LSTM based Framework for Time Series Anomaly Detection

N.A.A.H.Eranga

LSTM based Framework for Time Series Anomaly Detection

N.A.A.H.Eranga

Index : 14000334

Supervisor : Dr. M.I.E.Wickramasinghe

Submitted in partial fulfillment of the requirements of the

B.Sc. (Hons) in Computer Science Final Year Project (SCS 4124)

January 2019

Declaration

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

Candidate Name: N.A.A.H.Eranga

.....
Signature of Candidate
Date:

This is to certify that this dissertation is based on the work of Mr. N.A.A.H.Eranga under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Supervisor's Name: Dr. M.I.E.Wickramasinghe

.....
Signature of Supervisor
Date:

Abstract

Anomaly detection also known as Outlier detection is identification of data points, items or events that does not fit the expected behavior. In time series anomaly detection the objective is to detect anomalies in temporal data, a series of data points indexed in time order. This data can be network data, spatio temporal data and stream data, etc.

In this dissertation, an approache is proposed to detect anomalies in time series data based on deep neural networks and dynamic time warping algorithm. Initially we use a long short-term memory network to predict time series data. Then combined the LSTM network predictions with a convolutional neural network to get more accurate predictions. The predictions are compared with original time series data in a time window approach using DTW algorithm to identify the anomalies within the time series.

Evaluation process of the model is done using several steps. In the initial steps when developing the model three datasets are used to evaluate and test the models. For the final evaluation numenta anomaly benchmark dataset a novel benchmark for evaluating algorithms for anomaly detection, is used.

Keywords *- Anomaly detection, Long short-term memory networks, Convolutional neural networks, Dynamic time warping, hybrid neural networks*

Preface

A framework for anomaly detection in time series data based on LSTM networks is proposed in this research. The proposed framework consist of two modules, A prediction module and a detection module. In the prediction module we combined the LSTM prediction model with a CNN to get more accurate results. The LSTM network and the CNN was developed from the scratch without any neural network frameworks. In the detection module we use dynamic time warping algorithm to identify anomalies in a time series. The techniques used in the hybrid prediction model including the methods in feature combining layer are my own work.

Acknowledgements

I would like to express my sincere gratitude to my research supervisor, Dr. M.I.E.Wickramasinghe, lecturer of University of Colombo School of Computing for providing me continuous guidance and supervision throughout the research.

I would also like to extend my sincere gratitude to Dr. D.A.S.Atukorale, senior lecturer of University of Colombo School of Computing and Dr. K.Gunawardana, senior lecturer of University of Colombo School of Computing for providing feedback on my research proposal and interim evaluation to improve my study. I also take the opportunity to acknowledge the assistance provided by Dr. H.E.M.H.B.Ekanayake as the final year computer science project coordinator.

Many thanks to my beloved mother and my dear father for always being my strength, showing me the correct direction, and making me who I am today. This thesis is dedicated to all my family members, to primary and secondary school teachers, to lecturers of University of Colombo School of Computing and to anyone who supported even by words to make my dreams come true.

Table of contents

List of figures

List of table

Listings

List of Acronyms

ART1 Adaptive Resonance Theory 1.

CNN Convolutional Neural Network.

CNNLSTM CNN combined with LSTM network.

DNN Deep Neural Network.

FDR False Detection Rate.

FN False Negative.

FP False Positive.

KNN k-nearest neighbors.

LSTM Long Short-Term Memory.

LSTMCNN LSTM network combined with CNN.

LSTMCNNFC LSTM and CNN with Feature Combining Layer.

MLP Multilayer Perceptron.

MSE Mean Squared Error.

NAB Numenta Anomaly Benchmark.

TN True Negative.

TP True Positive.

Chapter 1

Introduction

Time series anomaly detection is a field of study that attempts to find abnormal patterns in time-based sequences and series. Anomaly detection of time-based signals has a large applicability among various disciplines such as system health monitoring, fault detection, and monitoring, surveillance, and vibration analysis of machines.

There are three main types of anomalies studied in the literature [1] named Point anomalies, Collective anomalies and Contextual anomalies. A point anomaly or an outlier is when an individual point is anomalous when compared to the rest of the data points in the temporal sequence. When a collection of several data points are anomalous when compared to the rest data it is identified as a collective anomaly. All the individual points in this collective anomaly need not be an anomaly but when they are occurring together it is a collective outlier. A contextual anomaly is when the data point is anomalous in a specific context but not otherwise. The below Figure 1.1, Figure 1.2 and Figure 1.3 shows the three maintypes of anomalies.

Figure 1.1: Point Anomaly

Figure 1.2: Collective Anomaly

Figure 1.3: Contextual Anomaly

1.1 Background

Time series anomaly detection is spread across large number of application areas. Due to the abundant growth of data streams at present in all application domains have paved the way for increased demand for anomaly detection models. In order to address this need, a number of supervised, semi-supervised, and unsupervised techniques based on anomaly detection models for time series data is available in the literature [1]. The learning models developed using these techniques have adopted various approaches such as neural network based approaches, statistical analysis based approaches, visual representation based approaches and signal processing based approaches. These approaches have different strengths and weaknesses and their performance vary with the type of temporal sequence being modeled.

1.2 Research Problem and Research Questions

Time sequences contain large volume of data with wide range of patterns. Processing these time series data to identify anomalies in them can be quite difficult due to the volume of data present in such series. This can lead to challenges in resource and data processing requirements. The presence of a large number of anomalies with widely spread patterns also make the anomaly detection more complicated as it is impossible to know all the anomalous patterns. In order to overcome the aforementioned challenges in anomaly detection, a LSTM based prediction model and how this prediction model can be used to identify anomalies by analyzing the predictions are explored.

The main goal of this research project is to develop a framework for anomaly detection in temporal sequences. The anomaly detection should based on the previous patterns of the temporal sequence and the system should learn new temporal patterns while predicting the future data points. The main aspects addressed are:

- Getting a better understanding about the current approaches to the anomaly detection in temporal sequences.

- Understanding the advantages and disadvantages of the different approaches and various types of time series data that can be used with.
- Studying anomaly detection approaches that are based on LSTM networks and problems addressed using this approach.
- Studying about Anomaly Detection approaches that use convolutional neural networks.
- Exploring different preprocessing methods of prediction model inputs to extract features from the temporal sequence.
- Exploring various methods of temporal data comparing approaches to detect anomalies more accurately according to the domain of the time series.

1.2.1 Research Questions

- What are the salient properties required to predict anomalies in time series data?
- How does these identified properties enable anomaly detection in time series data?
- What are the best approaches to develop prediction models for anomaly detection in time series data?
- How long short-term memory networks affect the predictions of the temporal sequences?
- How to analyze the similarity between temporal sequences?

1.2.2 Research Objectives

- Developing A LSTM neural network model for time series predictions.
- Improving the predictions of the model by using hybridization approach.
- Developing time series analyzing method based on dynamic time warping.

- Identifying different anomalies according to the temporal sequence properties.

1.3 Justification for the research

The variability of performance in some time-series anomaly detection techniques and the applicability of the model in some instances mainly stems from three aspects inherent to time-series data. The three factors are a higher prevalence of anomalies in a given time series, a large number of data points with sporadic positive samples and the tendency of the time signal varying over time in some application domains. The first aspect of a higher number of anomalies in a given series of data generally results in an unstable model due to the difficulty in training a model to all predefined anomalies in a given series. Furthermore, both the first and the second aspects mentioned above also leads to processing and resources challenges. In the third aspect of dynamically varying time-series, the variability in the series renders the previously developed anomaly detection models invalid in the varying context.

1.4 Methodology

Initially we study the currently using anomaly detection methods, how these methods are used to detect anomalies, their strengths and weaknesses. In this research we are using a neural network based approach for anomaly detection. The main reason for this is neural network approaches has shown great improvement in pattern recognition field.

There are many neural network approaches for anomaly detection based on multilayer perceptron networks, convolutional neural networks, recurrent Neural networks and also the Long Short Term Memory networks an extension of RNNs.

The neural network approach use prediction models for time series forecasting. The accuracy of the anomaly detection is depending on the accuracy of the prediction model. The prediction model is trained by feeding the information about the patterns in the temporal sequence. If the model can remember long term dependencies the predictions also will be more accurate. Therefore using a LSTM based prediction model will be more suited.

As suggested in previous works [2] and [3] to improve the prediction accuracy of the LSTM models, the hybrid approach can be used by combining with various other deep neural network architectures. The hybrid models can be developed in different combinations of neural networks.

In order to address the previously mentioned aspects related to time-series anomaly detection, we propose a framework for anomaly detection based on Long Short-Term Memory networks and the Convolutional Neural Networks. The proposed framework consists of two modules named as Prediction module and Detection module. In the prediction module, the CNN and the LSTM are combined in a hybrid configuration. The hybrid network model in the module is then used to obtain prediction on time-series data which are passed to the detection module which identifies the anomaly in the time series by measuring the similarity between the prediction and the real temporal sequence using Dynamic Time Warping (DTW).

1.5 Outline of the Dissertation

The rest of the dissertation is organized as follows. Chapter two explores the currently used approaches in anomaly detection, what are the strengths and the weaknesses of these approaches. Chapter three describes proposed research design and methodology and how the research questions are addressed from the proposed method. In chapter four it describes the implementation details of the proposed methodology. Chapter five dedicated for the results evaluation of the proposed approach. Final chapter, the chapter six presents the conclusion and the future work.

1.6 Delimitations of Scope

Our framework for anomaly detection is developed using a prediction model based on LSTM neural networks. The three main types of anomalies, point anomaly, collective anomaly and contextual anomaly detection will be covered for time series data.

The LSTM prediction model will use the basic LSTM memory cell structure with input gate, forget gate and the output gate. The structure of the LSTM network can be changed by altering the number of memory cells to get better results.

Furthermore we will attempt to combine the LSTM model with other deep neural network models to increase the prediction accuracy. The developed hybrid prediction model will allow to adjust the parameters and the network structures.

After the time series forecasting an algorithm is used to analyze the differences between the predicted points and the original time series data. According to the given parameters and threshold values anomalies are identified.

Chapter 2

Literature review

2.1 Anomaly Detection

Anomaly detection in time series is a very broad field and studied across large number of application domains. It can be also called as Outlier detection. Because of the large number of domains there have been a vast amount of approaches and methods used for anomaly detection in time series data [1]. Due to this variety of application domains the anomalies can be identified differently in each domain. There are mainly three types of anomalies studied in the literature point outliers, collective outliers and contextual outliers.

2.1.1 Statistical Approach

There are various approaches to detect anomalies in temporal sequences. Statistical methods are one of the mostly used technique to find the anomalies. In statistical techniques they use mathematical models to predict the future points in the time series. These prediction models are made up of a number of predictors, which are variables that are likely to influence future results.

In statistical approaches they use several kinds of prediction models [4] such as Autoregressive (AR), Moving Average (MA), Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA). These models can be used to predict the time series depending on the temporal sequence.

In AutoRegressive models the forecasting is done based on the past values. these models are represented as AR(p) where the p indicates the number of terms which the predicted output is depending on. In moving average model the forecasted output is depending only on the random error terms which follow white noise process.

This model can be represented as MA(q) where the q indicates the number of error terms used.

AutoRegressive Moving Average model is a combination of both AR and MA models. That means the forecasting is depending on both past values and the error terms. This model can be represented as ARMA(p,q) where p indicate number of past values and q for error terms. In the ARIMA model 'I' means integration. The time series is differentiated to forecast the values. The differentiation can be done number of times to make the time series stationary.

Prediction models AR, MA and ARMA can only be used to predict the time series data if the time series is stationary and if the temporal data sequence is non stationary the time series should be differentiated to make it stationary.

2.1.2 Clustering Approach

Other than the statistical approach another largely used anomaly detection approach is the clustering approach. Clustering is an unsupervised techniques based approach. In this approach several techniques can be used to detect anomalies in the time series data.

In clustering approaches a similarity function is developed and the temporal sequence patterns are clustered using these functions. Applying these functions the distance between the similar temporal patterns are minimized and the distance between the sequences which not that similar is maximized. By this method patterns within the temporal sequence can be identified that if they belonging to a already existing cluster or not.

Anomalies can be identified from these clustering based methods by calculating the distance from the center of the closest cluster to the point where the similarity function identify the given temporal sequence pattern. One of the main important aspect of this clustering technique is the similarity measure.

A method that can be used for clustering temporal sequence patterns is Self Organizing Maps. In [5] they use several prototype SOM algorithms to detect anomalies in time series data. Their goal was to evaluate SOM models for anomaly detection using SOM based algorithms such as Standard SOM, Kangas' Model, TKM-

Temporal Kohonen Map, Recurrent SOM. The comparison is done by evaluating false positive rates and decision thresholds.

SOM clustering is very efficient when dealing with high dimensional data. In [6] they use Particle Swarm Optimization (PSO) with SOM to detect anomalies in high dimensional data. SOM can also be used for dimensionality reduction in high dimensional data. In [7] a graphical representation of the clusters and genetic algorithm-based travelling salesman approach is used to increase the quality of the SOM to do more accurate clustering.

Another classification approach that used anomaly detection is Support Vector Regression method. SVR can be used as a classification algorithm to classify normal patterns in a time series. Time series patterns can be matched with the models and then can generate a value that denote how much the time series pattern is similar to the already existing patterns. Then we can use predefined time intervals and threshold values to detect if the pattern is an anomaly or not.

One of the main challenge in this approach is when the time series is very large it can contain large number of patterns. New patterns can also be added to the classification with the time. Then the number of support vectors increase and with that it can lead to processing problems. One way to avoid this is by keeping only constant number of support vectors. Then it will keep constant number of previous temporal patterns and forget the older temporal sequences [8].

2.1.3 Genetic Algorithms

In addition to these methods Immunological principles can also be used to detect anomalous patterns in temporal data. The natural immune system uses a decentralized system for classifying normal and anomalous cells. The immune system is able to extract features to identity patterns, learn to recognize new patterns and also has a content addressable memory. By observing these mechanisms an anomaly detection algorithm is implemented in [9].

In that approach wavelet analysis is used for feature reduction in the time series data and the data patterns were converted in to binary representations before giving as the input. The anomaly detectors were generated using negative selection algorithm. They compared the results by using positive detection with ART1 neural

network and concluded that the both methods gives similar results when detecting anomalies and also, the repeated feature reduction using the wavelet analysis gradually loses the semantic information in data patterns that causes failures in anomaly detection.

2.1.4 Neural Network Approach

When considering contemporary literature there are various approaches to detect anomalies in temporal sequences. Among these approaches, the neural network based approaches are sought by many researchers due to the greater degree of scalability and the flexibility offered by such neural models [2], [4], [10]. The neural network based approaches can be broadly categorized into two as Convolutional Neural Network (CNN) based models and Recurrent Neural Network (RNN) based models. RNN is considered as an ideal network model for time series prediction tasks due to the networks ability to remember historical patterns. However the RNNs are susceptible to the gradient problems (vanishing gradient and exploding gradient) when presented with long and continuous training process.

When training a RNN it use a technique called Backpropagation Through Time (BPTT). While training using this gradient descent method for many iterations the gradient is tend to explode or vanish. Therefore it prevents the neural network from further learning. Another problem in RNNs are it is hard to keep long term dependencies in the memory of RNN.

In order to address the gradient problems in the RNN, the Long Short Term Memory (LSTM) networks were introduced in [11] by Hochreiter and Schmidhuber. The LSTM is a RNN variant with an inclusion of a memory cell [12] which consists of a number of gates that enables the cell to retain information for a relatively long period. The structure of the memory cell and its overall effects to the LSTM were studied in detail by Klaus et al [13]. Among the findings it was observed that the overall performance of LSTM was not tightly coupled with the cell structures and the forget gate, and the output activations are the most critical components in a cell. Furthermore, it was observed that coupling the input and the forget gates to introduce an update gate simplified LSTM further without a significant decrease in performance.

Therefore using the basic LSTM cell structure [14] for the LSTM network in the

prediction model can give better results and to optimize the performance the proposed update gate can be used.

Furthermore, LSTMs also have the ability to remember long term dependencies and due to that are more suitable for time series forecasting. However, when the LSTM network was developed as an autoregressive model to forecast a time series and was compared with a multilayer perceptron (MLP) [2] it was observed that the LSTM was unable to accurately follow the input signal. The research further suggested that LSTM's core strength is to learn and remember single events for long time and alluded to the viability of hybrid prediction models to increase the accuracy of prediction models.

Such hybrid prediction models for various vocabulary tasks were explored in [3]. In this study, various neural network architectures were combined to see the overall prediction accuracy of the vocabulary tasks. The evaluation compared the hybrid models (CNN+LSTM, LSTM+DNN and CLDNN) results with the vanilla implementations of CNN, LSTM and DNN. LSTM depicted the highest accuracy when taken alone and the CLDNN had the overall highest accuracy. Another similar hybrid model for prediction slopes and trend in temporal patterns were developed by Tao, Tien and Karl in [15]. The prediction model named TreNet is a combination of the LSTM and the CNN. In this model the LSTM was provided with the trend sequence and the CNN was provided with the raw input sequence. The effectiveness of the hybrid model was evaluated against several baselines models as well and non-neural network based models. Similarly, stacking of hidden layers when building a LSTM network can increase the prediction accuracy of the network as it is able to learn higher level temporal features. In [4] a predictor was developed to model normal behavior of a time series using stacked LSTM network. The conclusions state that stacked LSTM network was able to learn higher level temporal patterns without prior knowledge and gave better or similar results when compared with stacked RNN.

When considering anomaly detection models there are many neural network approaches for anomaly detection in time series data. In [16], Shipmon et al. describes a comparison between DNN, RNN, LSTM and a Fourier model. The conclusions stated that Fourier and RNN were more effective prediction models and DNN and LSTM gave more false positives at peaks. In another research which compared LSTM and RNN [10], it was found that the LSTM performed better at the anomaly

detection task. It is clear that the two studies provide contradiction conclusions. This may be mainly associated with the inherent properties of the datasets used by each individual study and the anomaly type being modeled. Furthermore, LSTMs have been used to detect collective anomalies in time series data is described in [17]. In this research, the LSTM is used to predict the future points and calculate the relative error value comparing the real data points with the predictions from the network. Apart from LSTMs, CNNs also have a significant applicability to predict the time series data and anomalies. In [18] they have observed a multi-channel deep convolutional network (MC-DCNN) for multivariate time series data. In the prediction model it take a single dimension of the multivariate time series and learn the features individually. They compared the results and performance with a nearest neighbor (NN) prediction model. NN model increases linearly as the size of training data set grows and the prediction time of MC-DCNN model is almost constant does not depend on the size of the training data set. As conclusions they presented that the deeper architectures can learn more robust high level features.

Another attempt to observe multivariate time series data prediction is conducted in [19] .In this they have observed how LSTM network predict a multivariate time series using a large data set of multivariate clinical time series. The observations from this concluded that the LSTM can be successfully used to predict large multivariate temporal data. As the future work they suggested techniques to directly handle missing data values and methods to handle irregular sampling of the data.

Other than the prediction models, the most important part in an anomaly detection mechanism is to identify anomalies in time-series data by comparing the predicted time series with the original time series. The methods like probabilistic methods are used to detect anomalies by using an error value which can be generated by prediction values and original values. Another time series comparing technique is Dynamic Time Warping (DTW). In this algorithm a similarity value is calculated by finding the optimal global alignment between the two time series, by exploiting temporal distortions. The time complexity of the algorithm is $O(n^2)$ therefore when the size of the time series is increases the time consumption of this algorithm increase rapidly. As a solution to this FastDTW [20] was introduced.

2.1.5 Summary

As presented above, there are many approaches for time series anomaly detection. These different approaches has advantages unique to them and the best approach can be chosen according to the anomaly detection task. The main challenges which is faced by the detection models are processing and resource challenges and updating the model according to time series pattern drifts. The remainder of this paper proposes a framework to handle the issues pertaining to anomaly detection mentioned above.

Chapter 3

Design

3.1 Research methodology

Time series anomaly detection has various challenges. Since new data arrives at every time instant in a time series, the scale of the data is very large. This generally leads to processing challenges. Other main challenge is the data model that is used have to be updated when there is serious model drift. This is related to the transitions in the local patterns in the time series.

In this research we propose an anomaly detection framework to overcome these challenges. Our framework consist of two modules named Prediction module and Detection module. Prediction module is a neural network based time series prediction model. Detection module is based on dynamic time warping algorithm.

A time series can have massive amount of data points. Therefore these can contain a large number of anomalous patterns. To overcome this resource and processing problem in our approach we use a prediction model to predict the normal pattern in the time series and then compare the predictions with the real world data to identify anomalies.

Furthermore to overcome the model drift problem that occurs when the normal pattern in the time series shift from one another pattern with the time we are using a LSTM based neural network model for our prediction model. The memory cell in the LSTM neural network gives the prediction model the ability to handle long term dependencies.

3.2 Prediction Module

Prediction module contain the prediction model developed using a hybrid configuration of LSTM network and a CNN.

3.2.1 LSTM Network

As the result of the memory cells, LSTM networks are one of the best way to handle long term dependencies. As the initial step we develop a prediction model based on LSTM network. The vanilla memory cell structure with three gates was used for the development of the network.

The network contains a single LSTM layer with memory blocks. Each memory block contains one memory cell. Figure 3.1 shows the structure of a basic memory cell used for the network. The gates in a cell are input gate (i), forget gate (f) and output gate (o). Input for the memory cell is a vector which contains the input at time (t) and the output from the cell at time (t-1).

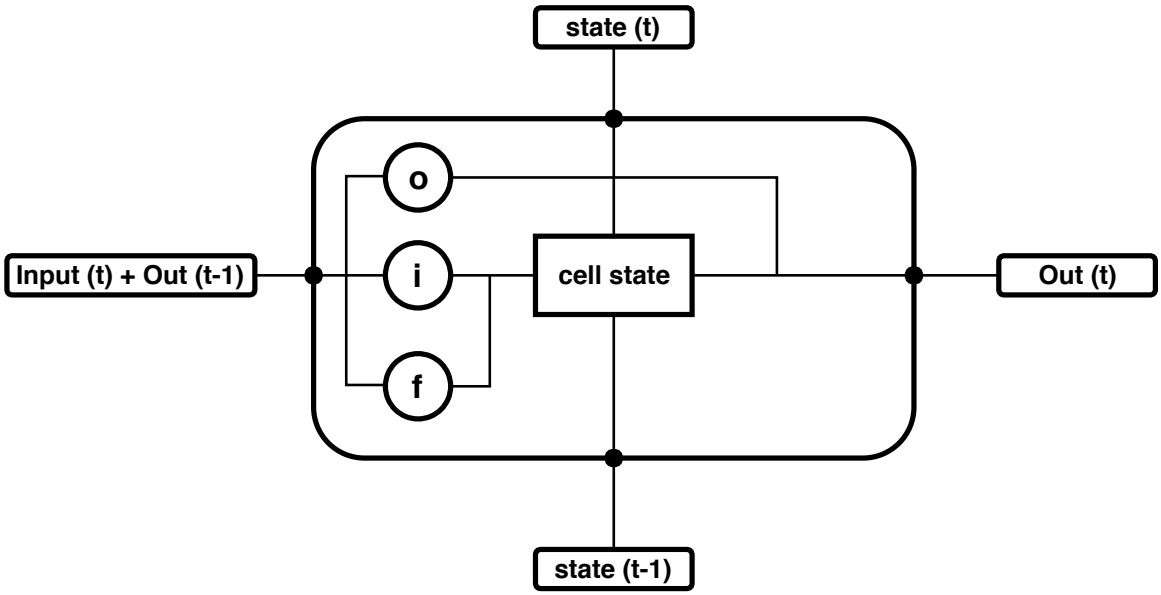

Figure 3.1: Memory cell structure

The cell state at time (t) is decided by the earlier cell state and the preceding input values. Current cell state is passed to the next instance of the cell. Several activation functions are used for the memory cell. For input and output activation

sigmoid function is used and other than that tanh activation is used.

The structure of the network in the prediction model is adjustable according to the input time series. This prediction model is individually trained and tested for several time series data sets including univariate and multivariate time series data to observe how the LSTM prediction model works.

In Figure 3.2 it shows the structure of the LSTM network that we used for our prediction model. Input layer is fully connected to the LSTM layer. The number of LSTM memory blocks can be adjusted. Learning rate of the network, number of time steps unfolded can be adjusted as parameters.

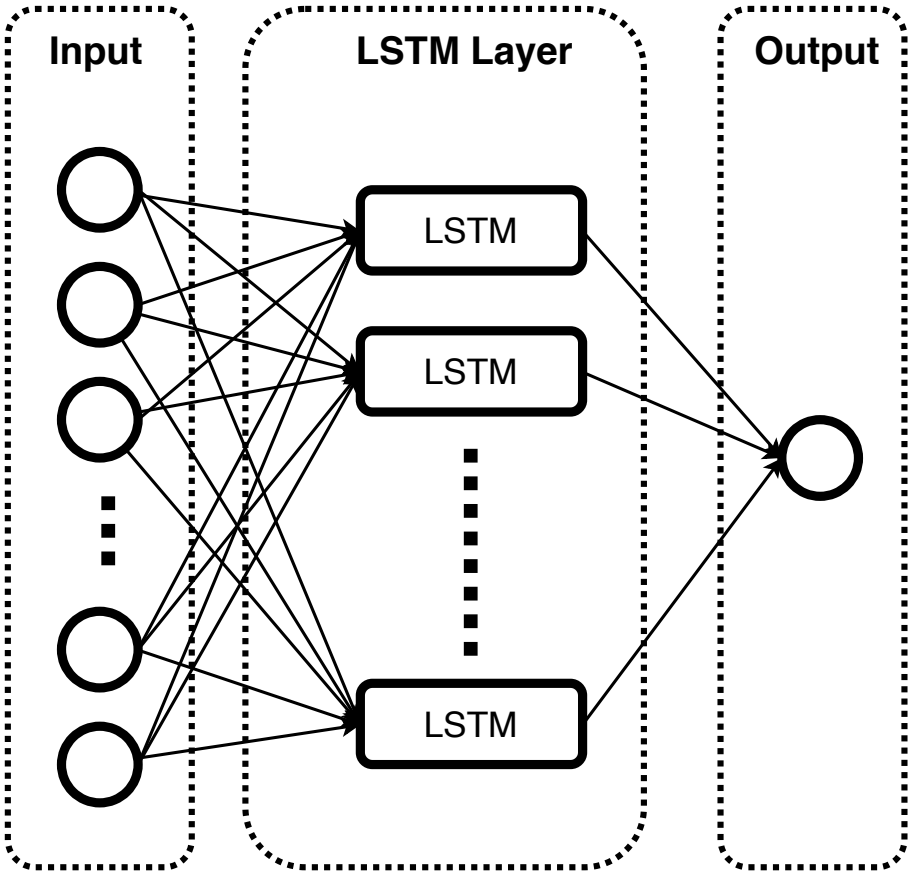

Figure 3.2: LSTM network structure

Time Series Processing

Training dataset is normalized before training the network. network is trained for

the normalized values. for the predictions in each instance the input time window is normalized before feeding to the network. the predicted value is then processed.

Post processing

A normalized time series is fed into the network. Therefore the prediction has to be processed to get the real world value. In the preprocessing step, the input time series is normalized by dividing all the values by the magnitude of the input vector, Then when post processing the prediction is multiplied by this vector magnitude.

$$vector = (a, b, c, d)$$

$$magnitude(m) = \sqrt{(a^2 + b^2 + c^2 + d^2)}$$

$$input(normalized) = (\frac{a}{m}, \frac{b}{m}, \frac{c}{m}, \frac{c}{m})$$

$$prediction = p$$

$$processed = p \times m$$

3.2.2 CNN Network

The structure of the CNN that used for the prediction model is explained in the Figure 3.3. The network includes Convolutional layers, Pooling layers and fully connected layers. In this deep neural network the number of layers can be changed to adjust the prediction model parameters.

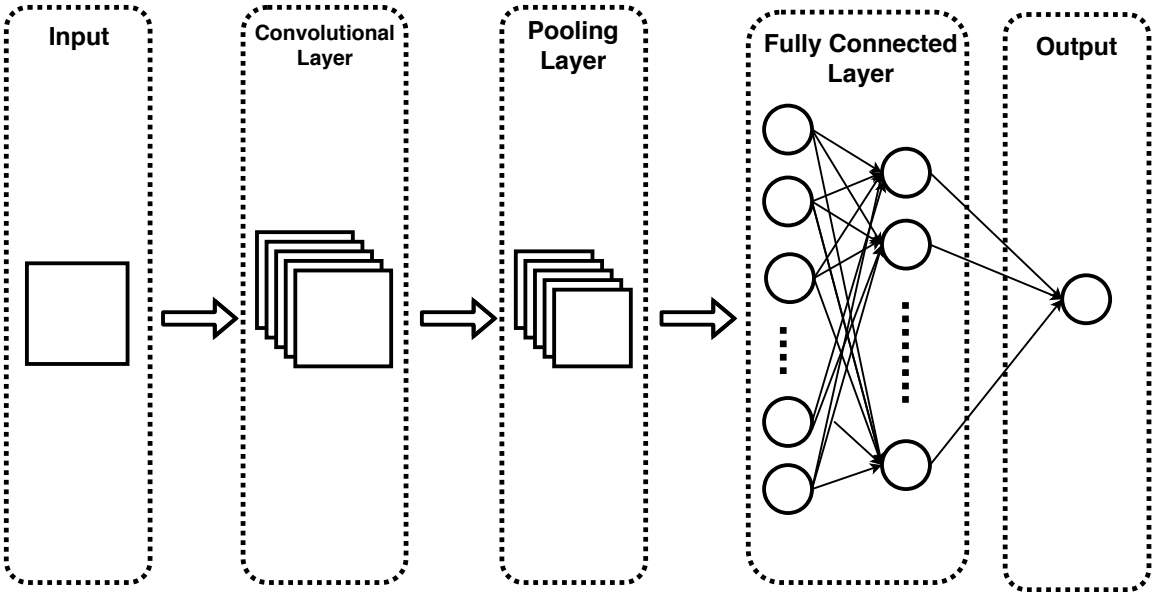

Figure 3.3: CNN structure

Convolutional layer and pooling layer in the CNN extract features from the time series and feed the data into the fully connected layers. The fully connected layers by using the learned information map the input vector to a corresponding output value.

Time Series Processing

Training set is normalized and input time windows are normalized as in the LSTM model. In the post processing a different method is applied to map the prediction points to original time series data point ranges. The equation 3.1 explain the technique used for mapping the prediction.

$$V_{new} = (V_{in} - P_{min}) \times \left[\frac{O_{max} - O_{min}}{P_{max} - P_{min}}\right] + O_{min} \qquad \text{(equation 3.1)}$$

V_{in} = predicted value
V_{new} = processed value
O_{max} = maximum value in training set
O_{min} = minimum value in training set
P_{max} = maximum prediction value for the training set
P_{min} = minimum prediction value for the training set

3.2.3 Hybrid configurations

Neural networks can be combined in several methods. In our approach we explore two linear combination structures and one ensemble combination structure of a LSTM network and a CNN. The structures in Figure 3.4, Figure 3.5 and Figure 3.6 shows the models which is explored.

LSTMCNN model

This is the first linear model we explore. The LSTM network and CNN is combined in the order which, the time series data is fed into the LSTM and then the output of this network becomes the input to the CNN.

The two networks, LSTM and CNN are trained individually by using the same set of training data. The output from the CNN become the final prediction of the model for a given input.

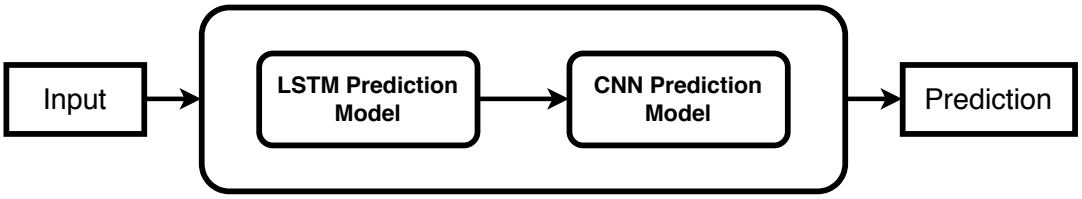

Figure 3.4: LSTMCNN structure

CNNLSTM model

In this model networks are combined linearly as same as the LSTMCNN model. The original time series data is fed to the CNN and the input for the LSTM is generated from the CNN predictions. Final prediction of the time series is obtained by the LSTM network. Training process of the networks are same as the previous model.

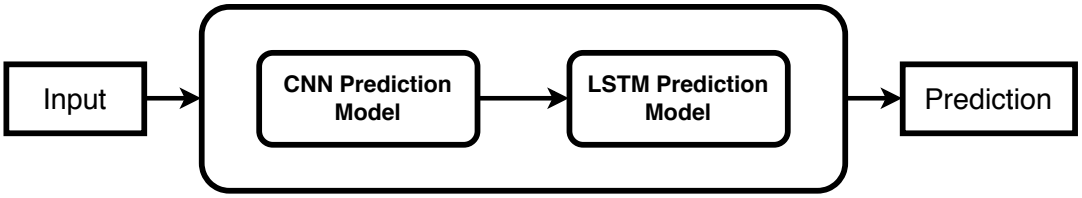

Figure 3.5: CNNLSTM structure

LSTMCNNFC model

This prediction model is developed using an ensemble combination of a CNN and a LSTM network. The main reason for this model is get more accurate prediction by combining the power of capturing long term dependencies using LSTMs and the power of CNNs concurrently.

In this model same as the previous linear models the LSTM and CNN networks are trained using the same data set. For the training the time series is partitioned into overlapping time windows. The same time widow technique is used when doing the predictions. Input time window for the both networks are similar.

For each time window the both models LSTM and CNN generate separate predictions and then these two predictions are combined to obtain the final prediction from the model.

Prediction Combining Layer

The process of combining the prediction results is conducted to extract details from from both predictions and make the final prediction more accurate. To get the most accurate prediction The information that obtained from each individual prediction model can be vary depending on the time time sequence.

The above mentioned issue is handled using a layer called prediction combining layer where we use a weighted multiplication technique (equation 3.2) to obtain more optimum predictions of the time sequence.

$$F_p = (P_l \times W_l) + (P_c \times W_c) \qquad \text{(equation 3.2)}$$

F_p = Final Prediction
P_l = Prediction from the LSTM model
P_c = Prediction from the CNN model
W_l = Weight for the LSTM prediction
W_c = Weight for the CNN prediction

The weights for the LSTM and CNN predictions can be adjusted depending on the time series. To get more accurate long term patterns the weight for the LSTM network prediction can increased. If more accurate predictions for the local patterns are required we can increase the weight for the CNN prediction value.

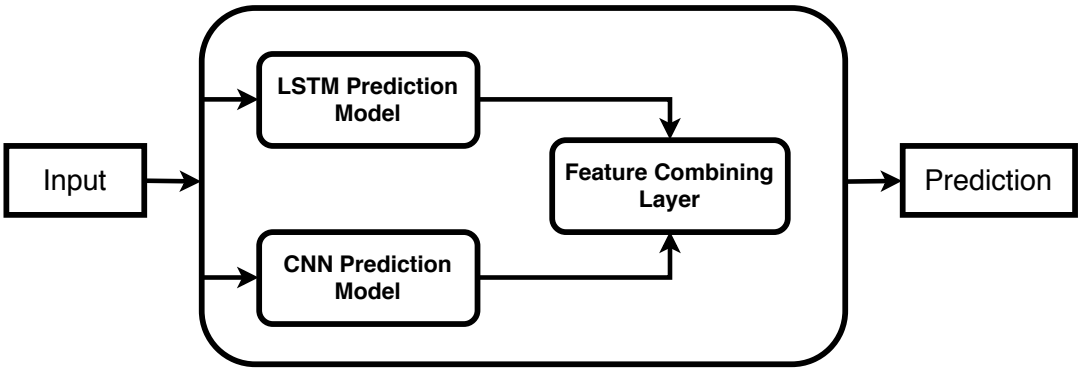

Figure 3.6: LSTMCNNFC structure

3.2.4 Predictions

Time Window approach

Input for the prediction model is given as a time window. A time series of length n is converted in to length m time windows. The value m depend upon the time sequence.

$$series = (t_1, t_2, t_3, ..., t_{\mathrm{n}})$$

$$inputs = ((t_1, t_2, .., t_{\mathrm{m}}), ..., (t_{\mathrm{q+1}}, t_{\mathrm{q+2}}, ..., t_{\mathrm{n}}))$$

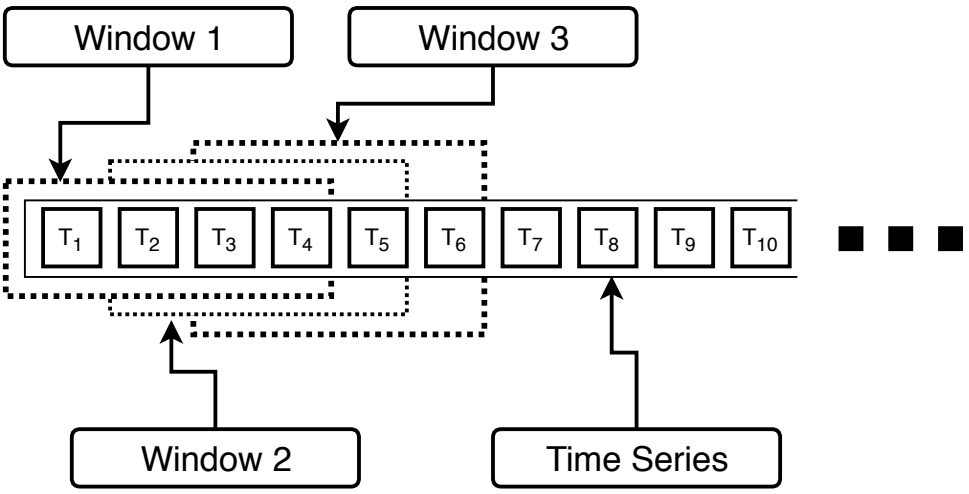

Figure 3.7: Input Time Window

Predicting Multiple Points

The prediction model take in a time window predicts the next point in the time sequence. To predict more than one consecutive point for a single input window a specialized technique is used.

In this technique initially one point is predicted for the given input time window. Then the input time window is updated by adding the predicted point to the input. Using this updated input window we can predict the next consecutive point. By using this technique several consecutive points ahead in time for a single time window. Figure 3.8 shows the visual representation of the technique used.

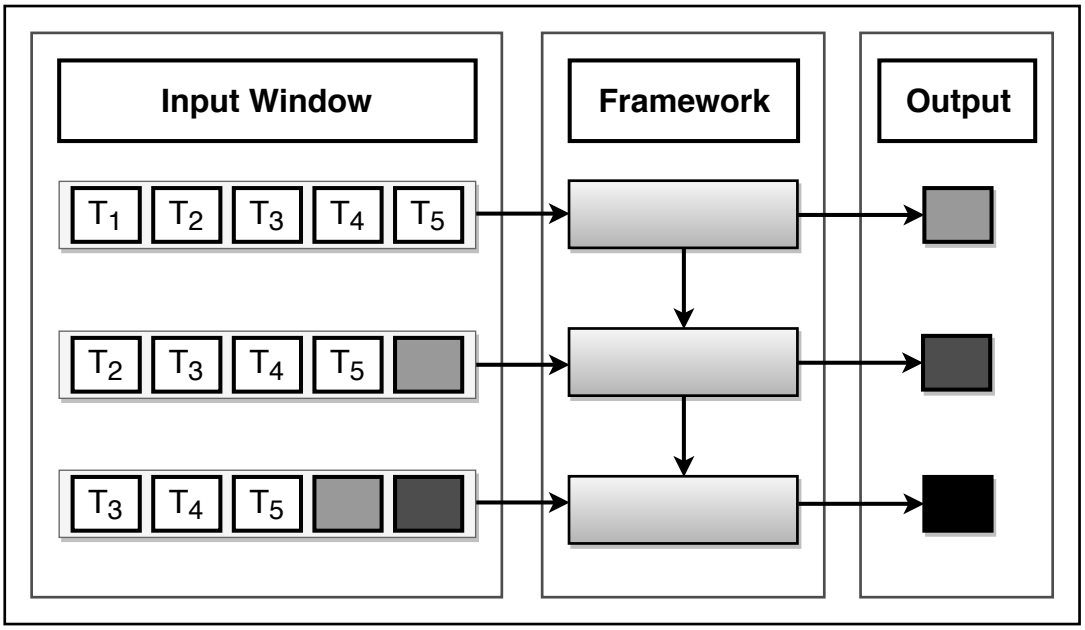

Figure 3.8: Multiple time steps prediction

When using this method, for a single time instance the prediction model generate multiple values. Therefore to make the prediction a single value of these multiple predictions for the same time instance the mean value is calculated.

3.3 Detection Module

To detect the anomalies the predicted time series data and the original time series data will be compared. To do the comparison an algorithm called dynamic time warping algorithm is used. The input for the algorithm is two time sequences. This algorithm gives a similarity value between the two sequences using the optimum global alignment (Figure 3.9). The similarity value is also called as the warp distance and less warp distance means more similarity between the two sequences.Predefined threshold value can be used to identify the anomalies in the sequence using this warp distance

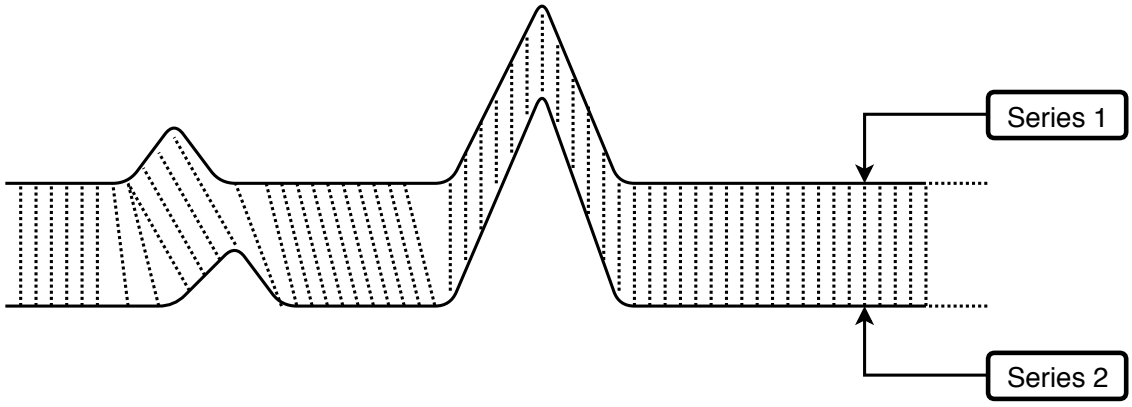

Figure 3.9: Alignment between two time sequences

3.4 Anomaly Detection Framework

Anomaly detection framework is developed by combining the prediction module and the detection module.

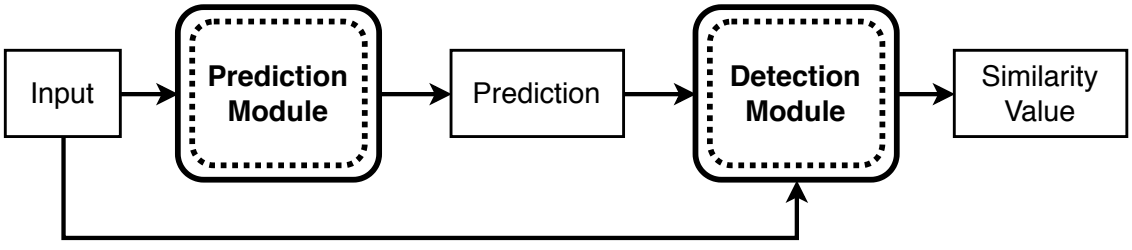

Figure 3.10: High level architecture of the framework

In the above Figure 3.10 it explains the high level architecture of the proposed

framework. The time series is partitioned into time windows and fed into the prediction model and predicted time series is compared with the original time series to get a similarity value.

For each time window the prediction model predicts the next time series point. Input for the detection model is also a time window. This time window is generated by the predictions. The obtained warp distance from applying the DTW algorithm can be used to identify the anomalies within the given time window.

3.5 Design Concerns

3.5.1 Normalization Problem

The time series is partitioned into overlapping time windows and normalized in the preprocessing step of the dataset. When the normalization is applied into the time windows in some specific cases two normalized time windows can be similar even the original time windows are not the same.

$$vector = (a, b, c, d)$$
$$magnitude(m) = \sqrt{(a^2 + b^2 + c^2 + d^2)}$$
$$input(normalized) = (\frac{a}{m}, \frac{b}{m}, \frac{c}{m}, \frac{c}{m})$$

Using the technique

$$v1 = [10, 10, 10, 10]$$
$$v1(normalized) = [0.5, 0.5, 0.5, 0.5]$$

$$v2 = [20, 20, 20, 20]$$
$$v2(normalized) = [0.5, 0.5, 0.5, 0.5]$$

The input vectors v1 and v2 are not the same, but normalized v1 and normalized v2 are similar. Therefore it affects the prediction of the model and predictions become less accurate.

Cannot use the model without normalizing the input vector because there are sigmoid functions used as activation function inside the LSTM memory cells and also in the CNN. when the input vector contain large values all the values are mapped into value '1' by the sigmoid functions

To overcome this issue the normalization technique is altered. In the initial technique we use the input time windows own magnitude to normalize. The new technique use the magnitude of the training data set in the normalization process.

3.5.2 Anomalous Input Data

As mentioned in earlier sections for the prediction model time series data is fed as time windows. When this sequences contain anomalies they are also fed to the prediction model within these time windows. When an input data window contain an anomaly the prediction become erroneous.

Therefore to overcome this issue we keep the information of our training dataset of the prediction model as the normal behavior of the time sequence within the prediction model. After each prediction the prediction is compared with the normal pattern and identify whether there is a significant difference between the prediction and the initial normal behavior.Therefore when such a difference is identified instead of adding the anomalous data point to the input window for the next prediction, the corresponding data point is matched using the information stored about normal behavior of the training dataset.

Chapter 4

Implementation

This chapter elaborates the implementation details of the proposed framework.

4.1 Long Short-Term Memory network

The LSTM network is implemented in C++ from the scratch. The network has memory blocks containing one memory cell in each block. Input layer is fully connected to the hidden layer. The weights for the network are randomly initialized. All the gates in a memory cell have bias values and they are also initialized randomly. Input for the network is given as vector. This vector contains data points of a time window of the time series. The training data size, learning rate and number of training iterations is depending upon to the dataset.

Listing 4.1: LSTM network

```
#include "LSTMnet/LSTMNet.h"

// Generating the LSTM
LSTMNet * lstm;
this->lstm = new LSTMNet(memCells,inputVecSize);
// Training the LSTM net
lstm->train(input, targetVector, trainDataSize, timeSteps,
    learningRate, iterations);
// Predictions
std::vector<double> * input;
input = new std::vector<double>[1];
double prediction;
prediction = lstm->predict(input);
```

4.2 Convolutional Neural Network

CNN is implemented in C++. Eigen library is used for matrix manipulations. Convolution layers, Activation layers, Pooling layers and Fully connected layers are implemented within the network. The number of Convolution layers, and Pooling layers and number of layers in the fully connected layers can be adjusted accordingly. sigmoid function is used as the activation function and non overlapping max pooling is used in the pooling layers. Network is trained using gradient descent method. weights for the filter matrices and the fully connected layers are initialized randomly. learning rate can be adjusted accordingly.

Listing 4.2: CNN network

```
#include "CNNet/CNN.hpp"

// Generating the CNN
CNN * cnn;
this->cnn = new CNN(dimensions, modelStruct->netStruct);
// Training the network
cnn->train(inMatArr, inLblArr, trainDataSize, iterations,
    learningRate);
// Predictions
Eigen::MatrixXd tstMatArr[1];
tstMatArr[0] = Eigen::MatrixXd::Zero(height,width);
double prediction;
prediction = cnn->predict(tstMatArr);
```

4.3 Feature Combining Layer

This layer is implemented to combine the prediction from LSTM and CNN. This is implemented within the prediction model. The final predictions of the model is generated by this layer.

Listing 4.3: Combinig the prediction results of LSTM and CNN

```
// LSTM predictions
input[0] = inVec;
for (int j = 0; j < numPredPoints; j++) {
    result = lstm->predict(input);
    input[0] = std::vector<double>(inVec.begin()+1, inVec.
        begin()+inputVecSize);
```

```
6       input[0].push_back(result);
7       lstmPredPoints[((i+inputVecSize+j)%numPredPoints)] +=
            result;
8   }
9   result = lstmPredPoints[((i+inputVecSize)%numPredPoints)]/(
        double)numPredPoints;
10  lstmPredPoints[((i+inputVecSize)%numPredPoints)] = 0;
11
12  // CNN predictions
13  for (int j = 0; j < numPredPoints; j++) {
14      prediction = cnn->predict(tstMatArr);
15      inVec = std::vector<double>(inVec.begin()+1, inVec.begin
            ()+inputVecSize);
16      inVec.push_back(prediction(0,0));
17      for (int a = 0; a < height; a++) {
18          for (int b = 0; b < width; b++) {
19              tstMatArr[0](a,b) = inVec.at(( a * width ) + b);
20          }
21      }
22      predPoints[((i+inputVecSize+j)%numPredPoints)] +=
            prediction(0,0);
23  }
24  prediction(0,0) = predPoints[((i+inputVecSize)%numPredPoints
        )]/(double)numPredPoints;
25  predPoints[((i+inputVecSize)%numPredPoints)] = 0;
26
27  // post process CNN prediction
28  val = prediction(0,0);
29  val = (val - predictMin)*((trainMax - trainMin)/(predictMax
        - predictMin)) + trainMin;
30
31  // combining the results LSTM and CNN
32  val = (result*lstmW + val*cnnW);
```

4.4 Dynamic Time Warping

DTW algorithm is implemented the using dynamic programming approach. The basic DTW algorithm with the time complexity of $O(n^2)$ is used.

4.5 Anomaly Detection Framework

The prediction module is developed by combining the LSTM network, CNN network and the feature combining layer. For the detection module DTW algorithm is implemented. Both These modules are combined to build the anomaly detection framework. The framework is able to generate five prediction model types, LSTM model, CNN model, LSTMCNN model, CNNLSTM model, LSTMCNNFC model. This is implemented as a dynamic library in C++.

Listing 4.4: Header File for the framework

```
#ifndef LSTMCNNET_HPP
#define LSTMCNNET_HPP

#include "LSTMPredictionModel.hpp"
#include "CNNPredictionModel.hpp"
#include "LSTMCNNFCPredictionModel.hpp"
#include "LSTMCNNPredictionModel.hpp"
#include "CNNLSTMPredictionModel.hpp"
#include "DTW.hpp"

#endif /* LSTMCNNET_HPP */
```

4.6 Network Parameters

The parameters for a model generated by the framework changes according to the prediction model type. The parameters for each model is stored in a C++ struct. Then this structure is passed as an argument when initializing the model.

Listing 4.5: Data structure for model parameters

```

struct NetLayers{
    int numCL = 0; // convolutional layers
    int numPL = 0; // pooling layers
    int numFCL = 0; // fully connected layers
    ConvolutionLayer * CL; // convolutional layer array
    PoolLayer * PL; // pooling layer array
    FCLayer * FCL; // fuly connected layer array
};

class ModelStruct {
```

```
12 public:
13     virtual ~ModelStruct();
14 public:
15     int trainingIterations; // training iterations
16     int trainDataSize; // train data size
17     double learningRate; // learning rate
18     // LSTM
19     int memCells; // number of memory cells
20     int inputVecSize; // input vector size
21     int predictions; // prediction points
22     int numPredPoints; // future points
23     std::string dataFile; // path to the data file
24     // CNN
25     int matWidth;
26     int matHeight;
27     int targetC;
28     struct::NetStruct netStruct;
29 };
```

Chapter 5

Results and Evaluation

Evaluation process is conducted in several steps. The framework is evaluated for all the five possible types of models LSTM model, CNN model, LSTMCNN model, CNNLSTM model and LSTMCNNFC model. The models are also tested for predictions with several time steps to future for a single time window input. Three datasets, sea level pressure[1], daily minimum temperature[2] and internet traffic[3] are used for the evaluation process. data without anomalies, data with artificial anomalies and data with known anomalies are used.

The framework consist of two modules prediction module and detection module. There are five possible prediction models that can e generated by the framework. Each prediction model is individually evaluated. Datasets for this evaluation is obtained by the UCI machine learning repository this evaluation

5.1 Prediction Module

The accuracy of the prediction model is calculated using the mean squared error of the predictions.

[1]The sea level pressure dataset for Darwin from the Climate Prediction Center

[2]Daily minimum temperatures in Melbourne, Australia, 1981-1990

[3]Internet traffic data (in bits) from an ISP. Aggregated traffic in the United Kingdom academic network backbone. It was collected between 19 November 2004, at 09:30 hours and 27 January 2005, at 11:11 hours. Hourly data

5.1.1 LSTM Model

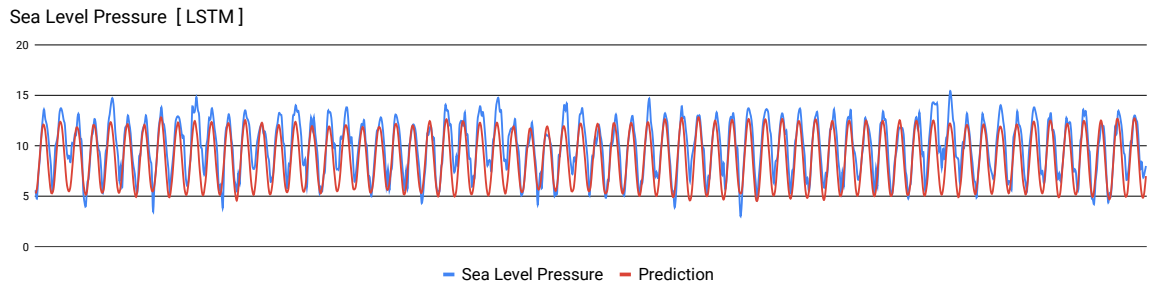

Figure 5.1: The sea level pressure dataset for Darwin from the Climate Prediction Center (LSTM Predictions)

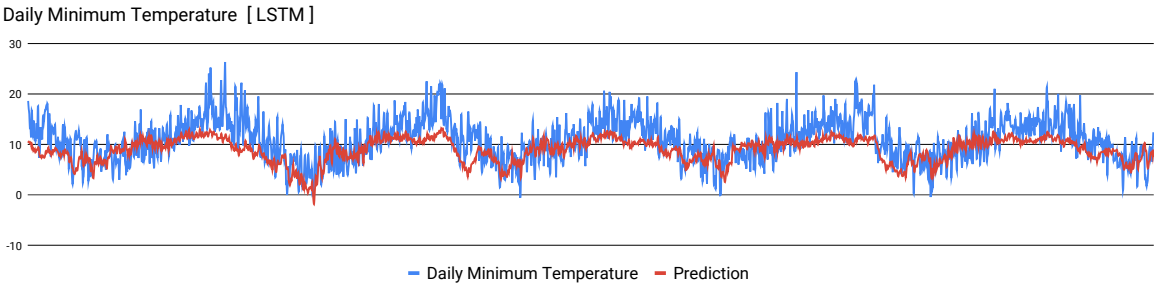

Figure 5.2: Daily minimum temperatures in Melbourne, Australia, 1981-1990 (LSTM Predictions)

5.1.2 CNN Model

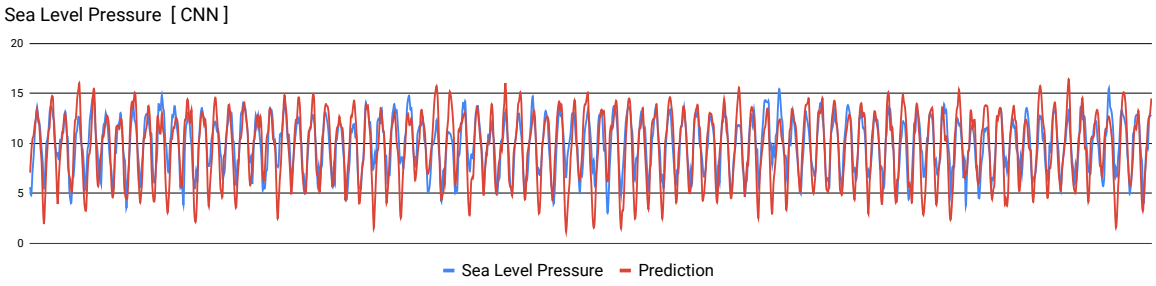

Figure 5.3: The sea level pressure dataset for Darwin from the Climate Prediction Center (CNN Predictions)

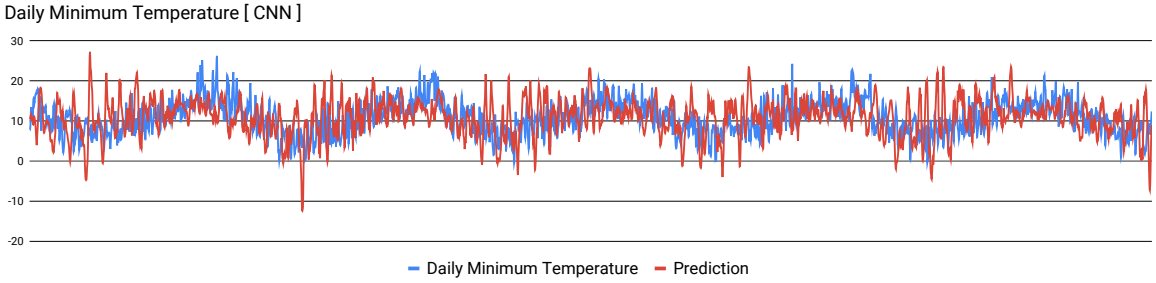

Figure 5.4: Daily minimum temperatures in Melbourne, Australia, 1981-1990 (CNN Predictions)

5.1.3 Hybrid Model

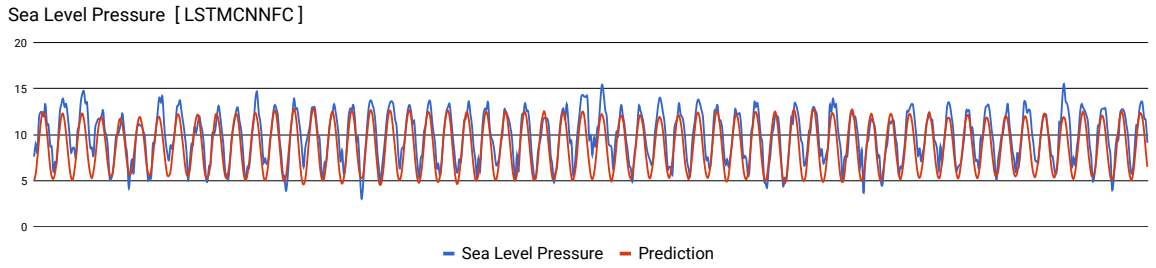

Figure 5.5: The sea level pressure dataset for Darwin from the Climate Prediction Center (LSTMCNNFC Predictions)

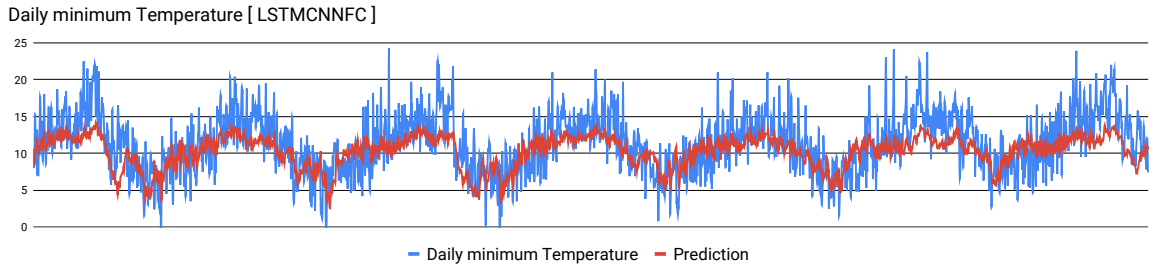

Figure 5.6: Daily minimum temperatures in Melbourne, Australia, 1981-1990 (LSTMCNNFC Predictions)

5.1.4 Mean Squared Errors

The Table 5.1 shows the mean squared errors of the prediction done by all the five the prediction models. Three datasets are used for the initial evaluation of theses models.

The most accurate predictions are identified by the mean squared error (MSE) of

	LSTM	CNN	CNNLSTM	LSTMCNN	LSTMCNNFC
Sea Level Pressure	2.00E-05	2.56E-05	1.53E-04	7.52E-05	**2.00E-05**
Internet Traffic	1.38E-04	2.30E-04	2.42E-04	2.03E-04	**1.34E-04**
Daily Temperature	2.87E-05	1.56E-04	9.05E-05	8.48E-05	**2.83E-05**

Table 5.1: Mean squared errors of prediction models

the predictions. The model with lowest MSE has the most accurate predictions. For all the datasets the lowest MSE is achieved by the hybrid model with the ensemble approach (LSTMCNNFC).

5.2 Feature Combining Layer

In this layer the predictions are combined using weighted multiplication. By altering the weight for each model we can alter the details of the predicting time series. Figure 5.7, Figure 5.8 and Figure 5.9 shows the results obtained for three different configurations of the prediction weights.

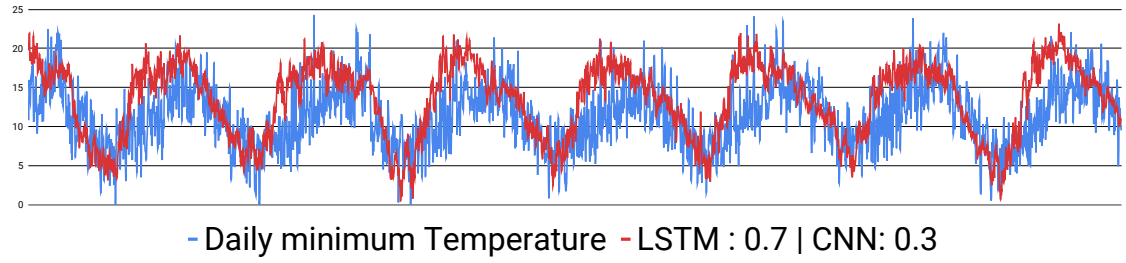

Figure 5.7: Minimum temperature dataset prediction from LSTMCNNFC model. More weight is applied to the LSTM's prediction. LSTM: 0.7 and CNN: 0.3

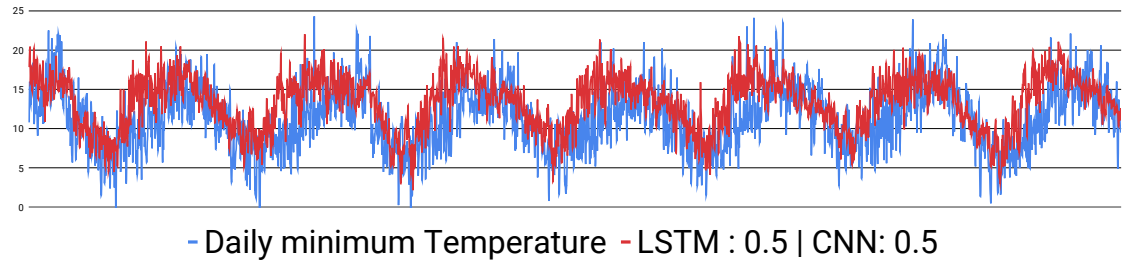

Figure 5.8: Minimum temperature dataset prediction from LSTMCNNFC model. Same weight is applied to the both predictions. LSTM: 0.5 and CNN: 0.5

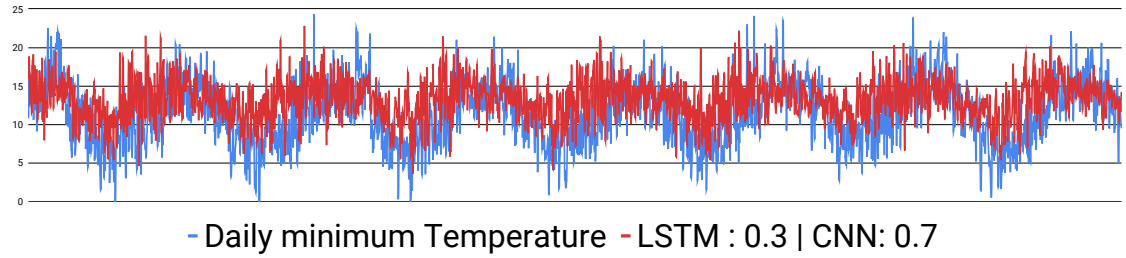

Figure 5.9: Minimum temperature dataset prediction from LSTMCNNFC model. More weight is applied to the CNN's prediction. LSTM: 0.3 and CNN: 0.7

CNN prediction and LSTM prediction effect for the final prediction in separate ways. When more weight is applied to CNN predictions it capture more details of the local patterns within the time series (Figure 5.9). When more weight is for the LSTM prediction (Figure 5.7) it captures more accurate global features.

5.3 Predicting Multiple Points

Multiple prediction points to the to the future for the same input time window was obtained to observe the accuracy. For this previously used datasets are used and tested for the LSTM and CNN models. Table 5.2 shows the MSE values for the LSTM model and CNN error values are showed in Table 5.3.

For the CNN prediction model all three datasets got the lowest error rate for one prediction in single input window. In the LSTM model also tests for two datasets showed the lowest error for single prediction. Therefore single prediction point approach used for the tests in the later part.

	1 Prediction	2 Predictions	3 Predictions
Sea Level Pressure	**2.00E-05**	2.20E-05	2.70E-05
Internet Traffic	1.38E-04	1.17E-04	**1.11E-04**
Daily Temperature	**2.87E-05**	4.01E-05	4.53E-05

Table 5.2: Mean Squared Error for multiple predictions of LSTM model

	1 Prediction	2 Predictions	3 Predictions
Sea Level Pressure	**2.56E-05**	3.10E-05	4.89E-05
Internet Traffic	**2.30E-04**	2.45E-04	2.61E-04
Daily Temperature	**1.56E-04**	1.83E-04	1.95E-04

Table 5.3: Mean Squared Error for multiple predictions of CNN model

5.4 Identifying Model Drifts

When the normal behavior of the temporal sequence shifts with the time if the changes are minimal and closer to the initial time series behavior the prediction model was able to adapt for the changing pattern and forecast the new pattern accurately. Figure 5.10 shows the prediction results for an artificially created dataset which has a model drift within the time sequence.

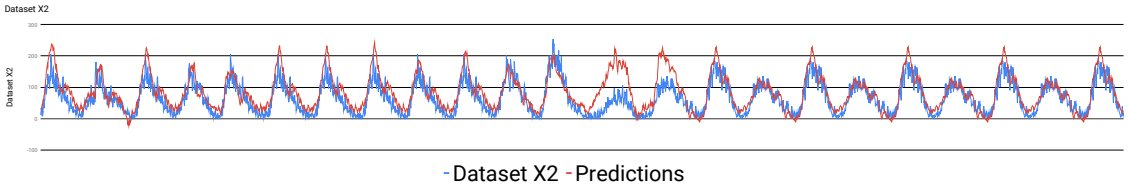

Figure 5.10: Artificial dataset. predictions from LSTMCNNFC model

But in the prediction module we keep track on the initial normal pattern to make the predictions accurate in time periods which contain anomalies. Therefore when the normal pattern shift we have to identify the new normal pattern. to identify this we use the changes in the DTW similarity value. The values obtained for each time window using the DTW algorithm is showed in Figure 5.11.

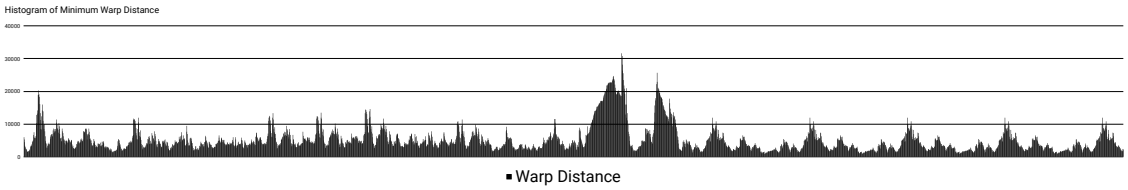

Figure 5.11: Histogram of Minimum Warp Distance

When the warp distance obtained by the DTW algorithm exceeds a given threshold the anomalies in the time series can be identified. After an anomalous point is

identified if there weren't any anomalies for a predefined period of time the prediction model update its normal behavior as the pattern starting after the previously identified anomalous points. ex: after detecting an anomaly, if an anomaly didn't occur for the next 1000 data points then the pattern starting after this anomaly become the normal pattern. length of this pattern is the length of the trained data set.

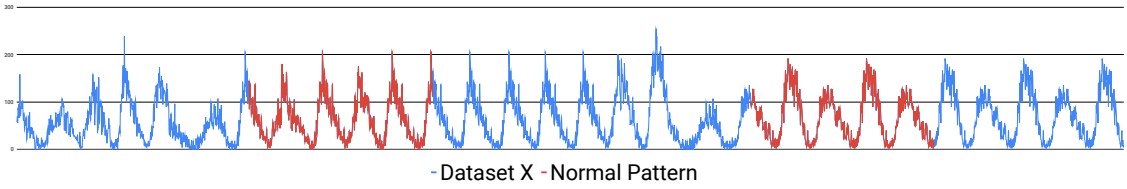

Figure 5.12: Identified normal behavior

5.5 Multivariate Predictions

The LSTM prediction model is also tested with a multivariate dataset[4]. The dataset describes measurements of a room and the objective is to predict whether or not the room is occupied. There are one-minute observations taken over the period of a few weeks. This is a classification prediction problem. There are 7 attributes in the dataset. date[time], Temperature, Humidity, Light, CO2, Humidity-ratio, Occupancy. For the training of the network only six attributes and 5000 data records are used. When training the prediction model the target values were +1 and -1 to state whether the room is occupied or not.

Two datasets are used for testing with 2000 records and 9750 records. Table 5.4 shows the data used for training and testing the model. Table 5.5 and Table 5.6 shows the results for two tests conducted.

After training the prediction model we use 0 as the prediction margin and test the model for two datasets containing 2000 and 9750 data records. For the both datasets we could achieve higher prediction accuracies (Tables 5.5 and 5.6). Then

[4] Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. Luis M. Candanedo, Véronique Feldheim. Energy and Buildings. Volume 112, 15 January 2016, Pages 28-39.

	Data	Occupied	Not Occupied
Training	5000	1141	3859
Test 1	2000	802	1198
Test 2	9750	2046	7704

Table 5.4: Data used for training and testing

margin	TP	TN	FP	FN	Accuracy
0	802	1137	61	0	96.95%
mid	802	1153	45	0	97.75%

Table 5.5: Results for the test 1 for the prediction model. 2000 data records are used for testing the model

we attempt to increase these prediction accuracies using several techniques.

When using the prediction margin as 0 for the test dataset with 2000 data points the prediction model misclassify 61 instances as the person is occupied but it is not. The reason for this is when the person leave the room for a small time period or it take some time for the room to adjust the environmental conditions back to normal after the person leaves.

Therefore we change the prediction margin and moved to it from 0 to +1 direction. Then we could see an increase in the prediction accuracy as show in the Figure 5.13

margin	TP	TN	FP	FN	Accuracy
0	2044	6979	725	2	92.54%
mid	2043	7124	580	3	94.02%

Table 5.6: Results for the test 2 for the prediction model. 9750 data records are used for testing the model

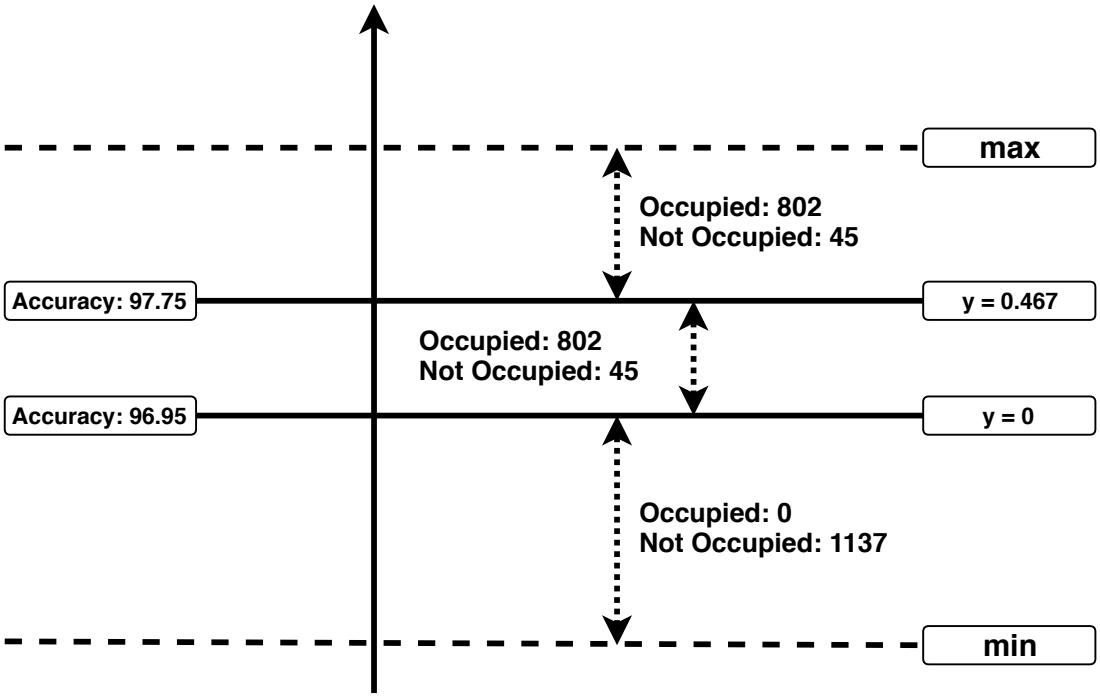

Figure 5.13: Classification margin

During the predictions sometimes the predictions were greater than or less than the +1 and -1 values. Therefore instead of increasing the prediction margin manually we get mid value of the maximum prediction and the minimum prediction and use this value as the margin. By using this technique to adjust the margin we were able to increase the accuracy of the predictions in the multivariate temporal sequence.

5.6 Predictions of time series data with anomalies

After evaluating the prediction module for datasets without any known anomalies we created some artificial anomalies in the same datasets and tested with prediction models.

When the time series has patterns with less local details Figure 5.14 prediction model predict the normal behavior, but when the pattern became complex Figure 5.15 the predictions from the model became less accurate. The main reason for this is the input time window for the prediction model contains anomalous data points.

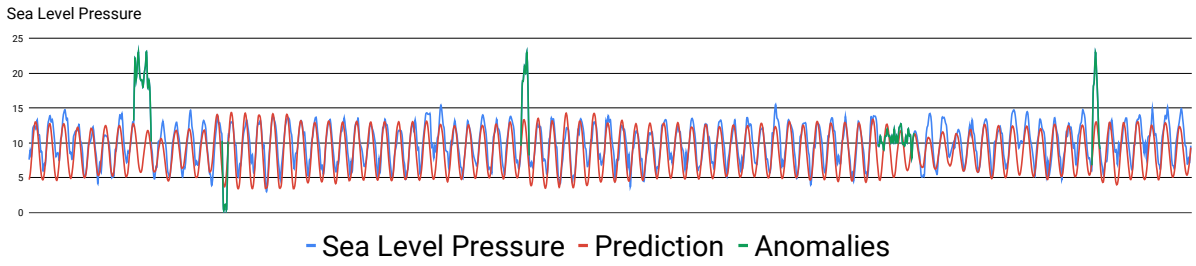

Figure 5.14: Sea level pressure dataset with artificial anomalies. predictions from LSTMCNNFC model

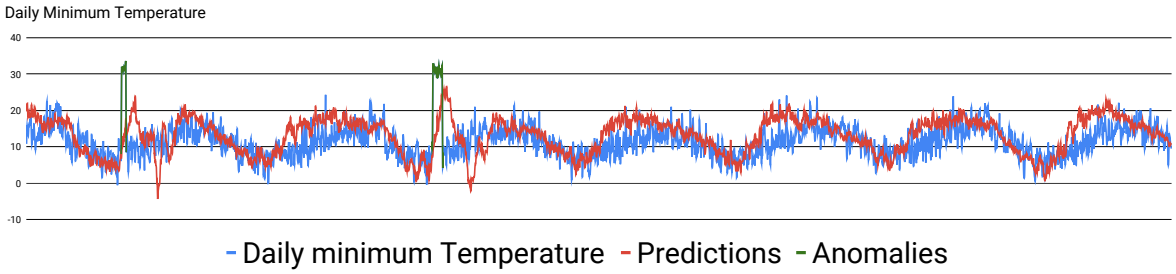

Figure 5.15: Minimum temperature dataset with artificial anomalies. predictions from LSTMCNNFC model

To overcome this problem we keep track on the pattern in the training data set as the normal behavior of the time series. When a prediction is done by the model the predicted data point is compared with the expected value. The predicted value and the expected value is compared and a predefined threshold is used decide whether the expected value is fed for the input time window for future predictions or the value should be obtained from the normal behavior of the time series. Figure 5.16 shows the results from using this technique.

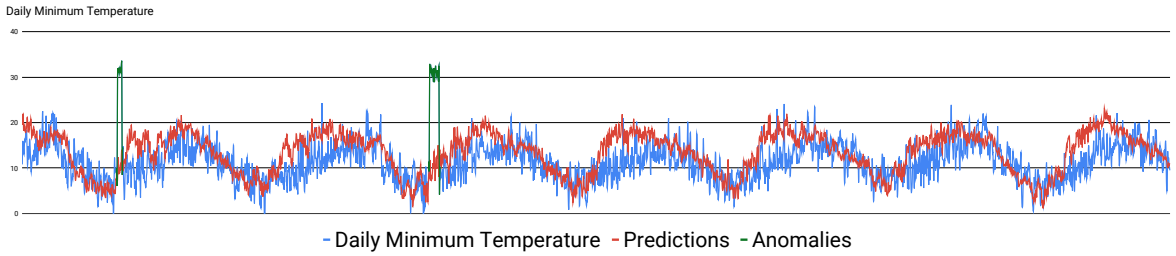

Figure 5.16: Minimum temperature dataset with artificial anomalies. predictions from LSTMCNNFC model

5.7 Anomaly Detection

Predictions from the prediction module and the original time series data points are compared using the DTW algorithm to identify the anomalies. The comparison is done in time windows. A warp distance is obtained for each time window and if this value is higher than a threshold value the time window contains an anomaly. Figure 5.17 and Figure 5.18 show the detected anomalies.

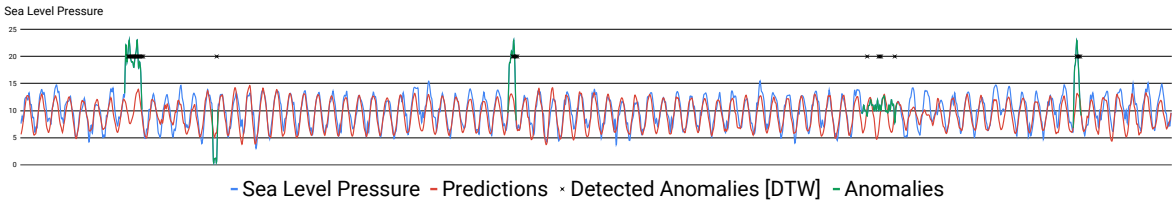

Figure 5.17: Sea level pressure dataset with artificial anomalies. predictions from LSTMCNNFC model. Anomalies detected using the DTW algorithm

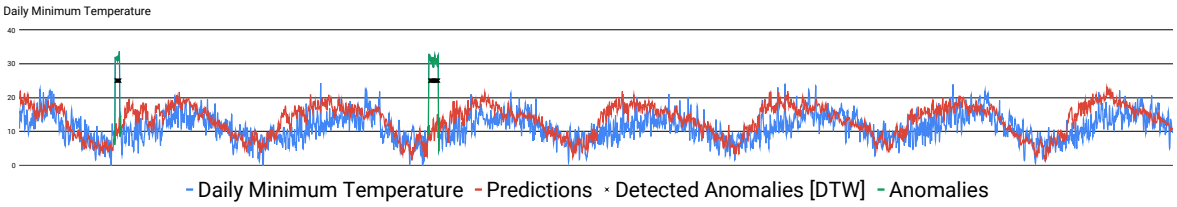

Figure 5.18: Minimum temperature dataset with artificial anomalies. predictions from LSTMCNNFC model. Anomalies detected using the DTW algorithm

The initial three datasets used for the evaluation of prediction models didn't contain any known anomalies. Therefore we evaluated our framework with Numenta Anomaly Benchmark which contain 58 datasets with known anomalies. Figure 5.19 and Figure 5.20 shows results form two datasets in NAB.

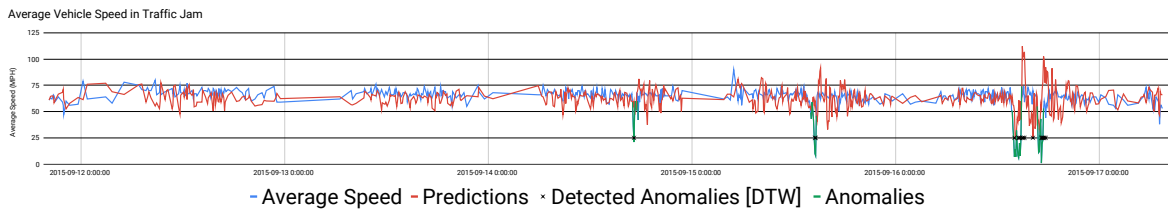

Figure 5.19: Average vehicle speed in a traffic jam dataset. predictions from LSTM-CNNFC model. Anomalies detected using the DTW algorithm

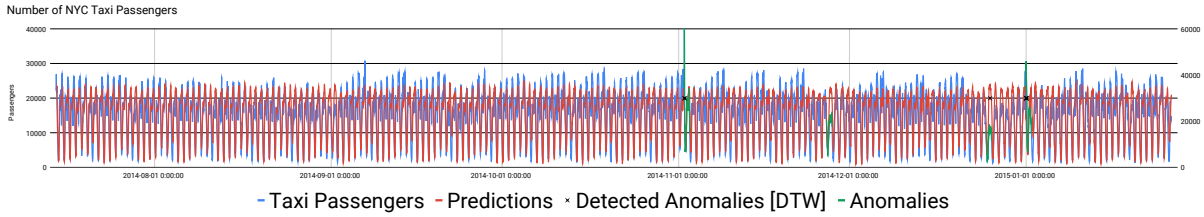

Figure 5.20: Number of NYC taxi passengers dataset. predictions from LSTMCN-NFC model. Anomalies detected using the DTW algorithm

5.8 Framework Evaluation

The proposed framework is evaluated using a Numenta Anomaly Benchmark (NAB) dataset. It is a novel dataset for evaluating algorithms for anomaly detection. The dataset consist of 58 different datsets with both artificial and real world with known anomalies.

The 58 datasets belong to seven categories, real tweets (A collection of Twitter mentions of large publicly-traded companies such as Google and IBM. The metric value represents the number of mentions for a given ticker symbol every 5 minutes), real traffic (Real time traffic data from the Twin Cities Metro area in Minnesota, collected by the Minnesota Department of Transportation. Included metrics include occupancy, speed, and travel time from specific sensors), real known cause (Ambient temperature in an office setting, Number of NYC taxi passengers, where the five anomalies occur during the NYC marathon, Thanksgiving, Christmas, New Years day, and a snow storm. The raw data is from the NYC Taxi and Limousine Commission, etc), real AWS known cloud watch (AWS server metrics as collected by the AmazonCloudwatch service. Example metrics include CPU Utilization, Network Bytes In, and Disk Read Bytes), real add exchange (Online advertisement clicking rates, where the metrics are cost-per-click (CPC) and cost per thousand impressions (CPM)), artificial with anomalies and artificial without anomalies. The Table 5.7, Table 5.8, Table 5.9, Table 5.10, Table 5.11 and Table 5.12 shows the results obtained for each category and Table 5.13 shows the summary of the results for the NAB.

	Anomalies	Detected	TP	FP	Accuracy	FDR
Twitter volume AAPL	4	5	3	2	75.00%	40.00%
Twitter volume AMZN	4	3	2	1	50.00%	33.33%
Twitter volume CRM	3	4	3	1	100.00%	25.00%
Twitter volume CVS	3	4	3	1	100.00%	25.00%
Twitter volume FB	2	2	2	0	100.00%	0.00%
Twitter volume GOOG	3	3	2	1	66.67%	33.33%
Twitter volume IBM	2	1	1	0	50.00%	0.00%
Twitter volume KO	3	5	3	2	100.00%	40.00%
Twitter volume PFE	3	2	2	0	66.67%	0.00%
Twitter volume UPS	4	7	4	3	100.00%	42.86%

Table 5.7: Detection accuracy for NAB dataset (Real Tweets)

	Anomalies	Detected	TP	FP	Accuracy	FDR
occupancy 6005	1	1	1	0	100.00%	0.00%
occupancy t4013	2	2	2	0	100.00%	0.00%
speed 6005	1	1	1	0	100.00%	0.00%
speed 7578	4	6	4	2	100.00%	33.33%
speed t4013	2	2	2	0	100.00%	0.00%
Travel Time 387	3	1	1	0	33.33%	0.00%
Travel Time 451	1	2	1	1	100.00%	50.00%

Table 5.8: Detection accuracy for NAB dataset (Real Traffic)

	Anomalies	Detected	TP	FP	Accuracy	FDR
Ambient Temperature	2	3	2	1	100.00%	33.33%
CPU Utilization	1	1	1	0	100.00%	0.00%
EC2 Request Latency	3	3	3	0	100.00%	0.00%
Machine Temperature	4	4	2	2	50.00%	50.00%
NYC Taxi	5	5	4	1	80.00%	20.00%
Rogue agent key hold	2	0	0	0	0.00%	0.00%
rogue agent key updown	2	0	0	0	0.00%	0.00%

Table 5.9: Detection accuracy for NAB dataset (Real Known Cause)

	Anomalies	Detected	TP	FP	Accuracy	FDR
EC2 CPU util 5f5533	2	2	2	0	100.00%	0.00%
EC2 CPU util 24ae8d	2	1	1	0	50.00%	0.00%
EC2 CPU util 53ea38	2	2	2	0	100.00%	0.00%
EC2 CPU util 77c1ca	1	2	1	1	100.00%	50.00%
EC2 CPU util 825cc2	2	2	2	0	100.00%	0.00%
EC2 CPU util ac20cd	2	2	2	0	100.00%	0.00%
EC2 CPU util c6585a	0	0	0	0	0.00%	0.00%
EC2 CPU util fe7f93	3	2	2	0	66.67%	0.00%
EC2 disk write byt 1ef3de	1	2	1	1	100.00%	50.00%
EC2 disk write byt c0d644	2	1	1	0	50.00%	0.00%
EC2 network in 5abac7	2	4	2	2	100.00%	50.00%
EC2 network in 257a54	1	1	1	0	100.00%	0.00%
ELB request count 8c0756	2	4	2	2	100.00%	50.00%
grok ASG anomaly	3	3	3	0	100.00%	0.00%
us-east-1 NetworkIn	2	2	2	0	100.00%	0.00%
RDS CPU util cc0c53	2	2	2	0	100.00%	0.00%
RDS CPU util e47b3b	2	2	2	0	100.00%	0.00%

Table 5.10: Detection accuracy for NAB dataset (Real AWS Cloud Watch)

	Anomalies	Detected	TP	FP	Accuracy	FDR
Exchange-2 CPC results	1	0	0	0	0.00%	0.00%
Exchange-2 CPM results	2	1	1	0	50.00%	0.00%
Exchange-3 CPC results	3	4	3	1	100.00%	25.00%
Exchange-3 CPM results	1	1	1	0	100.00%	0.00%
Exchange-4 CPC results	3	4	2	2	66.67%	50.00%
Exchange-4 CPM results	4	4	3	1	75.00%	25.00%

Table 5.11: Detection accuracy for NAB dataset (Real Add Exchange)

	Anomalies	Detected	TP	FP	Accuracy	FDR
Art daily flatmiddle	1	1	1	0	100.00%	0.00%
Art daily jumpsdown	1	0	0	0	0.00%	0.00%
Art daily jumpsup	1	1	1	0	100.00%	0.00%
Art daily nojump	1	0	0	0	0.00%	0.00%
Art increase spike density	1	1	1	0	100.00%	0.00%
Art load balancer spikes	1	1	1	0	100.00%	0.00%

Table 5.12: Detection accuracy for NAB dataset (Artificial With Anomaly)

	Anomalies	Detected	TP	FP	Accuracy	FDR
Real Tweets	31	36	25	11	80.65%	30.56%
Real Traffic	14	15	12	3	85.71%	20.00%
Real Known Cause	19	16	12	4	63.16%	25.00%
Real AWS Cloud Watch	31	34	28	6	90.32%	17.65%
Real Add Exchange	14	14	10	4	71.43%	28.57%
Artificial With Anomaly	6	4	4	0	66.67%	0.00%
Total	**115**	**119**	**91**	**28**	**79.13%**	**23.53%**

Table 5.13: Detection accuracy for NAB dataset (Summary)

The previous detection accuracy obtained for the NAB is 75.2%. Using our model we were able to get an accuracy of 79.13% for the dataset. The detection rate of these anomalies depend on the threshold values used in DTW similarity value. By using this threshold we can further increase the accuracy but it will result in increase of the fault detection rate.

Chapter 6

Conclusions

6.1 Introduction

This chapter includes a review of the research objectives and results, limitations of the current work and the directions for future researches.

6.2 Conclusions about research questions (aims/objectives)

The main aim of this research is to develop a framework for anomaly detection based on LSTM network prediction model. Initially we develop LSTM prediction model and to increase the accuracy we combine the model with a CNN based prediction model. The LSTM model was able to capture long term dependencies and CNN model contributed to the prediction of local pattern within the time series. For the analyzing of the predictions we use the dynamic time warping algorithm. Anomalies are detected using the warp distance

6.3 Conclusions about research problem

LSTM prediction model was able to capture more long term details of the time series whereas the CNN model capture details within small time intervals more accurately. Therefore the hybrid model was able produce better overall accuracy. By adjusting the contribution to the final prediction by each model we were able to optimize the prediction accuracy.

When the time series contain anomalous data points as we using time window

approach for the model input, when the input vector contain anomalous points the prediction become erroneous. To overcome this issue normal behavior from the training data is stored in the prediction model.

When predicting a time series if the time series has small deviations from the initial trained pattern prediction model was able to adapt for this pattern shifts and predict the data points according to the new behavior of the pattern.

6.4 Limitations

The normalization technique used for the vector normalization has limitation when the values in one vector is a multiple of an another vector the normalized vectors become similar. Then the predictions become less accurate.

Cannot use the model without normalizing the input vector because there are sigmoid functions used as activation function inside the LSTM memory cells and also in the CNN. when the input vector contain large values all the values are mapped into value '1' by the sigmoid functions

6.5 Implications for further research

In this research we use basic LSTM cell with an input gate, a forget gate and an output gate. We can optimize the LSTMCNNFC model by altering the memory cell structure in the LSTM network. e.i. replacing the input and forget gate with update gate. During the current approach we initialize the network hyperparameters randomly. Therefore a technique for tuning the hyperparameters in the prediction model can be explored.

The post processing step in the current hybrid prediction model contain a weighted multiplication and these weights are manually adjusted. The methods to auto adjust these weights according to the time series data can be explored.

The feature combining layer in the currently developed hybrid prediction model contain only one feature combining technique. New techniques that can be used to combine the outputs from the two prediction models could be explored further.

References

[1] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250–2267, Sep. 2014.

[2] F. Gers, D. Eck, and J. Schmidhuber, "Applying lstm to time series predictable through time-window approaches," in *Proceedings of the International Conference on Artificial Neural Networks*, ser. ICANN '01, London, UK, UK, 2001, pp. 669–676.

[3] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 4580–4584.

[4] K. Yamanishi and J. Takeuchi, "A unifying framework for detecting outliers and change points from non-stationary time series data." in *KDD*. ACM, 2002, pp. 676–681.

[5] L. Aguayo and G. Barreto, "Time series clustering for novelty detection: An empirical approach," 09 2007.

[6] M. L. Shahreza, D. Moazzami, B. Moshiri, and M. Delavar, "Anomaly detection using a self-organizing map and particle swarm optimization," *Scientia Iranica*, vol. 18, no. 6, pp. 1460 – 1468, 2011.

[7] P. Stefanovič and O. Kurasova, "Outlier detection in self-organizing maps and their quality estimation," *Neural Network World*, vol. 28, pp. 105–117, 01 2018.

[8] J. Ma and S. Perkins, "Online novelty detection on temporal sequences." in *KDD*, L. Getoor, T. E. Senator, P. M. Domingos, and C. Faloutsos, Eds. ACM, 2003, pp. 613–618.

[9] D. Dasgupta and F. Nino, "A comparison of negative and positive selection algorithms in novel pattern detection," in *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no.0*, vol. 1, Oct 2000, pp. 125–130 vol.1.

[10] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *ESANN*, 2015.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[12] "Understanding recurrent neural networks and long short term memory networks." [Online]. Available: colah.github.io/posts/2015-08-Understanding-LSTMs

[13] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct 2017.

[14] "Backpropagating an lstm, a numerical example using a memory cell." [Online]. Available: medium.com/@aidangomez/let-s-do-this-f9b699de31d9

[15] T. Lin, T. Guo, and K. Aberer, "Hybrid neural networks for learning the trend in time series," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 2273–2279.

[16] D. T. Shipmon, J. M. Gurevitch, P. M. Piselli, and S. T. Edwards, "Time series anomaly detection; detection of anomalous drops with limited features and sparse examples in noisy highly periodic data," *CoRR*, vol. abs/1708.03665, 2017.

[17] L. Bontemps, V. L. Cao, J. Mcdermott, and N.-A. Le-Khac, "Collective anomaly detection based on long short term memory recurrent neural network," 03 2017.

[18] Y. Zheng, Q. F. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *WAIM*, 2014.

[19] Z. C. Lipton, D. C. Kale, and R. C. Wetzel, "Phenotyping of clinical time series with lstm recurrent neural networks." *CoRR*, vol. abs/1510.07641, 2015.

[20] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, Oct. 2007.

Appendix A

Diagrams

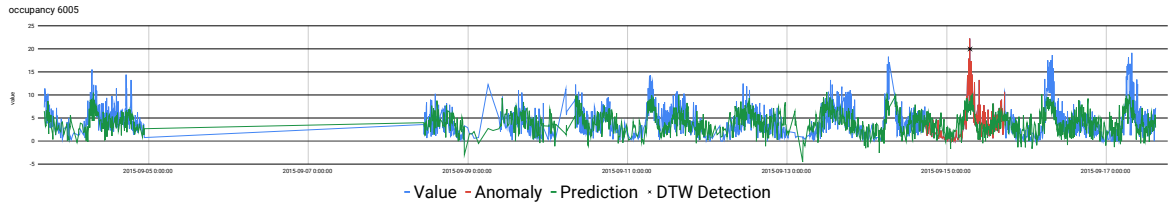

Figure A.1: Occupancy 6005

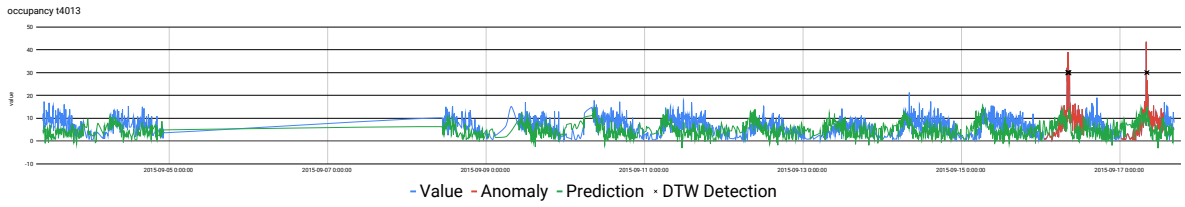

Figure A.2: Occupancy t4013

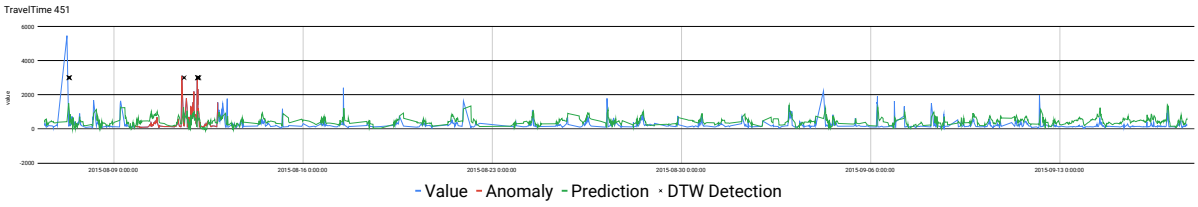

Figure A.3: Travel Time 451

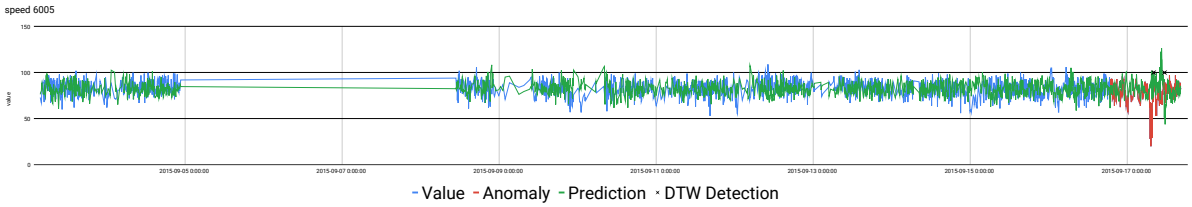

Figure A.4: Speed 6005

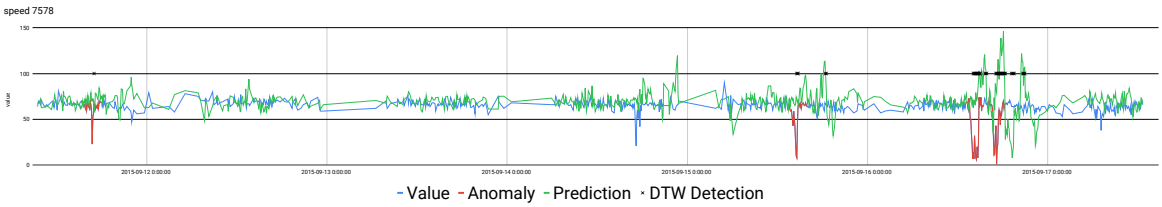

Figure A.5: Speed 7578

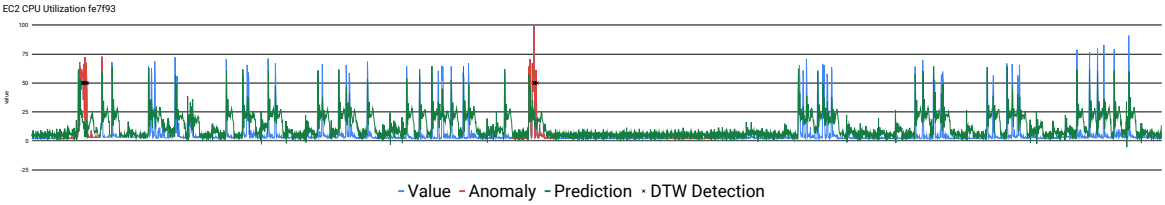

Figure A.6: EC2 CPU Utilization fe7f93

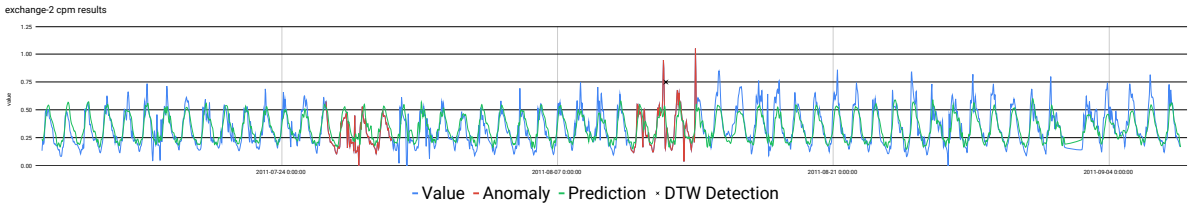

Figure A.7: Exchange-2 CPM Results

Appendix B

Code Listings

Listing B.1: Generating the anomaly detection models

```
/*
 * File:   main.cpp
 * Author: heshan
 *
 * Created on June 15, 2018, 4:38 PM
 */

#include <iostream>
#include <algorithm>
#include <vector>
#include <LSTMCNnet.hpp>

std::string datasets[] = {
/*0*/ "dataset1.txt",
/*1*/ "dataset2.txt"
}

std::string fileName = datasets[0];

// Initializing the structure
ModelStruct modelStruct;
modelStruct.trainDataSize = 600;
modelStruct.learningRate = 0.0001;
modelStruct.trainingIterations = 12;
modelStruct.numPredPoints = 1;
modelStruct.dataFile = "datasets/univariate/NAB/input/"+
    fileName;

// LSTM parameters
modelStruct.memCells = 10;
// CNN parameters
modelStruct.matWidth = 10;
```

```
32 modelStruct.matHeight = 2;
33 modelStruct.targetC = 1;
34 // Convolutional layer
35 struct::ConvLayStruct CL1;
36 CL1.filterSize = 2; // filter size: N x N
37 CL1.filters = 1; // No of filters
38 CL1.stride = 1;
39 // Pooling layer
40 struct::PoolLayStruct PL1;
41 PL1.poolH = 1; // pool size: N x N
42 PL1.poolW = 1;
43 // Fully connected layers
44 struct::FCLayStruct FCL1;
45 FCL1.outputs = 20; // neurons in fully connected layer
46 struct::FCLayStruct FCL2;
47 FCL2.outputs = 5; // neurons in fully connected layer
48 struct::FCLayStruct FCL3;
49 FCL3.outputs = 1; // neurons in fully connected layer
50
51 char layerOrder[] = {'C','P','F','F','F'};
52 struct::ConvLayStruct CLs[] = {CL1};
53 struct::PoolLayStruct PLs[] = {PL1};
54 struct::FCLayStruct FCLs[] = {FCL1,FCL2,FCL3};
55
56 modelStruct.netStruct.layers = 5;
57 modelStruct.netStruct.layerOrder = layerOrder;
58 modelStruct.netStruct.CL = CLs;
59 modelStruct.netStruct.PL = PLs;
60 modelStruct.netStruct.FCL = FCLs;
61
62 // Initializing the Detection model
63 LSTMCNNFCPredictionModel pm(&modelStruct);
64
65 // Training the networks in the model
66 pm.train();
67
68 // path for the target data file
69 std::string expect = "datasets/univariate/NAB/predictions/
     LSTMCNNFC/expect_" + fileName;
70 // path for the predicted data file
71 std::string predict = "datasets/univariate/NAB/predictions/
     LSTMCNNFC/predict_" + fileName;
72
73 // parameters for model outputs
74 int predictions = 4000;
75 int simVecSize = 5;
```

```
76 int marker = 50;
77 int similarityMargin = 350;
78 double lstmW = 0.5;
79 double cnnW = 0.1;
80
81 // getting predicted time series data points
82 pm.predict(predictions, expect, predict, lstmW, cnnW);
83
84 // getting anomalies identified by the model
85 pm.predict(predictions, expect, predict, simVecSize, marker,
       similarityMargin, lstmW, cnnW);
86
87 // getting DTW similarity values
88 pm.dtwSimilarity(predictions, expect, predict, simVecSize,
     lstmW, cnnW);
89
90 // getting predicted time series data points
91 // using normal behavior to identify increase the accuracy
     of predictions
92 pm.predictNorm(predictions, expect, predict, lstmW, cnnW);
93
94 // getting anomalies identified by the model
95 // using normal behavior to identify increase the accuracy
     of predictions
96 pm.predictNorm(predictions, expect, predict, simVecSize,
     marker, similarityMargin, lstmW, cnnW);
```