (p) gives the AUC for 10-step Prediction for their respective time series. Legends for each sub-figure given in 5.13 (a). AUC for NAB sudden change dataset is 0.931, 0.991, 0.988 respectively for 1, 5 and 10 step-a-head predictions shown in 5.13 (b), (c) and (d). AUC for NAB continuous change dataset is 0.890, 0.994, 0.992 respectively for 1, 5 and 10 step-a-head predictions shown in 5.13 (e), (f) and (g). AUC for NAB incremental change dataset is 0.938, 0.989 and 0.991 respectively for 1, 5 and 10 step-a-head predictions shown in 5.13 (g), (i) and (j). AUC for sine frequency change dataset is 0.765, 0.992 and 0.944 respectively for 1, 5 and 10 step-a-head predictions shown in 5.13 (k), (l) and (m). While AUC for sine amplitude change dataset is 0.724, 0.838 and 0.947 respectively for 1, 5 and 10 step-a-head predictions shown in 5.13 (n), (o) and (p).

5.6 Discussion and Conclusion

We have proposed a novel anomaly detection algorithm using LSTM (OLACD) that detects the anomalies based on the statistics of predictions error, details of model and methodology is given in chapter 3. In this chapter several experiments are undertaken as a means to fully comprehend the effectiveness and validity of OLACD algorithm. The formulated training and anomaly detection algorithm are experimentally analysed by implementing fifteen experiments using fully connected LSTMs model; train with backpropagation through time (BPTT), mapping univariate time series and these were implementing in MATLAB. Details of all experiments along with the parameter is given in table 5.2, where comparison of performance results in term of AUC is illustrated in table 5.5.

TABLE 5.5: COMPARISON OF RESULTS IN TERM OF AUC FOR PROPOSED ONLINE ANOMALY DETECTION AND CHANGE-POINT DETECTION MODEL USING LSTM FOR UNIVARIATE TIME SEREIS WITH Saurav *et. al.* [5]

DATA SETS	PERFORMANCE RESULTS FOR TIME SERIES					
	PREDICTION 1		PREDICTION 5		PREDICTION 10	
	AREA UNDER CURVE (AUC)					
	(Saurav *et. al.* [5])	(Proposed)	(Saurav *et. al.* [5])	(Proposed)	(Saurav *et. al.* [5])	(Proposed)
NAB-Sudden Change	0.939	0.931	0.977	0.991	0.967	0.988
NAB-Continuous Change	0.555	0.890	0.708	0.994	0.612	0.992
NAB-Incremental Change	0.738	0.938	0.893	0.989	0.944	0.991
Sudden frequency change (Sine Wave)	0.696	0.765	0.839	0.922	0.899	0.944
Sudden amplitude change (Sine Wave)	0.872	0.724	0.908	0.838	0.931	0.947

One of the online versions of technique that computes the statistics of error for streaming data and use these statistics for either transitory (outlier) anomaly detection or permanent (change-point) anomaly detection is proposed by Saurav *et. al.* in 2018 [5], which computes anomaly score based on error statistics and update the model parameter accordingly. Saurav *et. al.* in 2018 [5] validates the results of their online model in term of Area Under ROC Curve (AUC).

While our proposed novel approach for change-point detection is work in an online manner for streaming data and we also validate the results of our online approach in term of AUC, so we are comparing our results with online anomaly detection technique proposed by Saurav *et. al.* in 2018 [5]. For producing direct comparison, we are using same data sets and performance measures to validate our online anomaly and change-point detection model that use in [5]. Hence, the performance of proposed novel online change-point detection model is compared with the model by Saurav *et. al.* [5] in term of AUC and comparison is given in table 5.3. The results of investigations show that the proposed OLACD is working as good as the existing online model proposed in [5] in some cases, and novel OLACD is working better than existing model in most of case as shown in Table 5.3. Proposed OLACD model seems to be robust to change-points/permanent anomalies while working with multi-step ahead predictions as compare with the 1-step ahead prediction.

It is shown in table 5.3, 5.4 and 5.5 that OLACD model is less responsive to change-points/long-term anomalies in the 1-step ahead predictions and performing late detection rather than early detection, whereas the 1-step ahead predictions give advantage in term that it detects small number of false positives as compare to the multi-step ahead predictions. So, the AUC for 1-step ahead predictions for two data sets is lower than the model propsed in [5], for instance the AUC for NAB-Sudden Change dataset for 1-step ahead prediction is 0.931, for NAB-Continous Change dataset is 0.890, for NAB-Incremental Change dataset is 0.938, for Sudden Frequency Change datasest is 0.765 and for Sudden Amplitude Change data set is 0.724. The AUC resluts by Saurav *et. al.* [5] are not good for 1-step ahead predictions for NAB-Continous Change dataset which is 0.555, for NAB-Incremental Change dataset which is 0.738 and Sudden Frequency Change which is 0.696, while model in [5] perform as good as OLACD model for 1-step ahead predictions for NAB-Sudden Change dataset which is 0.939 and perform better than OLACD for 1-step ahead predictions for Sudden Amplitude Change dataset which is 0.872.

Comparison results demonstrated that our propsed online change-point detection algorithm is more robust to the changed in input data distribution when it work in multi-steps ahead predictions and performing early change-point detection that is one of the main characteristics of change-point detection techniques. Therefore, the AUC for 5-step ahead predictions 4 out of 5 compared datasets are higher than the model propsed in [5], for instance the AUC for NAB-Sudden Change dataset for 5-step ahead prediction is 0.991, AUC for NAB-Continous Change dataset is 0.994, for NAB-Incremental Change dataset is 0.989, for Sudden Frequency Change datasest is 0.922. On the other hand the AUC for 5-step ahead prediction for 4 datasets that produces by [5] are lower than proposed OLACD algorithm, which are, AUC for NAB-Sudden Change dataset for 5-step ahead prediction is 0.977, AUC for NAB-Continous Change dataset is 0.708, for NAB-Incremental Change dataset is 0.893, for Sudden Frequency Change datasest is 0.839. The AUC for 5-step ahead prediction for Sudden Amplitude Change data set is 0.838 which is lower than model in [5] that is 0.908.

The AUC for 10-step ahead predictions for all five compared datasets are also higher than the model given in [5], for illustration the AUC for NAB-Sudden Change dataset for 10-step ahead prediction is 0.988, AUC for NAB-Continous Change dataset is 0.992, for NAB-Incremental Change dataset is 0.991, for Sudden Frequency Change datasest is 0.944 and for Sudden Amplitude Change datasest is 0.947, which are better than the results computed by [5] which are AUC for NAB-Sudden Change dataset for 10-step ahead prediction is 0.967, AUC for NAB-Continous Change dataset is 0.612, for NAB-Incremental Change dataset is 0.944, for Sudden Frequency Change datasest is 0.899 and for Sudden Amplitude Change datasest is 0.931.

It is shown in table 5.5 that model trained with 1 -step ahead predictions produces less number of false positives as comapre to multi-step predictions (5-steps ahead or 10-steps ahead) so we can say that there is a trad off between early change-detection and prediction horizon, but model is more resposive to change-points in multi-step ahead predictions.

Chapter 6

Conclusion and Future Work

This final chapter will provide some discussion on approach in section 6.1 which was develop in Chapter 3 and verified for Short-term Anomalies in Chapter 4 and for Long-term Anomalies in Chapter 5. The research work is concluded in section 6.2 and some future work is presented in section 6.3.

6.1 Discussion

The ubiquitousness of digital technologies adjacent with the significant processing power and storage capacity gives the unprecedented and exceptional ability to researcher for retrieving and evaluating the streaming data. However, very often there is lack of the resources to utilise conventional data curation processes. For example, in supervised machine learning approaches, the data annotation process can be an expensive and cumbersome. Therefore, unsupervised learning methods are presently hot topics in the machine learning and data mining community.

Anomaly detection and change-point detection have become a crucial technique in data mining for identifying culpabilities, detecting malicious activities, and isolating erroneous measurements from data stream. Anomaly detection and change-point detection in time series are the significant tasks with several practical applications in real life, thus, there has been an increase in demand for practical online anomaly detection and change-point techniques. However, for anomaly detection to be efficient in the changing environments, there are several important research challenges which are addressed in this paper. These practical application environments can generate high speed data streams, which makes manual annotation of a large proportion of the observations difficult. Moreover, many monitoring environments of interest exhibit non-stationary behaviour, as normal behaviour can evolve over time. The conventional methodology of training a model in an offline way using historic data is probable to fail under non-stationary and dynamically changing environments where the description of normal behaviour changes over time making the model irrelevant and ineffective. Consequently, online unsupervised approaches to anomaly detection are vital and it is proposed, which can work with unlabelled and non-annotated datasets from non-stationary environments and can rapidly be retrained after changes in the environment. In this thesis we have focused on efficient and robust unsupervised model-based anomaly detection and change-point detection technique that were tailored to address the above challenges and has the potential to adapt to the complexity of changing environments.

It is very challenging task to detect the anomalies and change-points for non-annotated streaming data and in such case unsupervised methods are essential. To best of our knowledge the unsupervised methods for anomaly and change-point detection are model-free and require manual parameters tweaking which is main constraint in *online* process.

We have proposed a novel algorithm based on Long Short-Term Memory Networks (LSTMs) for time series anomaly detection and change-point detection which is capable to addresses the challenges posed by deviation from normal behaviour, either transitory (anomaly) or permanent (change point). The algorithm is capable to detecting spatial and temporal anomalies in predictable non-stationary environment and meets the requirements of real-time, online detection without look-ahead and supervision of continuously incoming data stream. The model is trained in incremental way for each new incoming data and can adapt the changes in the data distribution. A contemporary state-of-the-art recurrent neural network such as LSTMs is used to make single step/multi step predictions for each point of incoming data stream, and the prediction errors are used to update the LSTM model as well as detect anomalies and change points. Large prediction error is used to signify either transitory anomalous behaviour or a permanent change in normal behaviour. Additionally, the prediction errors are also used to update the LSTM training model in such a way that short-term transitory anomalies or outliers do not cause a drastic change in the model parameters. However high prediction errors over a period lead to substantial updates in the model parameters such that the model promptly adapts to the new norm.

The proposed method, unlike various of the existing methods, does not necessitate any prior distributional knowledge, trained model, or user-designated parameters such as threshold, getting it suitable for a wide range of domains. The proposed novel online anomaly and change detection model only necessitates a brief period for initialization of parameters at start which incrementally updated with time as new data arrives. Through empirical evidence, we demonstrate that the proposed novel model is capable to adapt to different types of changes including sudden change, gradual, and incremental changes in mean value as well as adapt to changes in frequency and amplitude for periodic time series.

As demonstrated in table 4.4 and 5.4, *accuracy* is always high when the model can identify the highest number of anomalies and changes points, regardless of how many false positives are identified. Since it relates to the proportion of an anomalous class of interest (anomalous or changes points) that was truly identified. The measure *recall* is rather simple. Therefore, when a model detects a large number of anomalous points, recall is always higher. The proportion of relevant examples among the recovered instances is known as *precision*, also known as positive predictive value. As a result, the model only provides high precision when it can anticipate all potential anomalous locations with a very low number of erroneous predictions, as shown in table 4.4 and 5.4. The performance results in table 4.4 and 5.4 for the proposed anomaly and change detection model show that high precision always leads in higher *F-scores* since F-score is the harmonic mean of a system's precision and recall values. In spite of the increased rate of false positive detections, accuracy is always high when the model is able to identify the greatest number of anomalies, as shown in table 4.4 and 5.4. Therefore, accuracy is not a good metric because of the class imbalance. Due to the modest ratio of changes to total data, the *G-mean*, which is frequently employed as an indication of model performance in cases with an unbalanced class distribution, is measured by many studies. *Sensitivity* and *Specificity* metrics are used to evaluate the algorithm's performance in terms of both the ratio of positive accuracy (Sensitivity) and the ratio of negative accuracy (Specificity). Table 4.4 and 5.4 demonstrates that the G-mean measure of the anomaly detection model is always greater when the model can forecast the largest number of anomalies/true positives, and values of G-mean are compromised when the model predicts the incorrect predictions.

Following are essential observations which can be made from the evaluation of proposed online anomaly and change-point detection system:

(1) We observe that online scaling only effectively converts the range of time series either non-stationary or stationary as can be seen in Figures 4.1 and it will not change the properties of time series and distribution of streaming data stream. This indicates that the proposed online approach is not only able to deal with the cases where only mean shifts such as in Figure 5.4b and 5.4d, but also it is similarly effective to the stream with continuously increasing (or decreasing) such as in Figure 5.4c.

(2) The model is successfully able to detect incremental anomaly (e.g., Figure 5.5g, 5.5j and 5.5m between points 2000-2200) also able to detect continuous anomaly (e.g., Figure 5.5f, 5.5i and 5.5l between points 2000-2200) and rapidly adapt to changes without any external interference.

(3) Pros and cons of multi-step prediction:

(a) The multi-step estimate portrays a significant role in detecting anomalies and change-points and adapting to the change. We perceive that model perform better for $l > 1$ compared to models with $l = 1$ for most data sets in terms of AUC as shown in Table 4.5 and 5.5.

(b) We observe that model trained for predictions of $l > 1$ are more competent to detect change points much earlier as compared to model trained to predict only the next point with $l = 1$.

(c) This is specifically useful for detecting changes that are of temporal nature (i.e., frequency change and amplitude change) and difficult and most of time not able to be detected by point-wise models where the network trained to predict only one point with $l = 1$.

(d) Predictions with $l = 1$ is not able to detect amplitude change in figure 5.7d and frequency change in figure 5.7e when they happen and, in some cases, it starts adapting much later when substantial changes in input distribution are visible to model, as shown in third subplot in Figure 5.7d and 5.7e. On the other hand, models with $l = 5$ and $l = 10$ are capable to detect changes in frequencies and amplitude considerably earlier as shown in Figures 5.7f, 5.7g, 5.7h and 5.7i. The models trained with $l = 1$ rely on the changes in input distribution to produce smaller probabilities whereas models trained on $l > 1$ (such as al $l = 5$ and $l = 10$) detect the temporal changes in the target output distribution and are consequently able to detect changes much early.

(e) We also observe that networks trained for $l = 1$ take more time to adapt to the changes as compared to the model trained with $l > 1$, shown respective subplots of figures 5.6 and 5.7

(f) The model with $l = 1$ is not able to detect small changes in frequency, such as the second change in cycle length started nearly from 9000 in figure 5.7e. However, the models trained at $l = 5$ and $l = 10$ are able to capture this small change in frequency as indicated in figure 5.7g and figure 5.7i.

(g) (d) Even though $l > 1$ in most case the model ensures to detection the changes either transitory or permanent, but it also leads to extended periods where the effect of change point is reflected in the probabilities. This is clearly visible in respective subplots of figures 5.6 and 5.7, where changepoint lead to continuous smaller probabilities at future time steps after the changepoint is over. So, we can say by observing model that networks trained for $l = 1$ take more time to adapt to the changes as compared to the model trained with $l > 1$, shown respective subplots of figures 5.6 and 5.7

(h) Unlike other method [5], our method has the advantage that only detect the anomalies as quickly as possible but did not let the anomalies to make drastic change in distribution that is clearly visible in Figures 4.2 and 4.3. But in case of point anomalies this suggests that there is a trade-off between how early the anomaly detection happened and how long do the false positives last is depending on prediction length l.

6.2 Conclusion

Anomaly detection and change-point detection are important problem within various research and application realms and have seen considerable advances in the last few decades. Among lot of available literature on the topic of the general anomaly detection, the amount of available literature is significantly lower when this problem is restricted to the online anomaly and changepoint detection in time-series. there exists now a suite of methodologies that can be utilized to the unsupervised problem for univariate time series but most of these approaches are target either anomaly detection or changepoint detection. The objective of this thesis has been to develop efficient and effective general techniques that can leverage these existing approaches [5] to improve anomaly and change detection performance in fully online manner.

We demonstrated the efficiency of the proposed approach on a diverse set of synthetic, publicly available and proprietary real-world datasets [6, 27], and our novel technique is capable to follow the ideal characteristics of online anomaly and change detection models that described in chapter 1, such as;

1. Model has been capable of making predictions in online manner, such that our proposed algorithm identifies the state of x_t from input data stream as normal or anomalous before receiving the sub- sequent x_{t+1}.

2. The algorithm has been learned continuously in an online way without a requirement of storing the entire time series.

3. The algorithm has worked in an unsupervised, automated manner, such that, it was not presume data annotation except assuming labels during evaluation.

4. The algorithm has integrated information about all previous data points and incorporated into the current detection.

4. The algorithm has independent of manual parameter setting and tweaking; our proposed system has not been relying on the choice of user-define threshold.

5. Algorithms adapt to dynamic environments and change points.

6. Algorithm has capable to target the class imbalance/rare class in streaming data and detect anomalies as early as possible.

7. Algorithms has able to minimize false positives and false negatives; and maximize the true positives and true negatives, and this requirement is mostly achieved by increasing the prediction horizon.

It is concluded that the aim of this study has been achieved which is to develop an effective and efficient unsupervised model-based parametric methodology for anomaly detection in continuous time series in an online setting by modelling the noise, such that the model computes the statistic of the error incrementally to detect the anomalies and change-points.

The proposed system is achieved all the objectives of research so far given in chapter 1, which are:

6) We have investigated and identified the knowledge gaps by critically analysed the current offline anomaly detection method and need of online anomaly detection method in chapter 2.
7) System has utilized contemporary RNN [19, 134] based approach to handle both outlier/point anomalies and change points simultaneous in an online way for streaming data, and methodology of proposed system was given in details in chapter 3.
8) System has introduced and evaluated a novel proposed online anomaly and change-point detection for non-linear time series regression problem of multistep ahead predictions using LSTMs and discussed in chapter 4 and 5.
9) We validated the performance achieved by proposed novel online anomaly detection algorithm with respect to conventional methods and results are discussed in details chapter 4 and 5.
10) We investigated that various architecture of LSTM for one/multistep predictions, to find out the best architecture that gives the better performance.
11) We investigated new features such as *threshold* automatic estimation to propose novel algorithms to make it fully online and responds to changing behaviour of data either transitory (i.e., outlier) or permanent (i.e., change points).

Altogether, the above-mentioned requirements are fulfilled by our proposed novel anomaly and change-point system which recommend that online anomaly and change detection for streaming data is a fundamentally different problem than static offline methods. As discussed in related work given in chapter 2, the majority of existing algorithms are not appropriate for both anomaly and change-point detection for streaming applications at same time. But the proposed method is capable for detecting both types simultaneously in an online way, that is shown by evaluating performance for anomaly detection in chapter 4 and for change-point detection in chapter 5. In next section we will discuss observations from proposed method.

6.3 Future Work

By observing and evaluation proposed model there are several works which we are intended to work-on in future.

(1) This is proved in this thesis and results demonstrated in table 4.5 and 5.5 that the proposed online anomaly and change detection algorithm is effective and efficient for univariate time series, therefore our next goal is extended the existing model to fully online single/multistep multivariate anomaly and change detection for time series in non-stationary environment.

(2) We would like to perform the sensitivity analysis of various parameters of proposed system, including model learning parameters such as weights, learning threshold, recurrent frame size etc., and anomaly detection parameters such as $\mu_t(mean), V_t(covariance\ ), \tau_t(threshold)$ and $\alpha\,(alpha)$. This will give us the opportunity to optimize the system by finding the parameter which effect the system performance.

(3) Currently, in this thesis we have evaluated the effectiveness and efficiency of proposed algorithm using contemporary one type of shallow recurrent neural network (LSTM with one hidden layer) [1, 134]. The efficiency of algorithm can be improved by utilizing deep recurrent neural networks which helps to train the model with high accuracy and precision that generate with low error. So, it would be beneficial to compare the efficiency of proposed algorithm using different modern RNN such as Bidirectional RNN [19], Convolutional RNN [94], Gated Recurrent Units [5]and etc.

(4) As the system is fully automated and initializes all parameter only from the small amount of presumably non-anomalous data which is available for training the networks before starting the actual anomalies/changepoint predictions, so we would like to

investigate and add new feature that improve the efficiency of proposed algorithms, such that the feature help improving the initialization of anomaly detector parameters

(5) that continuously updated with time and includes $\mu_t(mean)$, $V_t(covariance)$, $\tau_t(threshold)$ and also that initialized at one and remain same thought the completion of one experiment such as α (*alpha*). The value α (*alpha*) is the significance level for chi-square test of independence, which is the probability of rejecting the null hypothesis when it is true. For example, a significance level of 0.001 indicates a 0.1% chance of concluding that x_t is anomalous at time t and 9.99% chance of x_t being concluded as normal data point. This is important when the streaming data is habitually having rare class/ imbalance class.

(6) We performed all experiments using backpropagated phase of first order automatic differentiation [135] and we would like to extend the research by assuming and utilizing second order automatic differentiation [135] and compare the results with currently proposed algorithms.

(7) Finally, currently, we are utilizing general architecture of interconnected neurons/units that disseminated calculated information to next layers and combinedly make layered architecture fulfilling the characteristics of artificial neural networks that usually generates the error at last layer and network decision is dependent on the generated error [19]. We would like to investigate and introduce the probabilistic interconnected neuron [136] which presumably decide at individual level that either the calculated information is disseminated forward or discarded, and eventually final decision will make at final layer [136].

References

1. Gers, F.A., *Long-Short term Memory in Recurrent Neural Network*. 2001, University of Hannover, Germany.
2. Kumar, A., *Difference between Online & Batch Learning* 05/12/2022.
3. Faithfull, W., *Unsupervised Change Detection in Multivariate Streaming Data*. 2018.
4. Le, X.-H., Hung Viet Ho, Giha Lee, and Sungho Jung, *Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting Water 11, no. 7: 1387. https://doi.org/10.3390/w11071387*. 2019. **11**(7).
5. Saurav, S., et al. *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 Association for Computing Machinery*. 2018.
6. Ahmad, A.L.a.S., *Evaluating Real-Time Anomaly Detection Algorithms–The Numenta Anomaly Benchmark.* In Machine Learning and Applications (ICMLA), IEEE 14th International Conference on. IEEE, 2015: p. 38–44.
7. Ahmad, S.P., Scott, *Real-Time Anomaly Detection for Streaming Analytics.* arXiv: 1607.02480v1 [cs.AI] 2016.
8. Chauhan, S.a.L.V. *Anomaly detection in ECG time signals via deep long short-term memory networks*. in *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 2015.
9. Aminikhanghahi, S., and Diane J Cook *A Survey of Methods for Time Series Change Point Detection.* Knowledge and information systems, 2017. **51**(2): p. 339-367.
10. Andrew Cook, G.M., and Zhong Fan, *Anomaly Detection for IoT Time-Series Data: A Survey.* Internet of Things Journal IEEE, 2019.
11. Randy Paffenroth, K.K., and Les Sarvi *Robust PCA for Anomaly Detection in Cyber Networks.* arXiv:1801.01571v1 [cs.CR] 4 Jan 2018.
12. al, A.U.e., *IoT Healthcare Analytics: The Importance of Anomaly Detection*, in *IEEE 30th International Conference on Advanced Information Networking and Applications*. 2016.
13. Jonguk Kim, J.-H.Y., Hyoung Chun Kim, , *Anomaly Detection for Industrial Control Systems Using Sequence-to-Sequence Neural Networks.* arXiv:1911.04831 [cs.CR].
14. Pankaj Malhotra, L.V., Gautam Shroff, and Puneet Agarwal *Long Short Term Memory Networks for Anomaly Detection in Time Series.* In ESANN, 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2015(89–94).
15. V. Chandola, A.B., V. Kumar, *Anomaly detection: a survey.* ACM Comput. Surv., 2009. **41**: p. 1–72.
16. Ghafoori, Z., *Robust and efficient unsupervised anomaly detection in complex and dynamic environments*. 2018: http://hdl.handle.net/11343/213143.
17. Ergen, T.a.S.S.K., . I, *Efficient Online Learning Algorithms Based on LSTM Neural Networks.* EEE Transactions on Neural Networks and Learning Systems, 2018. **29(8)**: p. 3772-3783.
18. Specht, D.F., *A general regression neural network.* IEEE Transactions on Neural Networks, 1991. **2(6)**:: p. 568-576.
19. Greff, K., et al.,, *LSTM: A Search Space Odyssey.* IEEE Transactions on Neural Networks and Learning Systems, 2016. **PP(99)**: p. 1-11.
20. J. Gama, I.Z.e., A. Bifet, M. Pechenizkiy, and A. Bouchachia, *A survey on concept drift adaptation.* ACM Computing Surveys (CSUR), 46(4):44, 2014.
21. D. C. Montgomery C. L. Jennings, a.M.K., *Introduction to Time Series Analysis and Forecasting*. 2011: John Wiley & Sons.

22. LC Jain, M.S., CP Lim, P Balasubramaniam, *A review of online learning in supervised neural network.* Neural Computing and Applications. **25 (3-4)**: p. 491-509.
23. Haimin Yang, Z.P., Qing Tao, *Robust and Adaptive Online Time Series Prediction with Long Short-Term Memory.* Computational Intelligence and Neuroscience, 2017. **2017, Article ID 9478952** p. p. 9.
24. Guo, T., et al. *Robust Online Time Series Prediction with Recurrent Neural Networks*. in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 2016.
25. Schmidhuber, S.H.a.J.u., *Long short-term memory.* Neural computation 9, 1997. **8 (1997)**: p. 1735–1780.
26. A Malhotra, P.V., Lovekesh Shroff, Gautam, Agarwal. *Long Short Term Memory Networks for Anomaly Detection in Time Series*. in *ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence*

and Machine Learning. Bruges (Belgium) 22-24 April 2015,. 2015.

27. Nikolay Laptev, S.A., and Y. Billawala. , *Yahoo Labs News: Announcing A Benchmark Dataset, For Time Series Anomaly Detection* 2015.
28. S. Ahmad, A.L., S. Purdy, and Z. Agha, *Unsupervised real time anomaly detection for streaming data.* Neurocomputing, 2017. **262**: p. 134_147.
29. MOHSIN MUNIR, S.A.S., ANDREAS DENGEL , AND SHERAZ AHMED, *DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series.* Digital Object Identifier 10.1109/ACCESS.2018.2886457, 2018.
30. Shapiro, J.L., *Online Mean and Covariance* U.o. Manchester, Editor. 2019: Perosnal Communication.
31. Giraud-Carrier, C. *Meta-Learning by Landmarking Various Learning Algorithms*. in *Proceedings of the Seventeenth International Conference on Machine Learning*. 2000.
32. Hoi, S.C., Wang, J., and Zhao, P., *Libol: A Library for Online Learning Algorithms.* Journal of Machine Learning Research, 2014. **15**: p. 495–499.
33. Shalev-Shwartz, S., *Online Learning and Online Convex Optimization.* Foundations and Trends in Machine Learning 2011. **4**(2): p. 107–194.
34. Chongsheng Zhang, Y.Z., Xianjin Shi, George Almpanidis, Gaojuan Fan & Xiajiong Shen *On Incremental Learning for Gradient Boosting Decision Trees.* Neural Processing Letters 2019. **50**: p. 957–987.
35. Yaxiong Zeng, D.K., *Online Adaptive Machine Learning Based Algorithm for Implied Volatility Surface Modeling, Yaxiong Zeng, Diego Klabjan*. 2018.
36. E.J. Spinosa, A.P.D.L.F.D.C., J. Gama, *OLINDDA: a cluster-based approach for detecting novelty and concept drift in data streams* Proceedings of the 2007 ACM Symposium on Applied Computing, 2007: p. 448–452.
37. E.R. Faria, J.G., A.C. Carvalho. *Novelty detection algorithm for data streams multi-class problems*. in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. 2013.
38. P.Y. Chen, S.Y., J.A. McCann, *Distributed real-time anomaly detection in net- worked industrial sensing systems*, in *IEEE Trans. Ind.* . 2015. p. 3832–3842.
39. S. Lee, G.K., S. Kim, *Self-adaptive and dynamic clustering for online anomaly detection,.* Expert Syst. Appl. , 2011. **38 (2011) 1**: p. 4891–14898.
40. T. Ahmed, M.C., A. Lakhina. *Multivariate online anomaly detection using kernel recursive least squares*. in *Proceedings of the 26th IEEE Interna- tional Conference on Computing Communication*. 2007.
41. N. Laptev , S.A., I. Flint *Generic and Scalable Framework for Automated Time-series Anomaly Detection*. in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*. 2015.
42. J., W., *Netflix Surus GitHub, Online Code Repos* https://github.com/Netflix/ Surus 2015
43. E. Keogh, J.L., A. Fu. *HOT SAX: Efficiently finding the most unusual time series subsequence* in *Proceedings of the IEEE International Conference on Data Mining, ICDM*. 2005.

44. Sean J. Taylor, B.L., *Prophet: forecasting at scale.* https://research.fb.com/prophet-forecastingat-scale, 2017.

45. Xu, H.C., W.; Zhao, N.; Li, Z.; Bu, J.; Li, Z.; Liu, Y.; Zhao, Y.; Pei, D.; Feng, Y.; et al. *Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications*. in *In Proceedings of the 2018 World Wide Web Conference on World Wide Web*. 2018.

46. Kejariwal, A., *Twitter Engineering: Introducing Practical and Robust Anomaly Detection in a Time Series [Online blog]*. 2015: http://bit.ly/1xBbX0Z.

47. A. Stanway, E., *Skyline, Online Code Repos.* https://github.com/etsy/ skyline, 2013.

48. Ahmad, J.H.a.S., *Why neurons have thousands of synapses, a theory of sequence memory in neocortex* Frontiers in Neural Circuits, 2016. **10**: p. 23.

49. Grubbs., F.E., *Procedures for detecting outlying observations in samples.* Technometrics 1969. **11(1)**: p. 1–21.

50. A. Bernieri, G.B., C. Liguori. *On-line fault detection and diagnosis obtained by implementing neural algorithms on a digital signal processor*. in *IEEE Trans. Instrum*. 1996.

51. MacKay, R.P.A.a.D.J.C., *Bayesian online changepoint detection* CoRR, 2007.

52. Willis A. Jensen, L.A.J.-F., Charles W. Champ, and William H. Woodall, *Effects of parameter estimation on control chart properties: a literature review.* Journal of Quality Technology 2006. **38, 4** p. 349–364.

53. Fu, S., Liu, J., Pannu, H. *A hybrid anomaly detection framework in cloud computing using one-class and two-class support vector machines*. in *International Conference on Advanced Data Mining and Applications*. 2012. Springer, New York

54. Wang, C., Viswanathan, K., Choudur, L., Talwar, V., Satterfield, W., Schwan, K. . *Statistical techniques for online anomaly detection in data centers* in *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*. 2011.

55. Constantinos S. Hilas, I.T.R., and Paris Ast. Mastorocostas, *Change Point Detection in Time Series Using Higher-Order Statistics: A Heuristic Approach.* 2013.

56. Chengwei Wang, K.V., Lakshminarayan Choudur, Vanish Talwar, Wade Satterfield, Karsten Schwan, *Statistical Techniques for Online Anomaly Detection in Data Centers.* HP Laboratories, 2011. **HPL-2011-8**.

57. Angelov, P. *Anomaly detection based on eccentricity analysis* in *Proceedings of the 2014 IEEE Symposium Evolving and Autonomous Learning Systems*. 2014.

58. A.M. Bianco , M.G.B., E.J. Martínez , V.J. Yohai, *Outlier detection in regres- sion models with ARIMA errors using robust estimates.* J. Forecast, 2011. **20** p. 565–579

59. H. Lee, S.J.R. *On-line novelty detection using the Kalman filter and extreme value theory*. in *Proceedings of the 19th International Conference on Pattern Recognition*. 2008.

60. Y.J. Lee, Y.R.Y., Y.C.F. Wang, *Anomaly detection via online oversampling principal component analysis*, in *IEEE Trans. Knowl. Data Eng* 2013. p. 1460–1470.

61. Wang, S., Minku, L. L., and Yao, X. *A systematic study of online class imbalance learning with concept drift*. in *IEEE Transactions on Neural Networks and Learning Systems*. 2018.

62. Ditzler, G.a.P., R. *Incremental learning of concept drift from streaming imbalanced data*. in *IEEE Transactions on Knowledge and Data Engineering*. 2013.

63. Mirza, B., Lin, Z., and Liu, N, *Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift.* Neurocomputing, 2015. **149**: p. 316– 329.

64. Wang, S., Minku, L. L., and Yao, X. *Resampling-based ensemble methods for online class imbalance learning*. in *IEEE Transactions on Knowledge and Data Engineering*. 2015.

65. Arabmakki, E.a.K., M, *Som-based partial labeling of imbalanced data stream.* Neurocomputing, 2017(262): p. 120–133.

66. Li, P., Wu, X., Hu, X., and Wang, H, *Learning concept-drifting data streams with random ensemble decision trees.* Neurocomputing, 2015. **166**: p. 68–83.

67. Xing Wang, J.L., Nital Patel, and Martin Braun. *A Self-Learning and Online Algorithm for Time Series Anomaly Detection, with Application in CPU Manufacturing.* . in *In Proceedings of the 25th ACM International on Conference on Information*

and Knowledge Management. 2016. ACM.
68. Ahmed, T., *Online anomaly detection using KDE*, in *Global Telecommunications Conference, 2009. GLOBECOM*. 2009, IEEE. p. 1-8.
69. Yuan Yao, A.S., Leana Golubchik, and Ramesh and Govindan, *Online anomaly detection for sensor systems: A*

simple and efficient approach Performance Evaluation, 2010. **67**(11): p. 1059–1075.
70. Takeuchi, K.Y.a.J.-i. *A unifying framework for detecting outliers and change points from non-stationary*

time series data. in *In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2002. ACM.
71. Chengwei Wang, K.V., Lakshminarayan, Choudur, Vanish Talwar, Wade Satterfield, and Karsten Schwan., *Statistical techniques for online anomaly detection in data centers*, in *Integrated Network Management (IM), 2011, IFIP/IEEE International Symposium on. IEEE.* 2011, IEEE. p. 385–392.
72. Umaa Rebbapragada, P.P., Carla E Brodley, and Charles Alcock, *Finding anomalous periodic time series.* Machine learning 2009. **74**(3): p. 281–313.
73. Fomby, T.B., *Exponential smoothing models.* Mannual SAS/ETS Software: Time Series Forecasting System. , 2008. **Version 6**: p. 225–235.
74. Cynthia Freeman, J.M., Ian Beaver, and Abdullah Mueen, *Experimental Comparison of Online Anomaly Detection Algorithms*, in *Thirty-Second International Florida Artificial Intelligence Research Society Conference (FLAIRS-32)*. 2019.
75. Hochenbaum, J.V., O. S.; and Kejariwal, *Automatic anomaly detection in the cloud via statistical learning.* arXivpreprint arXiv:1704.07706 2017.
76. Choudhary, S.H., G.; and Saini, S. K. *Sparse decomposition for time series forecasting and anomaly detection*. in *Proceedings of the 2018 SIAM International Conference on Data Mining*. 2018.
77. Laptev, N.A., S.; and Flint, I. . *Generic and scalable framework for automated time-series anomaly detection*. in *In Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015.
78. Laura Rettig, M.K., Philippe Cudr´e-Mauroux, and Michal Pi´orkowski., *Online anomaly detection over big data streams*, in *In Big Data (Big Data), 2015 IEEE International Conference* IEEE, Editor. 2015. p. 1113–1122.
79. Hochreiter, S.a.J.S., *Long Short-Term Memory.* Neural Computation, 1997. **9(8)** p. 1735-1780.
80. David E. Rumelhart, G.E.H.u.R.J.W., *Learning representations by back-propagating errors.* Nature (London), 1986. **323, S**: p. 533-536.
81. Werbos, P., *Backpropagation Through Time: What It Does and How to Do It* in *Proceedings of the IEEE* 1990. p. 1550–1560.
82. Rumelhart, D.E., *Backpropagation: Theory, Architectures, and Applications*. 2013: Psychology Press. ISBN 978-1-134-77581-1.
83. Baydin, A.G., et al *Automatic differentiation in machine learning: a survey.* arXiv reprint arXiv:1502.05767, 2015.
84. A, G., *On automatic differentiation* Mathematical Programming: Recent Developments and Applications. 1989: Kluwer Academic Publishers.
85. Geoffrey Hinton, N.N.f.M.L.L.a., *Overview of mini-batch gradient descent,* http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.
86. John Duchi, E.H., Yoram Singer, *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.* 2011. **12(61)**: p. 2121–2159.

87. Nesterov, Y., *A method for unconstrained convex minimization problem with the rate of convergence o(1/k2)* Doklady ANSSSR (translated as Soviet.Math.Docl.) 1983. **269**: p. 543–547.
88. Kingma, D.P., & Ba, J. L, *Adam: a Method for Stochastic Optimization*, in *International Conference on Learning Representations*. 2015. p. 1–13.
89. Dozat, T., *Incorporating Nesterov Momentum into Adam.* ICLR Workshop, 2016. **(1)**: p. 2013–2016.
90. Zeiler, M.D., *ADADELTA: An Adaptive Learning Rate Method.* Retrieved from http://arxiv.org/abs/1212.5701, 2012.
91. Kozat, T.E.a.S.S., *Efficient Online Learning Algorithms Based on LSTM Neural Networks*, in *IEEE Transactions on Neural Networks and Learning Systems,*. 2018. p. 3772-3783.
92. Zipser, R.J.W.a.D., *A learning algorithm for continually running fully recurrent neural networks.* Neural computation, 1989. **1(2)**: p. 270– 280.
93. Cheng, M., Xu, Q. , Lv, J. , Liu, W. , Li, Q. , & Wang, J, *MS-LSTM: A multi-scale LSTM model for BGP anomaly detection.* In IEEE international conference on net- work protocols (ICNP) 2016: p. 1–6.
94. Tailai Wen, R.K., *Time Series Anomaly Detection Using Convolutional Neural Networks and Transfer Learnin.* arXiv:1905.13628, 2019.
95. Ren, Y., & Wu, Y, *Convolutional deep belief networks for feature extrac- tion of EEG signal*, in *In International joint conference on neural networks (IJCNN)*. 2014. p. 2850–2853.
96. Biao Yang , J.C., Rongrong Ni, and Ling Zou, *Anomaly Detection in Moving Crowds through Spatiotemporal Autoencoding and Additional Attention.* Hindawi Advances in Multimedia 2018. **2018 Article ID 2087574** p. 8.
97. Mingyi Zhu, K.Y., Yang Wang, *A Deep Learning Approach for Network Anomaly Detection Based on AMF-LSTM* Network and Parallel Computing, 2018. **11276**(ISBN : 978-3-030-05676-6).
98. Piotr S. Maciag , M.K., Robert Bembenik , Jesus L. Lobo , Javier Del Ser, *Unsupervised Anomaly Detection in Stream Data with Online Evolving Spiking Neural Networks.* arXiv:1912.08785v1 [cs.NE] 2019.
99. Yun-Long Kong , Q.H., Chengyi Wang, Jingbo Chen, Jiansheng Chen and Dongxu He, *Long Short-Term Memory Neural Networks for Online Disturbance Detection in Satellite Image Time Series ,.* Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing 100101, China, Technical Note, , 2017.
100. Narendhar Gugulothu, P.M., Lovekesh Vig, Gautam Shroff, *Sparse Neural Networks for Anomaly Detection in High-Dimensional Time Series* TCS Research, New Delhi, India.
101. Scott, J.K.a.C.D., *Robust kernel density estimation.* Journal of Machine Learning Research, 2012. **13**: p. 2529–2565.
102. J. Gao, W.H., Z. M. Zhang, X. Zhang, and O. Wu, *RKOF: robust kernel-based local outlier detection*, in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. 2011. p. 270–283.
103. He, B.T.a.H., *A local density-based approach for outlier detection* neurocomputing, 2017. **241**: p. 171–180.
104. S. Ramaswamy, R.R., and K. Shim,, *Efficient algorithms for mining outliers from large data sets* ACM Sigmod Record, 2000. **29(2)**: p. 427–438.
105. M. H. Sharif, S.U., and C. Djeraba, *Crowd behavior surveillance using bhattacharyya distance metric.* Proc. CompIMAGE, 2010: p. 311–323.
106. Huang, T., Zhu, Y., Zhang, Q., Zhu, Y., Wang, D., Qiu, M., Liu, L, *An lof-based adaptive anomaly detection scheme for cloud computing*, in *IEEE 37th Annual on Computer Software and Applications Conference Workshops (COMPSACW)*. 2013. p. 206–211.

107. Yamanishi, J.-I.T., G. Williams, and P. Milne, *On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms.* Data Mining and Knowledge Discovery, 2004. **8(3)**: p. 275–300.
108. Dengel, M.G.a.A. *Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm*. in *KI-2012: Poster and Demo Track*. 2012.
109. T. Zhang, R.R., and M. Livny, *BIRCH: An efficient data clustering method for very large databases.* ACM Sigmod Record, 1996. **25(2)**: p. 103–114.
110. V. Hautamaki, S.C., I. K arkkainen, T. Kinnunen, and P. Franti *Improving k-means by outlier removal* Scandinavian Conference on Image Analysis,

, 2005: p. 978–987.
111. F. T. Liu, K.M.T., and Z.-H. Zhou, *Isolation forest* Proceedings of the 8th IEEE International Conference on Data Mining (ICDM), 2008: p. 413–422.
112. S. Hawkins, H.H., G. Williams, and R. Baxter, *Outlier detection using replicator neural networks.* Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery (DaWaK), 2002: p. 170–180.
113. B. Scholkopf, J.C.P., J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, *Estimating the support of a high-dimensional distribution* Neural Computation, 2001. **13(7)**: p. 1443–1471.
114. Sinch. *Dynamic Threshold Estimation for Anomaly Detection*. 2021 [cited 2022 30/11/2022]; Available from: https://www.sinch.com/blog/dynamic-threshold-estimation-for-anomaly-detection/
115. Mehrotra, K.G., Mohan, Chilukuri, Huang, Huaming, *Anomaly Detection Principles and Algorithms*. 2017.
116. Greff, K., et al., *LSTM: A Search Space Odyssey.* IEEE Transactions on Neural Networks and Learning Systems, 2016. **PP**(99): p. 1-11.
117. *With QuickType, Apple wants to do more than guess your next text. It wants to give you an AI. WIRED [Online]. Available: https://www.wired.com/2016/06/apple-bringing-ai-revolution-iphone/ (Last Accessed Date 15-04-2017)*. 2016.
118. *Long short-term memory [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory (Last Accessed Date 15-04-2017).* 2016.
119. Sutskever., I., *Training Recurrent Neural Networks*. 2013, Graduate Department of Computer Science. University of Toronto.
120. Hochreiter, S. and J. Schmidhuber, *Long short-term memory.* Neural computation, 1997. **9**(8): p. 1735-1780.
121. Gers, F., *Long-Short term Memory in Recurrent Neural Network*. 2001, University of Hannover, Germany.
122. Séverin, P.D.J.a.E., *Predicting corporate bankruptcy using a self-organizing map: An empirical study to improve the forecasting horizon of a financial failure model* Decision Support System, 2011. **51(3)**: p. 701-711.
123. Distribution, G.o.t.M.N. *Geometry of the Multivariate Normal Distribution*. [cited 2018 11/06/2018].
124. Pajurek, T., *Online Anomaly Detection in Time Series*, in *Department of Theoretical Computer Science* 2018, Czech Technical University in Prague.
125. IDC, *https://www.idc.com/getdoc.jsp?containerId=prUS47560321 Date of retrieval: 20/02/2022*.
126. Tony Hey, S.T.a.K.T., *THE Fourth Paradigm: Data Intensive Scientific Discovery*. 2009.
127. IBM, *http://www-01.ibm.com/software/data/infosphere/streams/*.
128. Plugin, R.S., *http://www-ai.cs.uni-dortmund.de/auto?self=$eit184kc*.
129. Streambase, *http://www.streambase.com/*.
130. Analysis, M.M.O., *http://moa.cs.waikato.ac.nz/*.
131. S4, *http://incubator.apache.org/s4/*.
132. Storm, *https://github.com/nathanmarz/storm/wiki/Tutorial*.

133. Grok, *https://www.numenta.com/grok_info.html*.
134. Gers, F.A., J. Schmidhuber, and F. Cummins, *Learning to Forget: Continual Prediction with LSTM.* Neural Computation, 2000. **12**(10): p. 2451-2471.
135. Atılım Gune¸s Baydin, B.A.P., Alexey Andreyevich Radul, and Jeffrey Mark Siskind, *Automatic Differentiation in Machine Learning: a Survey.* Journal of Machine Learning Research **18 (2018)** p. 1-43.
136. Ferguson, A., *Learning in RAM-Based Artificial Neural Networks*. 1995, University of Hertfordshire.