Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Master's Thesis

Comparative Evaluation of Feature Extraction Methods for Time Series Classification

Janhavi Puranik

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Paul G. Plöger
Prof. Dr. Sebastian Houben

March 2023

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Janhavi Puranik

Abstract

The importance of energy conservation and efficient energy utilization has significantly increased in today's world, leading to the need for monitoring systems and fault detection methods for these system. Deep learning has evolved as a potent technique for fault diagnosis, allowing for accurate and automated analysis of complicated data patterns and allowing for the rapid discovery and resolution of defects, eventually minimizing downtime and enhancing system dependability. This thesis aims to analyze Electrical Load data from UCI collected from 2011 to 2014 and study efficiency of available libraries and algorithms for anomaly detection and give a comparative evaluation on this based on selected metrics.

The data was analyzed to obtain knowledge on to the patterns present in the consumption of electricity. Preprocessing was done to handle the noise and changes in data due to daylight saving, along with this the data was labeled to study supervised techniques based on assumptions. The thesis implements both supervised and unsupervised algorithms for anomaly detection. There are five methods implemented in total out of which two are open source libraries. The libraries have been reviewed based on the implementation on the data. The methods implemented are Anomaly detection using LSTM Autoencoders and DeepAnt library and anomaly classification using Torch time library, LSTM and LSTM Autoencoder.

It was observed from the results that the unsupervised methods work better at detecting anomalies in comparison to supervised methods which might be due to the assumptions made in the labeling of data. The model of the torch time library overfits the data due to the invariant nature of Inception time module [1]. LSTM Autoencoders performs better than the LSTMs in classification of anomalies. DeepAnt and LSTM autoencoder perform similar to each other in detecting anomalies in data in unsupervised manner.

Acknowledgements

I would like to thank my supervisors Prof. Dr. Paul G. Ploger, Prof. Dr. and Sebastian Houben for providing the opportunity to work on this Research and Development Project. I would like to especially thank Prof. Dr. Paul G. Ploger for supporting and motivating me throughout the project. I would like to thank Vincent Scharf, Proneet Sharma, and Deepan for their immense support while conducting the experiments. I would like to also thanks to my colleagues for there valuable suggestions to improve this report. Finally, I extend my gratitude to my family and friends for their enduring support, undying inspiration, and endless encouragement.

Contents

List of Figures

1

Introduction

As population and economic development increase, so does the demand for electricity. With the invention of appliances and electrical equipment, as well as the increasing energy consumption in residential properties and industries, accurate forecasting and analysis of energy usage has become a crucial research area. Power suppliers face challenges such as fluctuating electricity and fuel costs, as well as the integration of high levels of renewable energy into the energy supply [11]. To effectively manage resources, accurate load forecasting is necessary, but the data used for forecasting may contain anomalies. These anomalies may be caused by power failures, communication issues, substation circuit breaker activations, or meter errors [12]. Thus, it is essential to identify and address these anomalies to ensure accurate forecasting of energy usage.

Timeseries can be univariate where the series represents record of one variable over time or multivariate where there are more than one variable record. In univariate series anomalies can be data points that do not follow the pattern in the time-series data or can be also be called as an outliers in general terms. In time-series specifically the it can be a data point that does not follow the seasonality or the trend pattern of the series. Along with this the statistical properties of this data point do not match the properties of the series. This can be the case for one data point of cn be also set of data points or a pattern. To identify an label such patterns require a deep knowledge of the domain for which the time-series was captured. Another challenge that is faced while labeling the anomalies is that, there is no exact notion on to what can be called as an anomaly since the sensor behavior cannot be fixed. This leads to building problem-specific applications and in turn generalization of solution is not possible. Another issue faced is that the labels to classify the data are not available, due to unclear specification of anomalous behavior. Therefore mostly unsupervised machine learning methods are applied to solve this problem. Along with this, the raw time series signal data from the sensors contains redundant information and noise which might not be relevant to train the model. Therefore the data is subjected to preprocessing such as filtering and feature extraction.

Recent fault diagnostic research has shown that finding features that transmit fault information are crucial for recognizing anomalies in the process. Feature extraction methods commonly include recognizing a fixed collection of characteristics from data depending on domain. These characteristics are often based on time, frequency, or time-frequency measurements. Signal amplitude values are used to compute time-domain characteristics. These characteristics are simple to establish and presume that the signal is

steady. This is one of the disadvantages of the method. To solve this problem, time series are frequently transformed to frequency domain, which allows for simple detection of oscillations in the series as well as their amplitudes and phases. The downside of this approach is that retrieved characteristics might be reliant on one another, and it is computationally intensive. Therefore an automatic way of identifying these features is required which can be done with help of deep-learning methods.

Through this work, we are trying to explore different approaches and libraries present to extract the relevant features from the sensor automatically to identify the abnormalities in data efficiently as well as analyze the patterns in data that might be useful for future predictions.

1.1 Motivation

Continuous monitoring and studying the pattern of electrical load consumption provides numerous advantages to the industrial and household sectors, such as discovering defective equipment, faulty measurement systems, or potential electricity theft. These variables can cause safety hazards and financial loss, so identifying an irregular pattern is critical.

Anomaly detection methods can be a helpful tool for energy providers to improve overall service reliability by improving their power generation and distribution systems while lowering maintenance costs. For example, preventive measures can be implemented.

Anomaly detection methods can assist utilities and energy providers optimize their power generation and distribution systems, decreasing maintenance costs, and improve overall service reliability. Utility companies, for example, can identify potential power outages or equipment breakdowns before they occur by identifying abnormal load consumption patterns, enabling them to take preventive measures and minimize downtime.

Furthermore, anomaly detection in load consumption can be used to detect electricity theft, an issue in many countries that can result in substantial financial losses for energy providers. Energy providers can reduce financial losses and increase revenue by discovering anomalous power usage patterns that indicate potential electricity theft and taking suitable action to investigate and prevent such activities.

With the increasing use of renewable energy sources, anomaly detection can aid in identifying and resolving issues related to energy storage and grid integration. It can, for example, be used to monitor battery performance and identify abnormalities in renewable energy generation that could jeopardize the stability of the power grid.

Identifying peak load times and balancing the load across the grid can help to optimize energy consumption. This can assist in reducing grid strain during peak demand times and avoid power outages.

Anomaly detection can spot abnormal energy consumption trends that may suggest variations in market demand. This can assist energy providers in adjusting their production and pricing strategies to shifting market circumstances.

Anomaly detection can give customers feedback on their energy consumption habits, encouraging them to minimize energy loss and adopt more sustainable behaviors. Customer satisfaction and loyalty may improve as a result.

1.2 Challenges and Difficulties

The following are some of the significant challenges connected with anomaly detection in time series:

1. An anomaly happening in a time series can be described in a variety of ways. A time series occurrence, a time series subsequence within a time series, or a complete time series may be anomalous in comparison to a collection of normal time series.

2. The precise length of the subsequence is frequently uncertain when identifying anomalous subsequences.

3. The lengths of the training and evaluation periods can vary.

4. It is difficult to identify best similarity/distance measures for various kinds of time series. Performance of simple measures such as euclidean distance are limited because they are susceptible to outliers. Also, they cannot be applied to time series of varying lengths.

5. Distinguishing abnormalities from noise is a difficult task. Because of this, performance of many anomaly detection algorithms are limited for noisy time series data.

6. In real-world applications time series are typically lengthy. And because of this, the computational complexity is high.

7. Many anomalous detection algorithms anticipate numerous time series to be of comparable scale, which is not true for the majority of data.

8. The accuracy of anomaly detection in time series data depends on data quality. Anomaly detection algorithms can be hampered by missing values, outliers, and noisy data.

9. Time series data can be high-dimensional, which means it includes many features that are challenging to analyze. This can make detecting significant abnormalities in data difficult.

10. Time series data frequently displays seasonal patterns, such as daily, weekly, or monthly cycles, which can make distinguishing between normal and abnormal behaviour difficult. These seasonal patterns must be taken into consideration by anomaly detection algorithms.

11. Detecting anomalies in time series data is frequently hampered by imbalanced data, in which the number of normal samples far outnumbers the number of abnormal observations. This can make it difficult to create precise models capable of detecting anomalies with high precision.

12. Time series data can be susceptible to concept drift, which occurs when the statistical properties of the data change over time. This can make developing models that are accurate over extended periods of time difficult.

13. Detecting anomalies in time series data can be extremely challenging, particularly for large datasets with many features. This can make developing algorithms that can scale to manage real-world applications difficult.

1.3 Problem Statement

Feature extraction is used to supplement the dataset's representation with additional descriptive variables. These factors are used to improve the prediction models' accuracy. The most often used approaches for extracting features from time series data are autoregressive models and transformation methods such as Fast Fourier Transform (FFT) [13] or Discrete Wavelet Transform (DWT) [14]. Where physical features are extracted as additional data attributes. These approaches rely on handmade elements and require human calculation. Libraries such as tsFresh [15] have been created to automate this procedure. The drawback of these approaches is that the complexity grows with the size of the data, and many duplicate features are generated. The complex time series data utilized in this study is not very expressive. It is not possible to detect anomalous behavior with a single characteristic. Manual selection of physical characteristics from prior data analysis is necessary to categorize the signals. Thus, the first approach is to extract characteristics such as statistical, temporal, and FFT from raw signals, which requires human computations and interpretation. More testing will be done to determine the amount of characteristics to be considered based on significance and data balance. One disadvantage of this strategy is that essential characteristics with high predictive value for the situation are neglected. To solve the shortcomings of domain specific methods, deep learning models such as Auto-encoders, Convolution neural networks (CNNs), Echo state networks (ESNs), and Long Short-Term Memory (LSTM) have been used to extract features from sensor signalsÂă [16]. The issue with directly applying such deep-learning approaches for feature extraction of the time-series dataset used in this study is that it is complicated. In such a circumstance, a transfer learning approach that allows the re-usability of existing models. The hyper parameters of a existing model can be reused or can be starting point for fine tuning the classification model. First, relevant models for the time-series data will be chosen. Selected models will be fine tuned using parameter optimization methods, and experiments will be carried out to find the best architecture for training the model. Results of the implemented models will be compared to results from the Torch time library based on relevant evaluation metrics. The goal of this comparative evaluation is to investigate a suitable automatic feature extraction model for the provided time-series dataset classification.-

2

State of the Art

According to the survey in [17] anomaly detection in algorithms used for time-series data can be broadly classified into supervised, semi supervised and unsupervised methods. Supervised methods require data to be labeled as normal and anomalous for training the model therefore, the ability of these algorithms to detect unseen data is comparatively lower in comparison to unsupervised methods. As most of the real time data available is not labeled unsupervised methods are used where there is no prior information on the data is required in training. In semi supervised algorithms the model is trained on to normal data and the points that deviate from the pattern are considered to be anomalous.

These methods cannot be directly applied on to raw data there are various transforms applied on the data to over come the challenges faced discussed in the previous chapter. Further sections will discuss on these methods present for preprocessing of time-series for anomaly detection algorithms and different techniques for the detection.

2.1 Data transforms

there are many challenges that are faced while dealing with time series data such as high-dimensionality, noise, scaling as discussed in chapter above [18]. Data preprocessing plays a crucial step in increasing the efficiency of detection techniques by reduction of dimensionality of data and preserving important information. Along with this by data transformation reduces the computation cost for various algorithms that use nearest neighbor approach for detecting similarity of pattern in the series in turn to identify the deviant pattern. Algorithms require uniform behavior of various time-series used as input to it thus normalization and scaling methods are employed [19]. Following are some preprocessing steps employed before the detection.

2.1.1 Aggregation

Aggregation means summation of set of data point, for time-series data it compressing set of points in the series to create a alternative representation of data. Usually this is achieved by averaging these points. There are many advantages of this method such as dimensionality reduction, elimination of repetitive data, replacing of missing values and elimination of noise and smoothing of signal. The transformation of data is in the time domain for this method. Disadvantage of this transform is that it can eliminate

important features that are crucial for detection of anomalies. Most of the studies explore aggregation and the effect of this transforms on time-series forecasting algorithms for financial domain. These studies investigate the effect of aggregation on forecasting methods such as [20], [21], [22].

Amongst these methods one of the most commonly used is suggested by [2] that is Piecewise Aggregate Approximation(PAA). In this method a time series with n dimensions is reduced to m dimensions of equal sized frames. Vector of PAA is produced by calculating the mean of data points that are part of these data frames. Consider a time-series $T = t_1, t_2, ...t_n$ of n length is transformed to a a m dimensional vector $T = t_1, t_2, ...t_w$. Elements of this vectors are calculated by following equation

$$t_i = \frac{m}{n} \sum_{\frac{m}{n} \times (i-1)+1}^{\frac{m}{n} \times i} c_j \tag{2.1}$$

Important consideration for calculation of these values is that m should be of a value closer to n or lager than it. If the value of m is very small then most of the information from time-series can be lost.

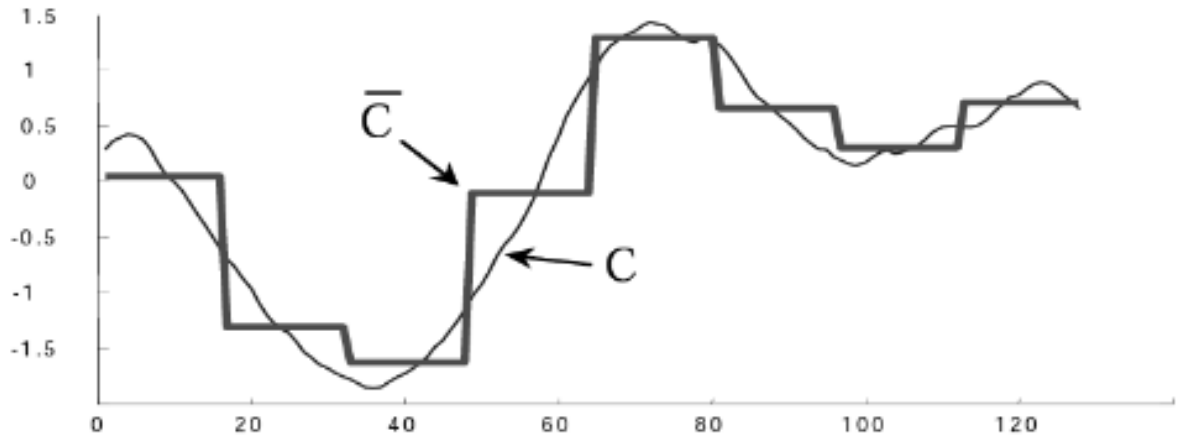


Figure 2.1: Visualization of PAA [2]

Another variation of this algorithm is provided by [23] called as Adaptive piecewise constant approximation. In this method instead of considering windows of same length for aggregation segments of multiple lengths is selected. The reason behind doing this is variation is not same along the series thus it considers small length segments in the area of high variation and lager segments for low variational area. First step of the algorithm is wavelet compressing that is finding optimal lengths of segments for the time-series. Then converting into vector by aggregating it on basis of mean value. Disadvantage of the Adaptive piecewise constant approximation of time-series data points is that the original series is not retained and the uniformity of the series is also lost. This makes this method unsuitable for anomaly detection

2.1.2 Discretization

This section has been written in reference to this [24] paper. Discretization's principal objective is to turn a provided time series into a discrete sequence of finite abbreviations. This transformation is focused on time-series of domain. The main reason for discretization is to make use of current symbolic sequence anomaly detection methods [19] and to increase computing efficiency [25]. This method has a draw back that it leads to information elimination. The algorithms that are used for discretization are broadly classified into supervised and unsupervised methods.

Unsupervised discretization methods

As known unsupervised methods are used when labeled data is not available.Various discretization methods are present amongst which Equal Width Discretization and Equal Frequency Discretization are most used [26]. other than this Kmeans clustering [27], SAX [2], Frequency Dynamic Interval Class [28] have been seen in many papers. These methods do not consider the order of time-series. Some of the methods are discussed in detail as follows:

1. Equal Width Discretization (EWD): is a fundamental approach of data discretization in which the observed values of a feature are separated into k equally sized bins. The range of characteristics is calculated using minimum and maximum values. The range is then divided by k to establish the interval size for each bin. Nevertheless, if the data is poorly distributed, EWD is susceptible to outliers and may yield overlapping ranges. The user determines the amount of intervals required, making EWD a parametric, static, and unsupervised technique [27].

2. Equal Frequency Discretization: is a data discretization approach in which observed values of a characteristic are arranged in increasing frequency order and separated into k bins, each of which comprises about nÂădata values with neighboring internal values. This method addresses the limitations of equal width interval discretization by separating the data into intervals of equal data points. Nevertheless, due to similar values being placed in the same interval, it may not always be feasible to construct exactly k equal frequency intervals. This approach has been used to discretized picture histograms by [27].

3. The K-means clustering approach is used for temporal data discretization because it aids a variety of applications in the identification of natural groups. In this technique, a euclidean distance metric is used for clustering the data into k clusters expressed by centroids. The clustering process starts with a random or informed selection of cluster centroids, and each data point is allocated to the cluster that is closest to it. The centroids are then calculated again as the mean value of each cluster, and the procedure is continued until no data points are reallocated or the k centroids do not change any longer. This approach is superior to EWD and EFD since it organizes data based on their qualities. It is, however, a parametric approach in which the user must supply the value of k as well as the seed value of the cluster, which defines the number of intervals for discretization.

Supervised discretization methods

When labeled dataset is available entropy based methods are use as given in [29]. As discussed above clustering based methods are used in context unsupervised methods but it can be utilized here as well as discussed in [30].Time domain information and characteristics along with labels is used to achieve lass-Attribute Contingency Coefficient algorithm [31].

1. Clustering-based discretization: is a method that utilizes both class labels and natural groupings to discretized data. The method uses K-means clustering with Euclidean distance and shared nearest-neighbor clustering algorithms to assign each data instance to a particular cluster or "pseudo-class." Entropy based discretization is then used to find the partition of a continuous feature. This partition minimises the class variable uncertainty learned on the discretized feature variable. This process is recursive until the end criterion is satisfied. The method is advantageous because it simultaneously uses both class and cluster-based information, resulting in better discretization of continuous variables [32].

2. The Class Attribute Contingency Coefficient (CACC) method: Top-down and supervised discretization method that utilises the CACC to evaluate the dependence of variables with each other. For every attribute, CACC first calculates the min and max values. After that it creates a set containing all the values in ascending order. It then partitions the attributes into intervals by finding the maximum CACC value for each plausible interval boundaries and adjacent boundaries midpoints [31].

2.1.3 Signal Processing

Many a times time-series have to be transformed into different domain space to find anomalies one such transform is frequency domain transform. Various algorithms are present to achieve signal processing such as Wavelet transform [33], Fourier transform [34] and Spectrograms. These algorithms are used to achieve computational efficiency by lowering the bound of dimensionality of the data.

One such algorithm is Fourier transform. It is a tool that converts the time-series into alternate representation, characterized by the sine and cosine functions of varying frequencies. According to this theory any complex function can be represented as the combination of periodic functions which is known as Fourier series.

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n cos(\frac{2\pi n}{T}x) + b_n sin(\frac{2\pi n}{T}x) \tag{2.2}$$

Any periodic time-series can be represented as sum of sinusoidal components which is given by the equation 2.2. Time series is transformed into Fourier series by measuring every possible cycle and returning the amplitude, offset and rotational speed for every detected cycle. One of commonly used methods to achieve the transformations for real world signals is Fast Fourier transform (FFT). It is a optimized algorithm for obtaining Discrete fourier transform. A signal's frequency components are derived from sampled data over a period of time. Each component of a signal is a single sinusoidal oscillation with its own amplitude and phase. Figure 2.4 shows a example of transformation. Another such method which is

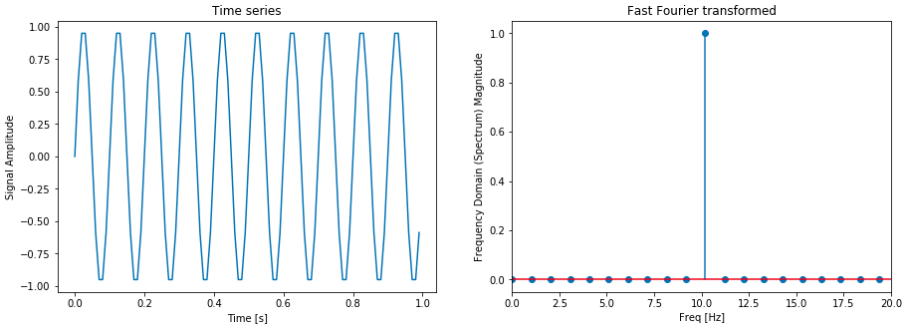


Figure 2.2: Fast Fourier transform example [**?**]

commonly used for audio data is Spectrogramm. It is a visualization method where the temporal and spectral components of a signal can be viewed simultaneously. A spectrogram can be constructed using the short-time Fourier transform, that is a time window is determined, and then translated across the field. By calculating the Fourier transform of each windowed field, the spectrogram is created. As it provides a analysis of time-frequency simultaneously it can be used to identify the nonlinearities in signal. Thus it proves a important tool for analysis of real world data such as sound. Figure 2.3 shows a example of spectrogram.

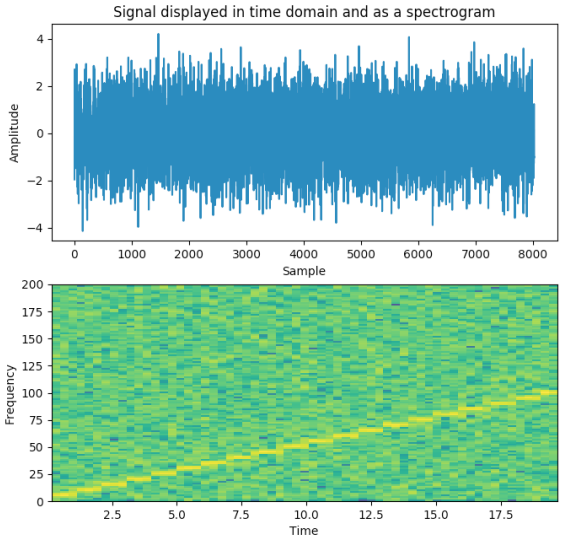


Figure 2.3: Spectrogram plot example [3]

According to this paper [3] consider we have a signal x of length N. If we create consecutive segments of x having length of m where $i << N$ and $X \in R^{i \times (N-i+1)}$ are the consecutive columns of the matrix. That is, $[x[0], x[1], ..., x[i-1]]^T$ is the first column $[x[1], x[2], ..., x[i]]^T$ is second and so on. Time indexing is used in this matrix. As we can observe that matrix X is repetitive representation of x. So the spectrogram

of signal is calculated by applying DFT to the columns of X and obtaining a matrix $\hat{X}$. Therefore,

$$\hat{X} = \bar{F}X \tag{2.3}$$

$$X = \frac{1}{i}F\hat{X} \tag{2.4}$$

In the matrix $\hat{X}$ columns are indexed by time and rows by frequency. So each value is representation of a point in frequency and time. $\hat{X}$ is invertible due to the transformations performed above. The information in this matrix is also highly redundant. The matrix is spectrogram and can be visualized with help of a plot.

Other than this [35] has used wavelet transform to extract the temporal features of the data and this performs better than methods like DFT and SVD as the wavelet transform considers different resolution of data in comparison to the others which only consider frequencies. Support vector machines have beren used in [36] which take set of vectors as a input to the algorithm instead of the whole time-series and create embeddings based on time lag. This algorithm allows the vectors to be projected on orthogonal space to remove low frequencies and thus only the high frequencies remain which are the anomalies.

2.2 Anomaly detection techniques

In the section above there was discussion on various methods present for time-series preprocessing for detection algorithms. In this section various detection techniques are discussed. Basic steps that all algorithms follow for anomaly detection are:

1. First is calculations of anomaly scores for the test dataset
2. Deciding the threshold based on aggregation of these anomaly scores calculated using either max values, mean or running average of them.
3. This threshold is then used to distinguish abnormal data from normal.

2.2.1 Window based

These techniques aim to pinpoint the cause of anomalies within a time series by dividing it into fixed-size windows or subsequences. The underlying assumption is that anomalies can be caused by one or more anomalous subsequences within the time series. In window based methods, the training and test time series are used to obtain the fixed-length windows [37]. This is done by sliding a window of size m, one or more symbols at a time. For the test time series, the anomaly score is evaluated by the accumulated score of its window. A generic window-based scheme according to [37] can be described as follows:

1. Extract p windows from each time series Si in the training database $S_{training} = s_1, s_2, s_3..., s_n$, where p can be calculated as $|s_i| + m - 1$. Similarly, divide each test time series Ti in the test database $S_{test} = t_1, t_2, ..., tn$ into $|ti| + m - 1$ windows.

2. For each test window, evaluate the anomaly score by $(A(t_{ij}))$ by measuring its similarity to the training windows using a distance metric such as Euclidean distance, Manhattan distance, or correlation.

If the window sliding is done one step at a time, than the number of sliding operations will be nearly equal to the length of the time series and this can be quiet computationally expensive. To address this, rather than step by step sliding, the next window is started after skipping certain âĂIJhâĂİ observations in the current window from the initial position. The âĂIJhâĂİ is called hop.

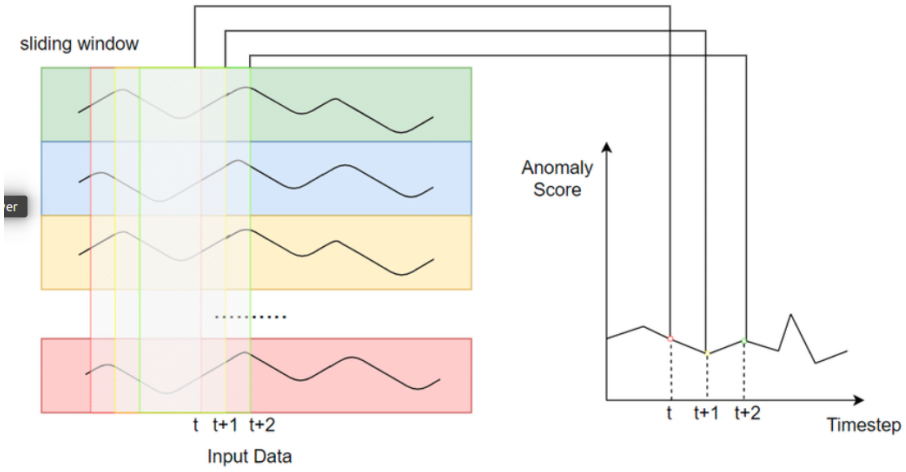


Figure 2.4: Sliding window real time anomaly detection example [4]

The size of the window is a critical parameter in the window-based technique and should be carefully selected to capture the anomaly. The optimal window size is dependent on the anomalous region length in the time series. Comparison of every training and test windows pair makes this method computationally complex, with a complexity of $O((nl)2)$. Usually the window based techniques are recommended forthe discord detection problem setting, which addresses the above-mentioned drawbacks of window size and computational complexity. Window-based techniques can capture all types of anomalies, including anomalous observations within a time series, anomalous subsequences within a time series, and anomalous time series as a whole. By breaking the series into small intervals, window based techniques can analyze if the observation or interval is anomalous or not. In the case entire series is anomalous, then all the intervals are anomalous and so, such anomalies are also effectively captured.

2.2.2 Proximity Based

Proximity based approaches uses the pair wise proximity between train and test data to calculate the anomaly score of the test data using a suitable distance or similarity kernel. This method assumes that the difference between the anomalous and normal data can be identified using proximity measure. Most commonly used proximity based approaches are kNN and Clustering. To calculate the anomaly score

different similarity metrics are used such as the Euclidean distance cosine, correlation and Dynamic time warping [38].

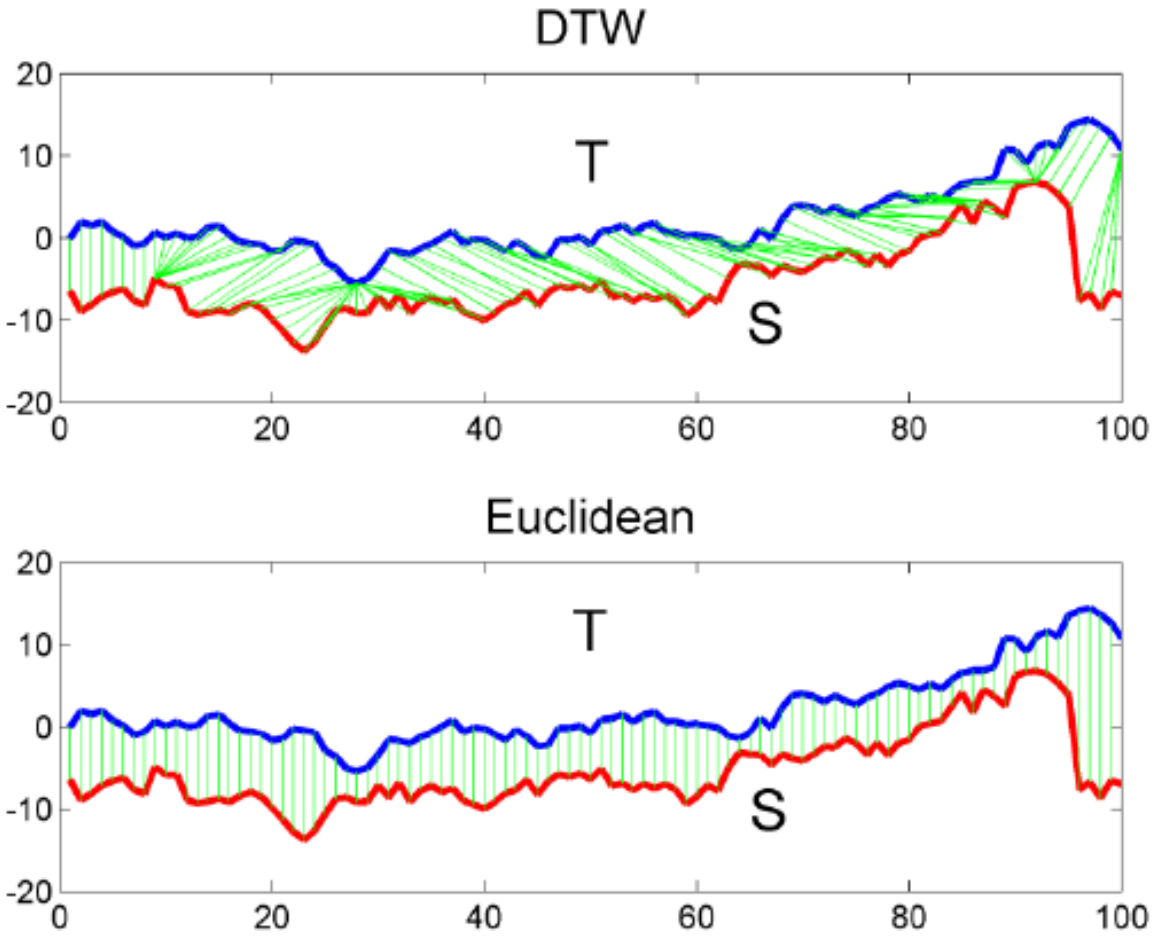


Figure 2.5: Visual depiction of distance calculation between two timeseries using DWT and Eucidean distance [5]

When the time-series are not of same size these similarity measurement techniques do not work and the algorithms assign high anomaly score to the nonlinearly aligned time-series with spikes at different timestamps. To overcome this DTW has been used which is more appropriate for calculating the distance between non linearly aligned or time data of different lengths. DWT can cause distortion of actual distance between the time-series by performing excessive matching [39]. The figure 2.5 shows difference between DTW and Euclidean distance measure.

To overcome the misalignment problem Cross correlation were introduced by [39] in which different time-series are generated in non synchronous manner by application of different conditions and times. In here different phase shifts of the pair of time data are selected and correlation between is of them is calculated. Maximum values from this correlation is used as the distance measure. Which implies that

the time-series with phase misalignment will have high correlation in turn indicating high similarity.

Problem of theses similarity based methods mentioned above is that these methods cannot detect the exact anomalous sequence in the data instead can only determine if the entire time-series is anomalous or not. The proximity measures are not easy to choose on which these methods are highly dependent. These methods are useful when the anomaly has to be detected in the time-series data or between many time-series to declare one of the instances to be anonymous.

To overcome this issue [39] suggested Periodic Curve Anomaly Detection(PCAD) which is a variant of k means clustering ng algorithm. The similarity between two time-series is calculated by maximum correlation between the time-series and the centroids that were calculated by PCAD.

2.2.3 Prediction based

Prediction based methods have been developed for usage in the statistical field, which has focus on to detecting single observation outliers. Assumption made by these algorithms is that normal time series follow the statistical rules while the anomalous do not. Key step to this procedure is that learning the normal instance parameters and estimating the probability of the test set generated from the process.

There are two major steps followed by this algorithm. First is to learn the history of n elements to predict or forecast the n+1 element. Second step is predicting all the observation using the predictive model trained above. Further the prediction error is calculated for each of the observations by calculating the diffrence between the original and predicted data. Some other parameters such as variance are also calculated for the model.

There are different prediction models present which can be used for detection of anomalies, some of the techniques present are as follows

1. **Moving Average**: This model creates a representation of time data by passing it through linear filter. "It produces the output at any time t using the previous input values $x(t-\tau), 0 \leq \tau \leq n$." It is also called as non-recursive filter. [40]

2. **Autoregression** : "The model generates representation of data by passing it through linear filter, producing output y(t) at time t using the previous output values $y(t-\tau), 0 \leq \tau \leq n$." It is also called recursive filter. [41]

3. **Autoregressive Moving Average**: Model generates representation of the data by passing it through recursive and non-recursive linear filter one after the other. It is combination of Moving average and Autoregressive models. [42]

4. **Autoregressive Integrated Moving Average**: It is extension of the previous model that calculates the Autoregressive Moving Average by not passing the whole time-series instead the difference between the consicutive values and the original series. [43]

5. **Kalman Filters**: Filters used in this algorithm are based on Markov property given by the equation:

$$x(t+1) = Hx(t) + q(t) \tag{2.5}$$

where the x(t) is the state of the system, H is matrix that describes casual link between current state and next state. Kalman filter for timeseries is described as :

$$y(t+1) = Hx(t) + K(x(t) - y(t)) \tag{2.6}$$

where K is the Kalman gain [44].

The limitation of prediction based methods is that the choice of history length is play important role in accurately detecting anomalies. Model will perform poorly if the history length is less than the cycle length and if the history is of length higher than the cycle length then the dimensionality of data increases that increases the computational complexity. In addition to this the data might be subjected to dimensionality curse if the size of history is high.

The technique of prediction models are highly dependent on to the right process and estimating its parameters accurately because of the assumption that data is generated from statistical process. If this assumption is not fulfilled then the algorithm fails to identify anomalies.

There are several advantages of this method such as it can capure all types of anomalies by usage of dynamic history. They are a powerful tool if the right size of history is chosen and the data is produced by statistical process.

2.2.4 Deep-learning

Unsupervised learning techniques can be used for anomaly detection of time-series to address the shortcomings of supervised approaches [45]. Deep learning approaches are also effective at modeling complicated real-time data. Deep networks are meant to distinguish the elements that influence input data variance at lower levels and combine representations at higher layers. As noted in the study [46], different deep-learning models have been examined for feature extraction of time series data, the most generally employed being Autoencoder. Many implementations, such as [47], [48], are available for modeling time-series data using LSTMs and Recurrent neural networks (RNNs), as these models are ideal for sequential data.

The problem with RNNs is that they have difficulties learning long-term deficiencies because of the vanishing gradient problem [49], which LSTMs can fix. The approach given here lacks an effective signal processing unit, and data labels are ignored. To solve the pattern ignorance exhibited in DBN, CNN, a common deep neural network architecture for image classification data, can be applied [36] [50].

This implementation states that CNN can serve as a competitive tool of feature learning and classification in comparison to the state of art feature engineering techniques. Overview of methods that can be used for the feature engineering of signal data is shown in figure A.1.

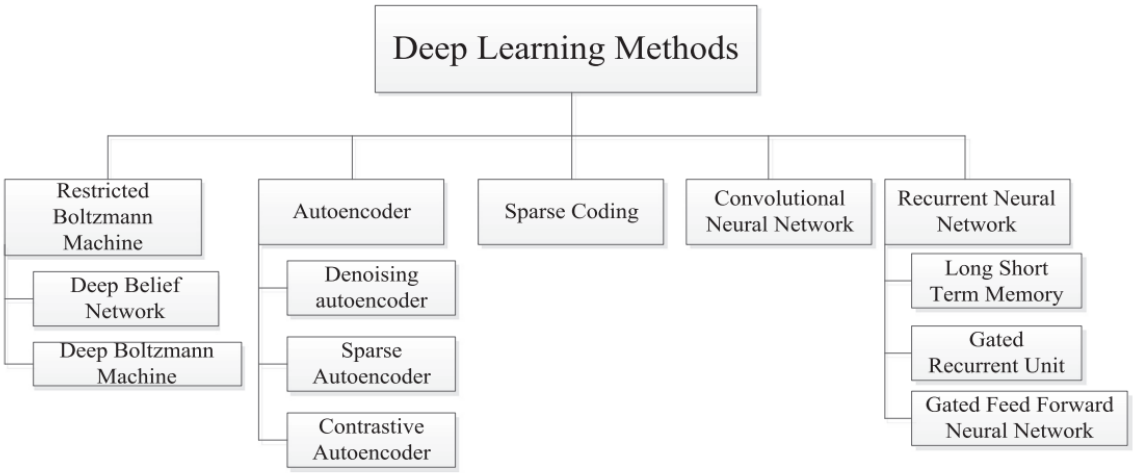


Figure 2.6: An overview of the different deep learning approaches for time series anomaly detection [6]

3

Methodology

3.1 Dataset

The dataset selected for testing the experiments conducted in this thesis is Electrical Load dataset from UCI database. This data was collected from different resources in one of the cities of Portugal. The data is contain information of electrical consumption of four years from 2011 to 2014 for 370 customers. The data is collected at interval of 15 minutes and the values recorder are in kilo watt unit. There are some customers that have been added after 2011 and thus, the values of consumption is marked zero for them for this year. There is time changes in March and October in which data recorded has 23 hours and 25 hours of the consumption values. This needs to be handled in data preprocessing. The data can be used for various tasks such as forecasting, clustering and anomaly detection. In this work it has been used for anomaly detection purpose. The data can be found here [51].

3.2 Data Analysis

As stated in above section there are 370 clients and data is collected every 15 minutes per day therefore, the number of data points for each clients is 140256. There are 96 values recorded per day and is indexed by date time in format of $yyyy - mm - ddhh : mm : ss$. Figure 4.1 gives a example of electrical consumption for a client.

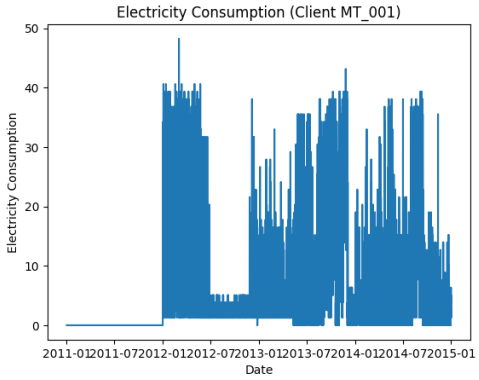


Figure 3.1: Electrical consumption for client MT001

As it can be seen the time-series shows data consumption for a client from 2011 to 2014. It contains noise that has to be filtered. Along with this re-sampling of data has to be done to get a clearer view. Also as mentioned above some of the clients were added after 2011 this client is amongst them as the values for 2011 are observed to be zero.

To analyze the data further the data is re-sampled to hourly consumption and averaged over time. This will help to analyze the seasonality of data properly over a period of time. In figure 4.2 it can be seen that

1. Power consumption is zero for the client in 2011 may be it is a client that was added after 2011.
2. Consumption of electricity is seen high in 2013 and 2014 in comparison to 2012.
3. A pattern can be noticed in the graph that the consumption from August of each year to January can be seen low in comparison to other months.

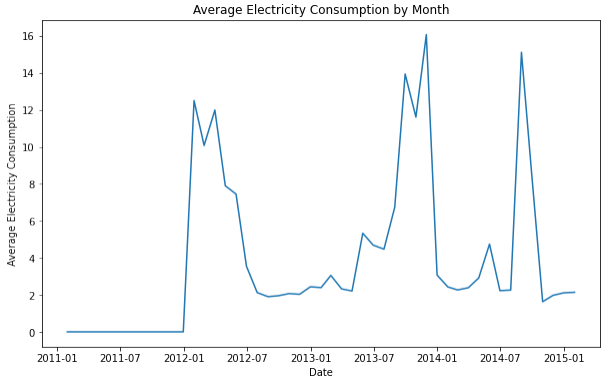


Figure 3.2: Average yearly electrical consumption for client MT300

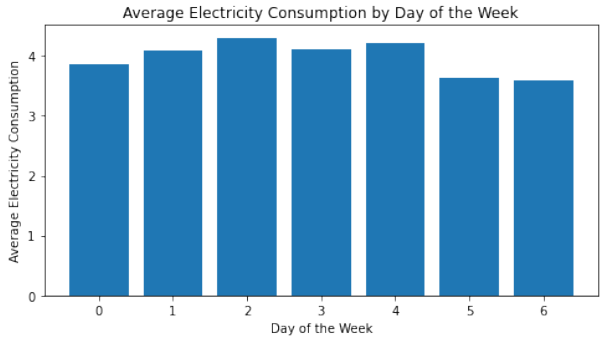


Figure 3.3: Average weekly electrical consumption for client MT300

To look for the weekly patter the average is done for weekly data in figure 4.3. Following is observed from the bar plot:

1. The bar plot gives the average electrical consumption for a client from Monday which is considered 0 here to Sunday that is 6.
2. Consumption is high for the client on 2 and 4 day that is Wednesday and Fridays.
3. There is not much difference between daily consumption of power.

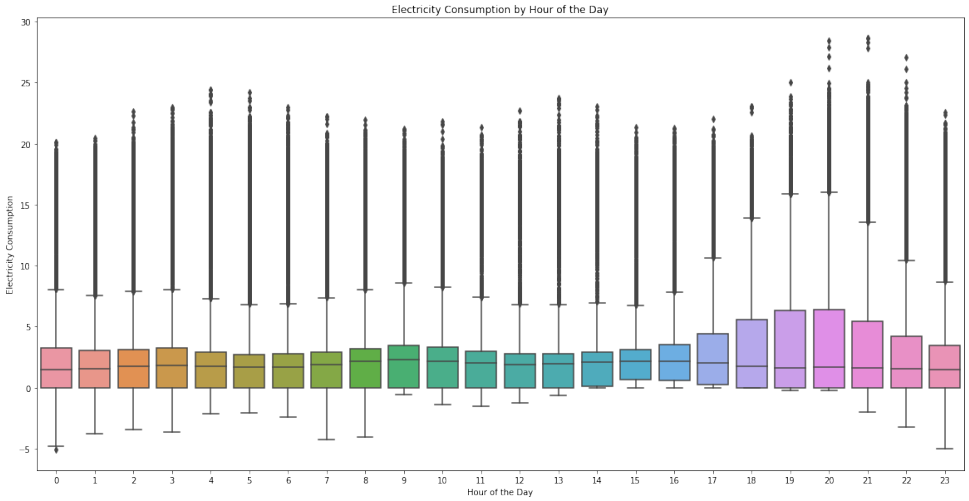


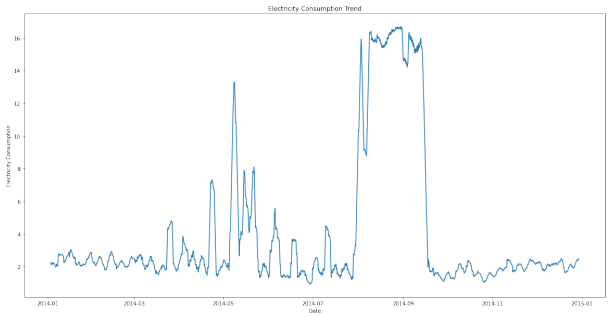
Figure 3.4: Hourly electrical consumption for client MT300

To analyze the consumption pattern of each day box plots are plotted for one of the clients which is shown in figure 4.4 and following observations are made:

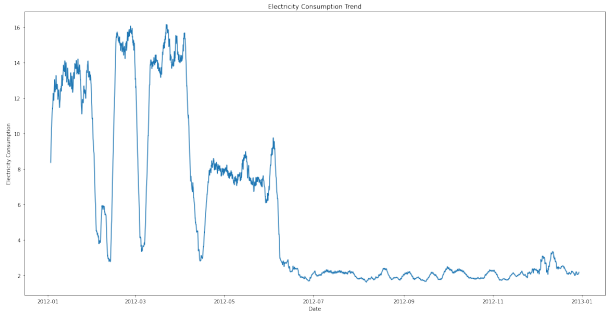
1. The x axis of the box plot contains hourly measurements from 0 to 23 that is from 12 night to 11 next night with duration of 24 hours.
2. The consumption is more in day in comparison of night.
3. Looking a the plot the variation of consumption shows that this might be electrical consumption of a house. As people are mostly at home

To study does the if there is any similarity of trends over the years or has it changed in figure 4.1. Following can be observed from the graphs:

1. The client does not follow similar trend for year of 2012 and 2014 the reason can be introduction of appliances that have high energy consumption or increase in number of them.
2. Energy consumption from Jan 2012 to May 2012 has been very high where as in 2014 the consumption is high in month of May and September.
3. Similar trend pattern is energy consumption is low from October to December for both the years it is plausible that energy consumption is low in winters.



(a) Trend for 2014



(b) Trend for 2012

Figure 3.5: Trends for client MT300 for year 2012 and 2014

In conclusion the consumption rate is high in 2013 to 2014 showing the introduction of new appliances and technology that has lead to more power consumption than before. Wednesdays and Fridays are high consumption days one observation that can be drawn for this Friday is a week end and power consumption is more on weekends in comparison to other days. Also Night consumption is more which is shows a noticeable patterns in household.

3.3 Data preprocessing

3.3.1 Data Filtering

The main reason behind filtering is to eliminate noise from the signals. This can be done by use of various filters which allow only certain frequencies to pass through then and discard rest of the data. This preprocessing step helps in achieving better training accuracies of machine learning algorithms. There are three mostly used filters:

1. Low pass: Eliminates frequencies higher than cutoff frequency and allows other lower than cutoff.
2. High pass: Eliminates frequencies lower than cutoff frequency set and allows the higher frequencies.
3. Band pass: It is combination of both low and high pass fitters.

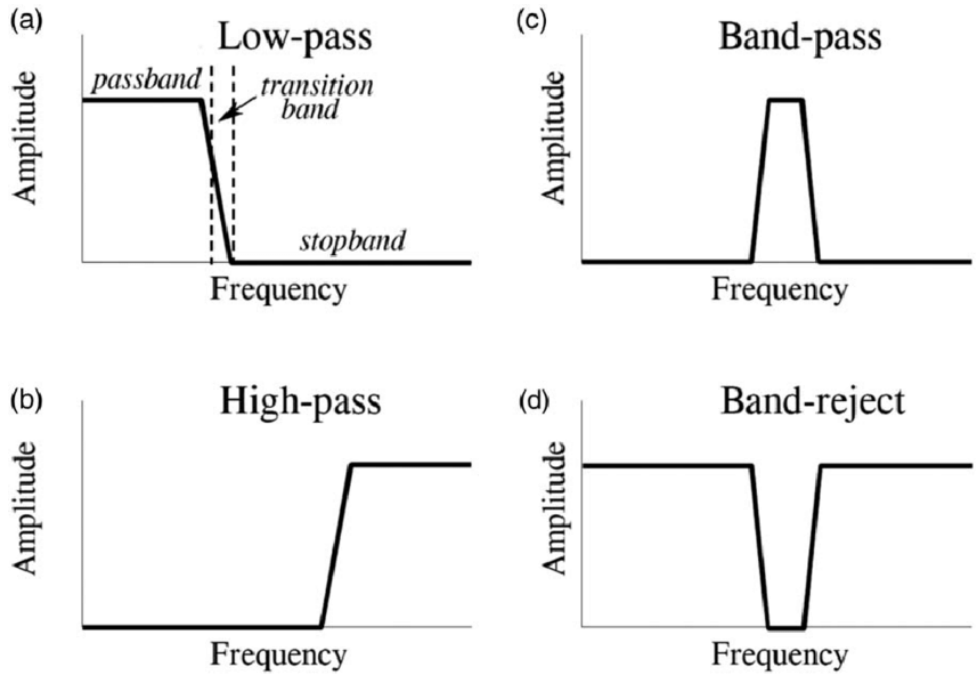


Figure 3.6: Different types of filters [7]

For this thesis Low pass filtering is done to discard the noise from data. To know the cutoff frequency of the signal Fast Fourier transform(FFT) is performed. The peak of FFT can be utilized to know the cutoff frequency. Figure 4.6 shows the highest peak is in between 0 to 1. By testing different values cutoff frequency of 0.005Hz was chosen.

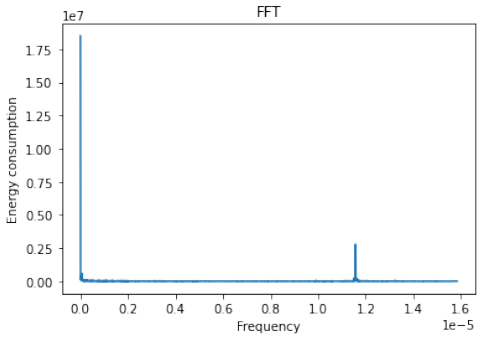
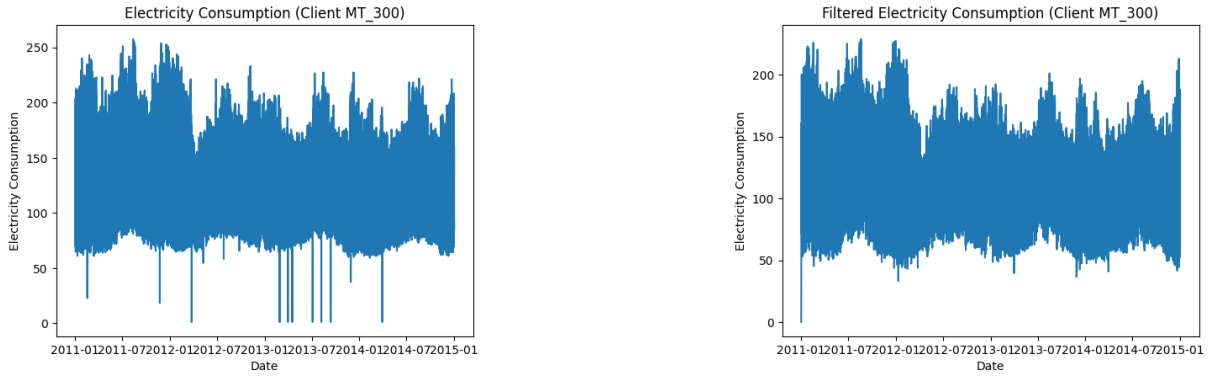


Figure 3.7: FFT of signal

The figure 3.8 shows the rqw signal and signal after filtering. To visualize if the filtering has removed most of the noise for different clients figure 4.8 is plotted for just month of January.



(a) Raw signal

(b) Filtered signal

Figure 3.8: Example filtered signal for MT300 client

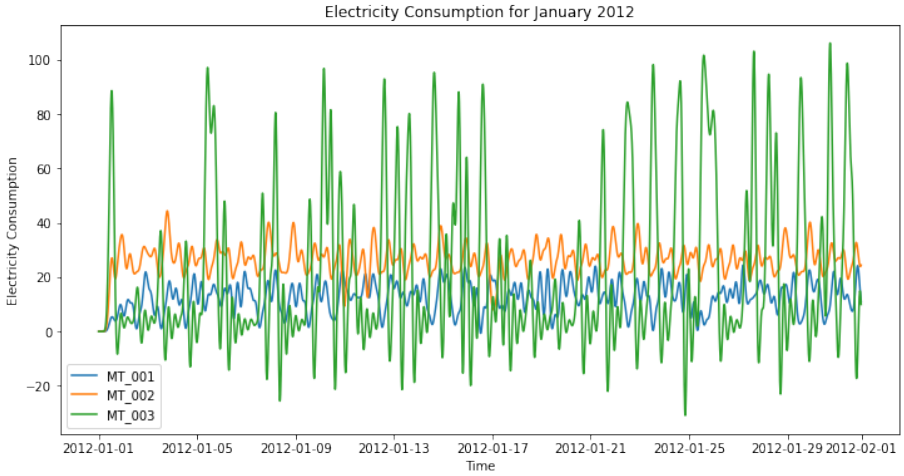


Figure 3.9: January consumption filtered data

3.3.2 Day light saving handling

As stated in above section there is change in time in month of March and October to handle by adjusting the time axis and aggregating the consumption data accordingly. The dates for change of time are 26rd March and 29th October in which there are 23 and 25 hours respectively.

For the day when time changes in March, the time jumps forward by one hour at 2:00AM, which means that the clock skips the values for the interval of 1:59 AM to 3:00 AM. This affects the time-series by creating duplicate hour. For example if the change is between 2:00AM to 3:00AM there will be values for standard time and duplicate values for offset. To handle this the duplicated time is omitted or the timestamps that are affected are adjusted. In this work the time axis is shifted by 1 hour.

For the October time change the time falls back by a hour at 2AM that is the 2:00AM becomes

1:00AM. This also affects the time data as it creates a extra hour in the series. For example if the change is between 1:00AM to 2:00AM there will be values for standard time and duplicate values for offset. To handle this the duplicated time is aggregated for the affected hourand the timestamps are adjusted.

3.3.3 Data Labeling

To test the torchtime library data has to be turned into classification data. As to test how well can the inception time algorithm learn the time-series. In this thesis it is assumed that the weekend has high consumption than the week days. Thus Saturday and Sunday have been marked high and other weekdays are marked low. This assumption is done on the basis that people are generally at home on weekends and thus the consumption might be high for this period. This might not be true for all cases the values that do not follow this pattern are marked as anomaly.

3.3.4 Overview of methods

As stated in the problem statement objective of thesis is to review the use of torchtime library and compare it to other selected methods. This chapter gave a review on the structure of the data and how it was processed in order to obtain a timeseries that can be used as input to these algorithms. Following algorithms will be discussed in the next chapters:

1. Implementation of time-series classification of by using torchtime library
2. LSTM auto-encoder for anomaly classification
3. Unsupervised anomaly detection using Auto-encoders
4. Unsupervised anomaly detection using DeepAnt library

4

Solution

Following chapters discusses two strategies first is supervised anomaly detection and unsupervised anomaly detection. The supervised anomaly detection uses torchtime library, LSTM and Autoencoder for classification of timeseries data. For this the data has been labeled as stated in previous chapter. The unsupervised learning uses Autoencoder and DeepAnt libray for detecting anomalies in the time-series data. The algorithms and results have been discussed in detail in further sections.

4.1 Torchtime library

This section is written in reference to the RND thesis shared by Vincent Scharf for the library. The libray can be found on github at [52]. Torch time library is a tool for time-series analysis and classification. The library mainly consists of four modules which are io,datasets,models and transforms. Each of the functionalities corresponds to the general pipeline followed by any classification algorithm data loading then data preprocessing and training. Figure 4.1 shows the structure of the library.

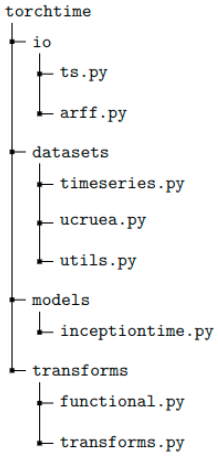


Figure 4.1: Library structure

Following are the functionalities of each module:

1. **IO**: The library is built for a large number of openly available datasets such as UCR and UEA that contain data in different formats. This library module can handle the ts and arff formats. The functionality of these module take care of the parsing and load the data.This module has not been used in this thesis as the data available is in csv format which does not require parsing.
2. **Datasets**: Each of the datasets are different formats has to be loaded in a fashion that has a generalized structure suitable for the algorithm. This module can handle various data structures and convert them to algorithm suitable structure that is converting it to pytorch loader format. In this thesis the Pandas functionality is used.
3. **Models**: Currently this library has only one model that is Inception time. Details on this model have been discussed in next section.
4. **Transforms**: This module corresponds to the data preprocessing step of the pipeline. The module has following functions:
    - Handling NaN values
    - Conversion to tensor
    - Padding of tensors
    - Normalizing the data
    - Encoding the labels in suitable format
    - Re-sampling of data

4.1.1 Inception-time

The network consist of stacked Inception time modules followed by a Global Average Pooling layer and a Dense layer with a softmax activation function. This architecture has many similarities with the GoogleNet net [53]. The difference is that the Inception time has added residual connections after every three inception modules. According to [1] the network has high variance in accuracy. By this the author means that performance of a model is highly variable or unstable. The accuracy can fluctute depending on the data used and can lead to over fitting sometimes. Due to this the model may perform well on the training data but not on unseen data. As a result of the random nature of weight initialization and stochastic optimization, the model's performance can vary widely depending on the specific values chosen. This can lead to instability and inconsistency in the model's performance. To overcome this the author suggests a ensemble of five modules that are randomly initialized with weights and each module is assigned a weight tat assures equal contribution to prediction [54]. This is represented in following equation from [1]:

$$y_{i,c} = \frac{1}{n} \sum_{j=1}^{n} \sigma_c(x_i, \theta_j) \quad \forall c \in [1, C] \tag{4.1}$$

where $y_{i,c}$ is the ensemble output probability that has a input of x and is of a class c calculated by sum of log output $\sigma_c(x_i, \theta_j)$ averaged by n models initialized randomly. The inception time is inspired by

computer vision algorithm therefore it works in similar way. To illustrate a example the low level features such as lines and curves are learned by lower layers, the high level learn complicated structures such as peaks and valleys which is similar to image pattern learning in CNNs. This can be visualized in 4.3.

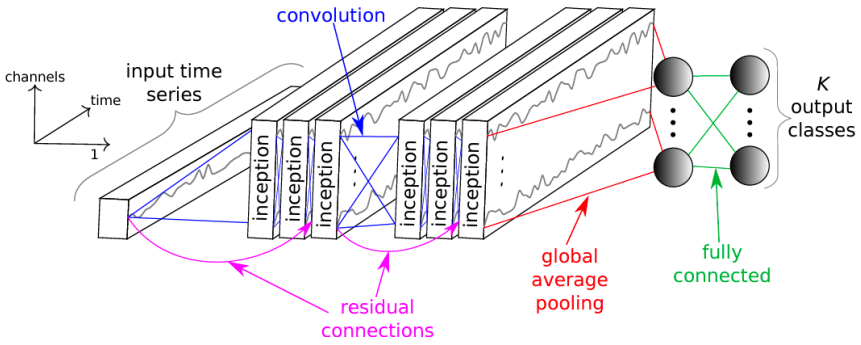


Figure 4.2: Inception time architecture [1]

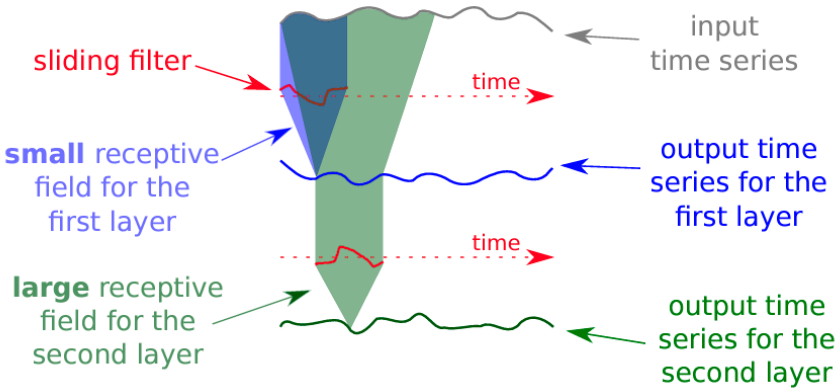


Figure 4.3: Receptive layers of CNN [1]

4.1.2 Inception-time module

This consists of the following layers that can be seen in figure 4.4:

1. It consists of a bottle neck that is incorporated to reduce the dimension of the data . Along with this it lowers the computational cost and generalizes the model.

2. Next are three 1D convolution layers with kernal size of 10, 20 and 40 respectively to which the input is the output of bottle neck.

3. Each of this convolution is passed through Maxpool layer to reduce invariability.

4. The outputs from these three convolution layers or the maxpool are concatenated along the dimensional depth.

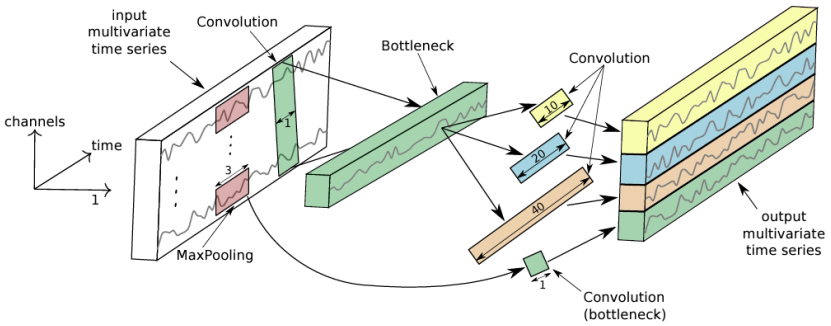


Figure 4.4: Inception time module [1]

4.1.3 Experiment

The inception module parameters that were specified above are in reference to the [1] paper. Following modifications have been made to the parameters of inception time module for this experiment:

- The number of modules used for this experiments is six.
- The convolution layers in each Inception block is set to 3 and each is of size 32 with kernel sizes of 42, 21 and 7.
- The module written for this experiment inherits from the inception module in the torchtime library and adds more functions to it.
- **Configure optimizers**: The use of this method is to setup a optimizer scheduler during the training. The purpose of the learning rate scheduler is to reduce learning rate by some factor when when the validation loss plateaus. For this experiment optimizer used is Adam with learning rate of 0.001.
- **Train, Test, Validation loaders**: The datasets obtained from the pandas data-frame using the Pandas module of torchtime library and are loaded int train, test and validation loaders of pytorch with the specified transforms . The transforms applied here are conversion of the values to tensors, handling of NaN values and Label encoding. This is explained in detain in section 4.1.
- **Step definition for train test and validation**: Each of these function implement steps in which the input is passed through the Inception time model and labels are predicted and loss is calculated accordingly. There can be various metrics used for this training such as precision, accuracy and recall.

To obtain the graphs for the train loss and training accuracy tensor board has to be used which is depicted in following figure 4.5. The log files of training have to uploaded on this to get the graphs.

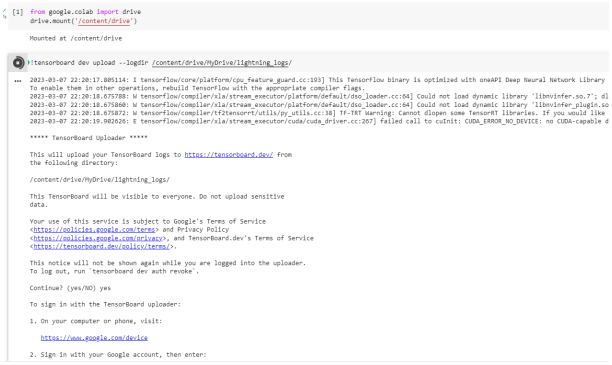


Figure 4.5: Tensor board loading

Following to which the accessed tensor board gui is as follows in 4.6. As it can be seen the scalar values from log files are uploaded onto the tensor board and graphs can be obtained.

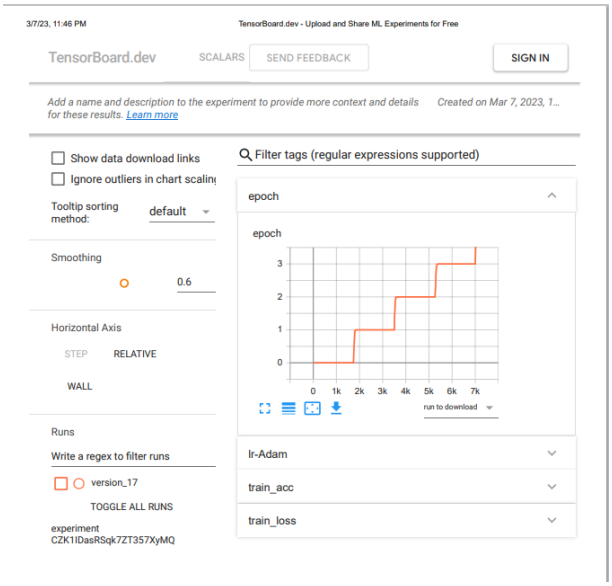
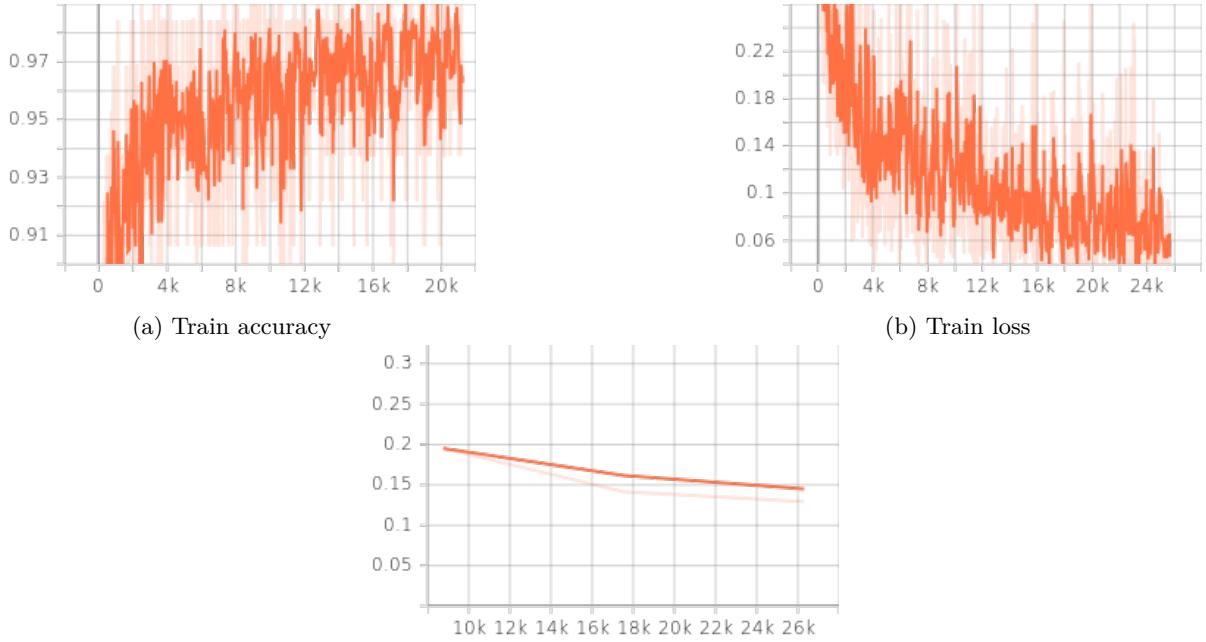


Figure 4.6: Tensor board loading

4.1.4 Results

The training accuracy and train loss have been visualized in 4.7. It can be observed that the training loss is highly fluctuating which means that either the model is not converging or over-fitting the data. From the validation loss plot it can be seen that validation loss is decreasing with increase of train loss.



(a) Train accuracy

(b) Train loss

(c) Validation loss

Figure 4.7: Visualization of training parameters

The accuracy achieved for this experiment is **96%**, this shows that the model is overfitting the data. As stated in [1] the model may have high variance over the accuracy. To solve this, the author had suggested using convolution with different kernel sizes which has been applied in this case also. Then the problem might be with the depth of the Inception model. The number of modules used are 6 and experimentation can be conducted to find the optimal depth that in turn reduces the variance.

4.1.5 Review on the library

According to the implementation of the library following observations were made:

- The library has a clean code structure and easy to understand and debug. The codes are short and variable names are understandable.
- Documentation in still not available since the work has not been published which caused some difficulties during the implementation.
- The results on the implementation of electrical load dataset shows that there is high variance which can be seen in train loss as well as the accuracy on test set. The model needs to be fine tuned.
- The library is built to achieve generalization of pipeline for time-series classification which has been implemented in very clean manner.

- Visualization of torch lightning module has given easier way to plot using tensorboard.
- The usage of tensor board is a complicated process which requires permissions on the local machine to be accessed which makes it difficult. Thus it was done using google colab by mounting the drive and loading the logs.
- Some bugs were solved in the implementation such as problems of input of pandas data-frame in the inception module format. There are some bugs still in using precision and recall as metrics for training which need to be solved. Currently only accuracy has been used.
- Visualization for results has to be added for more clearer view on the functionality.

4.2 LSTM for classification

4.2.1 Long Short-Term Memory (LSTM)

Both Recurrent neural networks and LSTM are commonly used to learn sequential data and especially LSTMs solve the problem of long term dependencies which is essential for learning the time data. This network does not allow the error to backpropogate through the layers by preserving the errors. This network has similar functionality like computers memory where in the information can be read written or deleted. The gates of LSTMs are analog by nature that multiply each elements to a sigmoid function leading to values in rage of 0 to 1. Analog signals have a advantage over digital that is they are differentiable thus this is a advantage during backpropogation.

The structure functionality of the gates is similar to that of a neural network which allows the data to pass or blocks it according to strength and importance filtered by the weights. The weights has a direct effect on to the input and hidden states which are updated during the learning process. This allows the network to learn which data to save, delete or discard. This follows a iterative process by using gradient decent to backpropogate. Figure 4.8 shows the process flow of a memory cell.

In figure 4.8 the input of information at flow into the cell marked by three arrows. The information of past and present is fed back to the cells as well as the gates of the cell. As stated above the gates decide which data to keep or discard, the gates in the figure are marked by black dots. In the equation written in the block S_c is the current state of memory and g_y is the input. At each iteration the gates have combination of begin either open or closed for a particular data input. The flow of the data at each time steps is represented by this combination of state of the gates.

Figure 4.10 can be used to compare LSTM to RNN. The main difference between LSTM and RNN is in how the transformation of input is done. Looking at the figure the plus sign in the between of both the cells at left and right marks the difference which has the influence of s preservation of error in LSTMs. In LSTM block the instead of multiplication of new input to current state to obtain the next state addition is done. Weights act as filter for the state of the gates. The gate that discards information is represented using identity function thus the current state is multiplied by 1 in forward step. The forget gate or the gate that discards value improves the performance of the network as it is useful to discard the information that is no longer required.

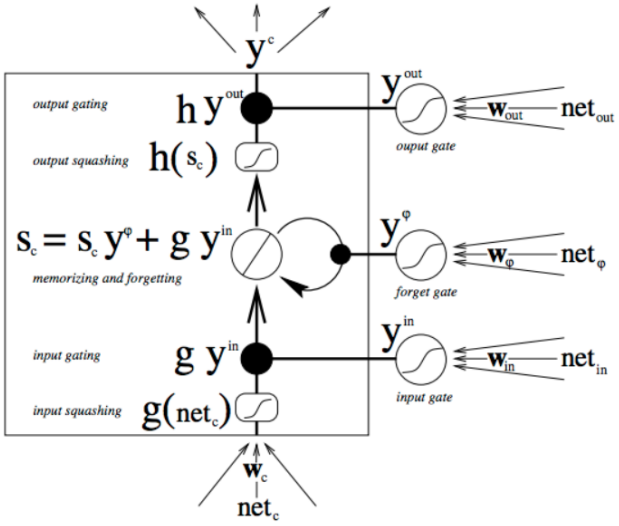


Figure 4.8: Process flow of memory cell [8]

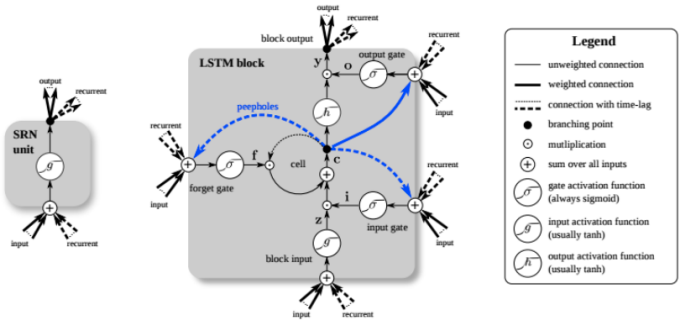


Figure 4.9: Simple RNN and LSTM blocks [8]

Figure 4.10 shows a representation of the work flow of the gate during the learning process. A it can be seen the inputs can me mapped on to one or more than one outputs.

4.2.2 Experiment

Following are the details of the network used by the the experiment of this implementation:

- The input to network is preprocessed time-series which have been split into train and test by dividing data into 70:30 ratio for each of the consumption of each client.
- The labels are marked based on assumption discussed in the previous chapter.
- The architecture for the network written for this experiment is shown as shown in figure 4.11. It has two LSTM layers of size 128 and 64 followed by a dense layer with sigmoid activation function

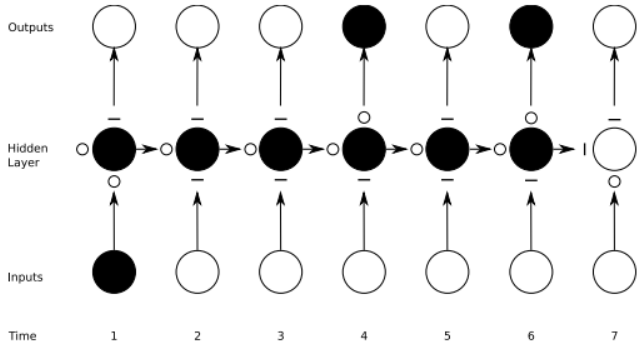


Figure 4.10: Function of the LSTM gates [8]

for binary classification.

- Loss plot is obtained for analyzing the training of the data and evaluation is done using accuracy, ROC curve and F1 score as the metrics.

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_5 (LSTM)               (None, 370, 128)          66560

 lstm_6 (LSTM)               (None, 370, 64)           49408

 dense_2 (Dense)             (None, 370, 1)            65

=================================================================
Total params: 116,033
Trainable params: 116,033
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.11: Architecture for LSTM model for time-series classification

4.2.3 Results

From figure 4.12 it can be observed that there is slight increase in the validation loss with decrease in training loss which indicates over fitting.

The accuracy achieved was **90%** this means that the algorithm is able to classify at-least the low consumptions correctly but it marks high consumption low as well. To clarify this further ROC plotted.

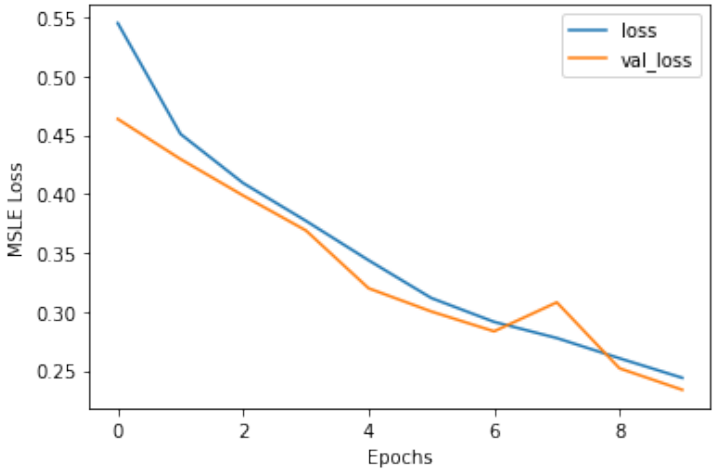


Figure 4.12: Train loss

It indicates that AUC score is 0.5. To make it clear why the ROC value is **0.5** F1 score is calculated to be **0.23**.

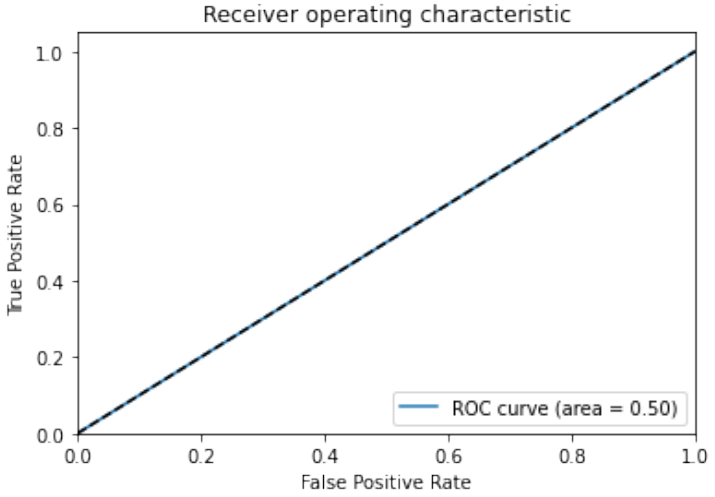


Figure 4.13: ROC curve for the implementation

4.2.4 Discussion

As it is observed by the result, the test accuracy is **90%** which means that the test set were classified correctly by the model. But the AUC score is 0.5 indicating the models ability to differentiate between positive and negative samples is not good. Also it shows that the model is just randomly classifying the data. Another reason might be the model is biased towards one of the classes than other. Possible reasons

for such a result are as follows:

- The value of prediction probabilities is very near to 0.5 which the threshold set for distinguishing the anomalies.
- Assumption of the labels might not be correct.

4.3 Autoencoder for classification

4.3.1 Autoencoder

Autoencoder is a unsupervised deep-learning algorithm which learns data by encoding it. Objective of this algorithm is to learn lower dimension of data by encoding it instead of complexed high dimension data such as images. Autoencoders consists of three parts discussed as follows:

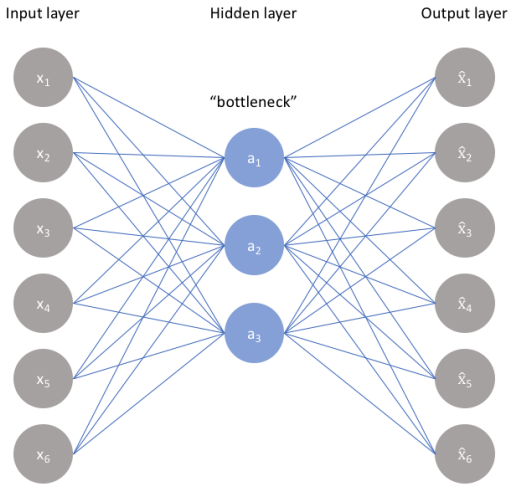


Figure 4.14: Autoencoder with LSTM layers [9]

1. **Encoder**: In this module the input data is converted to encoded representation by reducing the dimensionality of data. In terms of neural networks the encoding is accomplished by reducing nodes in subsequent layers. The process reduces the information and only the most relevant or average information is forwarded.
2. **Latent space**: This space contains the reduced or compressed knowledge of the data. The representation obtained can be used as feature input to classification algorithms.
3. **Decoder**: This module reconstructs the encoded information of the data to it original, that is it decompresses the data.

Figure 4.15 shows the flow of sequencial data through a Autoencoder built using LSTM layers. It depicts the example how the data of one time step is passed from one layer to another for each time step. This example uses a multivariate time series with 2 features as input. The output is in form of vector of size 3×2 which is the reproduced multivariate for the original data. As it can be noticed it also is consists a bottle neck layer.

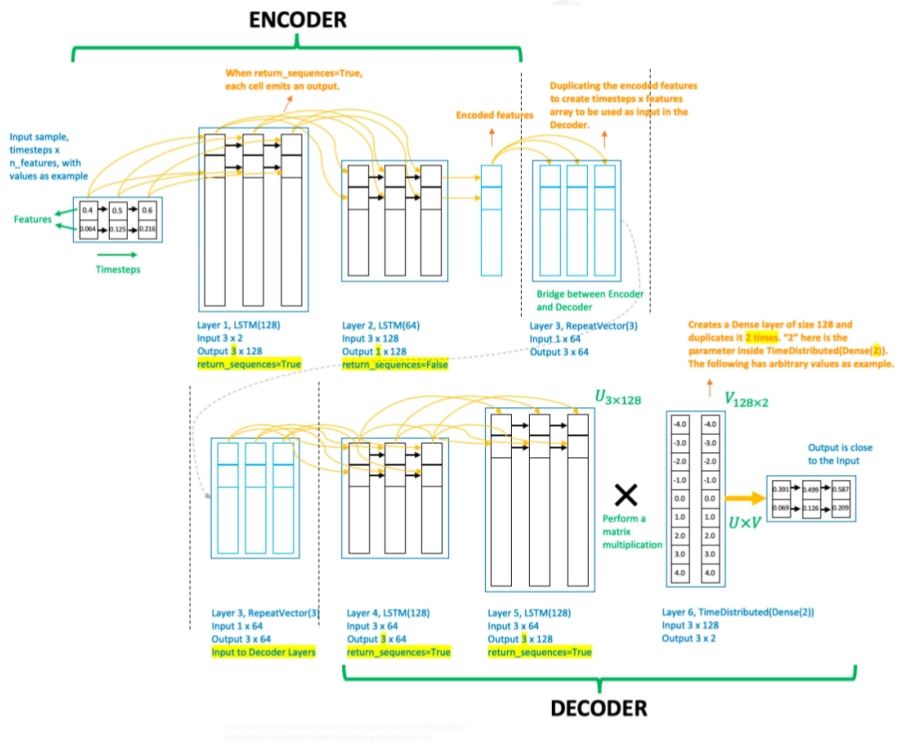


Figure 4.15: Autoencoder with LSTM layers [10]

4.3.2 Experiment

Following steps are followed for this experiment:

- The input to this experiment is labeled data from the preprocessing step. The obtained dataset is scaled using Min-max Scaler and split into train and test set with 70 to 30 ratio.
- The network is autoencoder with LSTM layers that has been explained in previous section. The encoder and decoder consist of two layers of size 64 and 32 each with a bottleneck of 16 size.
- The model here is trained in unsupervised manner using mean squared logarithmic loss and mean squared error as metric.
- The network is trained for 50 epochs with Adam optimizer and with learning rate of 0.001.

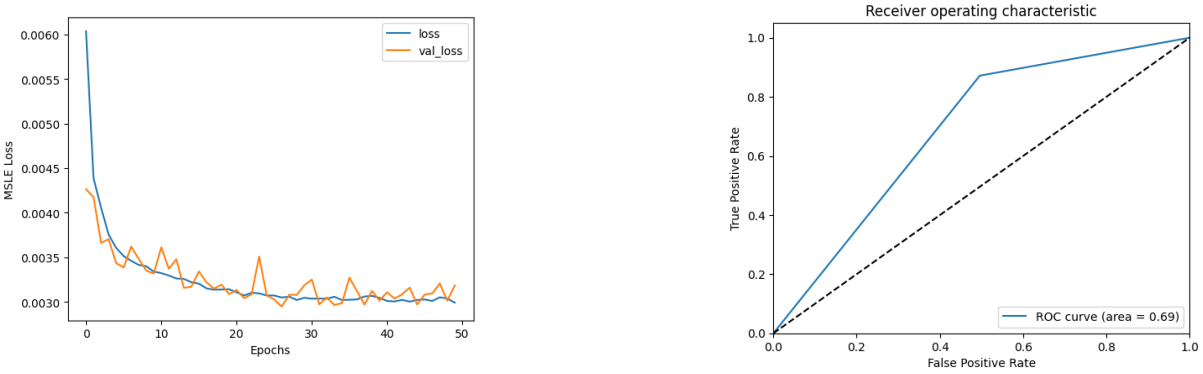
- To analyze the performance of model, training loss is plotted.
- Reconstruction error was used to define the threshold for classification of data.
- There are two ways defined to calculate the threshold. First is by taking the mean of reconstruction errors and second is to select a reconstruction error that falls in the 95th percentile of the data.
- The method is evaluated on accuracy, F1 score, ROC curve and Confusion matrix.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder_1 (LSTM)            (None, 4, 64)             111616

 encoder_2 (LSTM)            (None, 4, 32)             12416

 encoder_3 (LSTM)            (None, 16)                3136

 encoder_decoder_bridge (Rep  (None, 4, 16)            0
 eatVector)

 decoder_1 (LSTM)            (None, 4, 16)             2112

 decoder_2 (LSTM)            (None, 4, 32)             6272

 decoder_3 (LSTM)            (None, 4, 64)             24832

 time_distributed (TimeDistr  (None, 4, 371)           24115
 ibuted)

=================================================================
Total params: 184,499
Trainable params: 184,499
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.16: Architecture for LSTM- Autoencoder model for time-series classification

4.3.3 Results

The accuracy achieved was **74%** but accuracy does not explain the full picture of how well the model is performing. Thus to clarify this further ROC is plotted and it indicates that AUC score is 0.69 which is better than the previous model that had over fitting issues. F1 score is calculated to be **0.83**. The loss plot has some fluctuations in the validation loss. The confusion matrix shows that there are 6456 false positive values and 3728 false negative values. The original test set contains 29060 values for high consumption and 13017 for low consumption values.



(a) Loss plot

(b) ROC curve

Figure 4.17: Visualization of training parameters

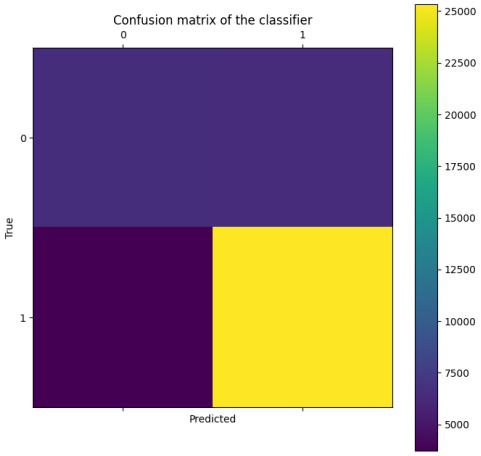


Figure 4.18: Confusion matrix

4.3.4 Discussion

As stated above accuracy does not clarify the performance of the model. The F1 score indicates that the model is performing well. Generally F1 score lies between 0 to 1 and is harmonic mean of precision and recall. The AUC score also lies between 0 to 1. For this experiment, AUC score is 0.69 which is more than 0.5 thus the model is performing well than randomly guessing the values. Over all, the performance of the model is good. Looking at the confusion matrix it seems that there are many number of values that have been misclassified. The reason behind this might be assumption made for the labeling of data. There is possibility that there are some anomalies in the labeled data that means days which have been marked as high consumption might be low consumption days or vise versa. As there is not much information on to the environment changes or the changes that took place while collection of data nothing much can be implied from this.

4.4 Autoencoder for anomaly detection

Figure 4.19 depicts simple algorithm for anomaly detection of timeseries data.

Algorithm 2: Residual error-based anomaly detection algorithm.

Input: Sets of data to classify $x_1, x_2 \ldots x_n$
Output: Residual error $L_{(x,\hat{x})}$
$\phi, \theta \leftarrow$ train network parameters with Algorithm 1
$\alpha \leftarrow$ set threshold based on training set
repeat
for i=1 to N **do**
Compute residual error $L_{(x,\hat{x})}$
$L_{(\phi,\theta;x_i)} = \sum_i || x_i - g_\theta (f_\phi (x_i)) ||^2$
if residual error $L_{(x,\hat{x})} > \alpha$ **then**
x_i is abnormal
else
x_i is normal
end if
end for

Figure 4.19: Algorithm for anomaly detection [**?**]

4.4.1 Experiment

Following are the details of the model for this experiment:

- One of the customers is randomly chosen to detect anomalies in the electrical consumption data.
- There are equal number of inputs and outputs in the network. The purpose of this network is to reduce the reconstruction error between the decoded and original signals.
- The design of the model is specific to sequence data. Input to this model is sequence of 30 with one feature at each step .
- The following layer is LSTM with 64 units and dropout layer of rate 0.2. The dropout layer helps in reducing over-fitting during the training.
- The model uses Adam optimizer and the loss is mean absolute error loss.

```
Model: "lstm_ae_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 30, 1)]           0

 lstm (LSTM)                 (None, 64)                16896

 dropout (Dropout)           (None, 64)                0

 repeat_vector (RepeatVector  (None, 30, 64)           0
 )

 lstm_1 (LSTM)               (None, 30, 64)            33024

 dropout_1 (Dropout)         (None, 30, 64)            0

 time_distributed (TimeDistr  (None, 30, 1)            65
 ibuted)

=================================================================
Total params: 49,985
Trainable params: 49,985
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.20: Architecture for LSTM- Autoencoder model for time-series anomaly detection

- Train loss is plotted to analyze training process of the model.
- Reconstruction error threshold to detect anomalies is calculated using the mean absolute error between the original signal and reconstructed signal from the encoding of the model.
- This method assumes that the mean absolute error is unimodal and anomalies are those outliers which have a very high mean absolute error.
- Then the anomalies are detected by comparing the threshold to error of each data points and plotted on to the original signal to visualize them.

4.4.2 Results

Following observations are made for this algorithm:

- Figure 4.21 is a plot for train loss and validation loss. There is not much fluctuation observed in the graph which means the model is able to generalize over the test dataset.
- In the threshold plot shown in figure 4.22a, it is observed that there is a clear distinction onto the values that have very high values which are near 0.7. Therefore, the threshold chosen was 0.7.
- In the figure 4.22b, the threshold is plotted on the test set of time series. The threshold line marks a clear distinction of the anomalies but to clarify it further, anomaly data points are plotted in 4.22

4.4.3 Discussion

The algorithm detects both positive and negative anomalies as seen in the figure 4.22. The electrical data consumption is affected by various factors therefore it aids to analyze the occurrence of the anomalies in this case. As it can be seen in the 4.24, anomalies detected are mostly in July around 5:00pm. It can

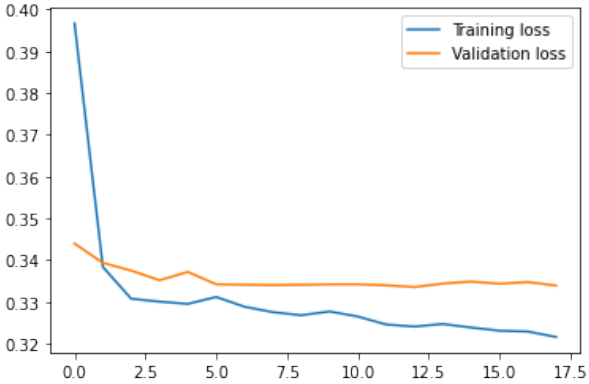
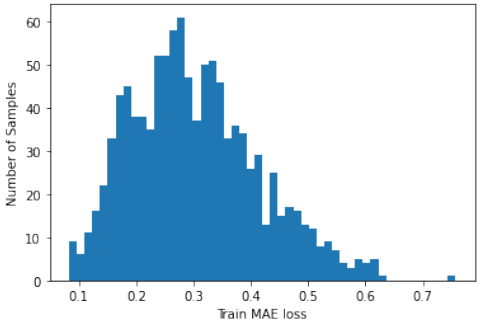
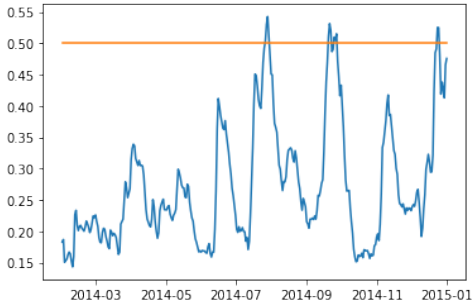


Figure 4.21: Loss plot



(a) Bar plot to determine threshold



(b) Threshold visualization on the timeseries

Figure 4.22: Thresholds

be said July lies in the peak of summer and around the time mentioned people return to their houses from work which is when sudden change in consumption is observed. It can be assumed sudden usage of appliances during this time may have caused changes leading to change in normal pattern. Another set of anomalies belong to September and December, around same time similar deduction as before can be made here too. The only difference is that it is Winter.

4.5 DeepAnt library

This section describes the algorithm defined by [55] and the modification for implementation to fit Electrical load dataset.

4.5.1 Algorithm

The model of this library consists of two modules Time-series predictor for forecasting the time data and Anomaly detector to mark the anomalies in the data. Deep-learning models especially the CNNs are very good at extracting features from the images as well as sequential data. This property of the network

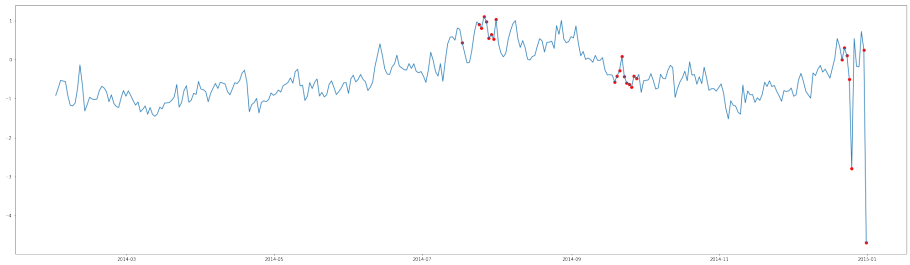


Figure 4.23: Anomalies

Date	value	loss	threshold	anomaly
2014-07-26 17:00:00	1.105710	0.510867	0.5	True
2014-07-27 17:00:00	0.978872	0.536777	0.5	True
2014-07-28 17:00:00	0.554805	0.549039	0.5	True
2014-07-29 17:00:00	0.643964	0.515748	0.5	True
2014-09-19 17:00:00	-0.417727	0.522997	0.5	True
2014-09-20 17:00:00	-0.280843	0.542396	0.5	True
2014-09-21 17:00:00	0.078656	0.529256	0.5	True
2014-09-24 17:00:00	-0.628276	0.519507	0.5	True
2014-09-25 17:00:00	-0.698003	0.517656	0.5	True
2014-09-26 17:00:00	-0.424184	0.527385	0.5	True
2014-12-23 17:00:00	0.105347	0.517322	0.5	True
2014-12-24 17:00:00	-0.503360	0.518405	0.5	True

Figure 4.24: Table of detected Anomalies

can be useful in detecting anomalies in the data. The architecture of DeepAnt is shown in 4.25

TIME SERIES PREDICTOR

Timeseries predictor module has is a Convolution neural network. The network consists of three layers convolution, max-pool and fully connected layer. The library uses Stochastic Gradient Descent (SGD) to optimize its parameters. The forecasting of data using CNN requires some modifications to the time-series for it to be suitable input to the convolution layer. Consider a time-series with an element x_t and the next element to be x_{t+1}. A sequence of overlapping windows is created with second time-step to be label of the previous element. The forecasting method predicts prediction window which is defined by the number of time steps to be predicted. In this implementation the left data is the input and right acts as a label. This module implements many to one prediction.

ANOMALY DETECTOR

Input to this module is the predicted time step from the time series predictor. The anomaly detector is a threshold based detector algorithm that uses Euclidean distance as threshold to detect if the predicted

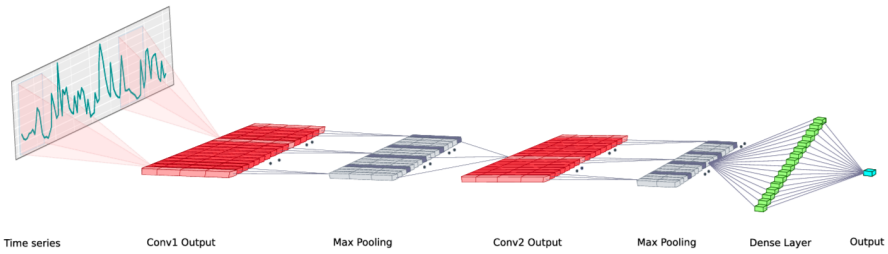


Figure 4.25: Table of detected Anomalies

time step is a anomaly.

$$d = \sqrt{(y_t)^2 + (y'_t)^2} \tag{4.2}$$

where (y_t) and (y'_t) are real time-step and predicted time-step. d denote the Euclidean distance between the two.

4.5.2 Experiment

Following are the details of this implementation:

- To make the data suitable for being used as input to this library is converted into sequences of size 10 .
- The algorithm uses mean squared error as the loss function with learning rate of 0.001 with Stochastic Gradient Descent optimizer.
- The model has been built using pytorch lightning module.
- Threshold is plotted using the visualization module of the library.
- To visualize the anomalies further anomalous points are plotted on to the test data.

4.5.3 Results

Following observations are made for the implementation:

- Figure 4.27a, shows the distribution of loss. This helps in defining the threshold value for anomaly detection. Here it can be observed it lies somewhere in between of 0.3 and 0.4. Therefore 0.34 was chosen.
- Figure 4.27b marks the threshold line onto the test set which helps to distinguish the normal from the abnormal.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder_1 (LSTM)            (None, 4, 64)             111616

 encoder_2 (LSTM)            (None, 4, 32)             12416

 encoder_3 (LSTM)            (None, 16)                3136

 encoder_decoder_bridge (Rep  (None, 4, 16)            0
 eatVector)

 decoder_1 (LSTM)            (None, 4, 16)             2112

 decoder_2 (LSTM)            (None, 4, 32)             6272

 decoder_3 (LSTM)            (None, 4, 64)             24832

 time_distributed (TimeDistr  (None, 4, 371)           24115
 ibuted)

=================================================================
Total params: 184,499
Trainable params: 184,499
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.26: Architecture for DeepAnt algorithm

(a) Bar plot to determine threshold

(b) Threshold visualization on the timeseries

Figure 4.27: Thresholds

- The algorithm is able to detect positive as well as negative anomaly values. Positive anomalies represent unexpected high consumption and the negative indicate unexpected low consumption.

4.5.4 Discussion

Similar pattern of anomalies can be visualized in figure 4.29 as in the algorithm of previous section. The anomalies that have been detected are in the peaks of the Summer and Winter season that is July ,August, January and December around 5:00pm. The plausible reason for this might be high usage of appliances after returning home. As there is not much information on to how the parameters of the environment for this client change, deductions are just based on the assumption.

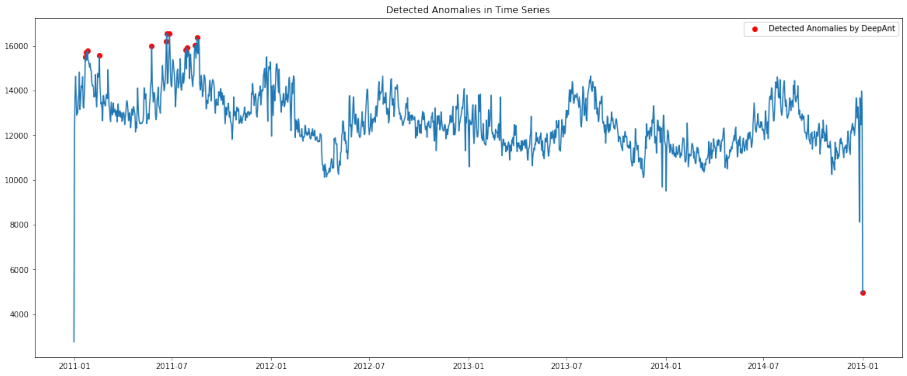


Figure 4.28: Anomalies

```
Anomalies Detected:
2011-01-21 17:00:00    0.350862
2011-01-23 17:00:00    0.365409
2011-01-25 17:00:00    0.356676
2011-02-16 17:00:00    0.353449
2011-05-24 17:00:00    0.422989
2011-06-20 17:00:00    0.385604
2011-06-21 17:00:00    0.409536
2011-06-26 17:00:00    0.367551
2011-07-26 17:00:00    0.358273
2011-07-28 17:00:00    0.357440
2011-08-12 17:00:00    0.354636
2011-08-13 17:00:00    0.364151
2011-08-16 17:00:00    0.367580
2014-12-31 17:00:00    0.412911
```

Figure 4.29: Table of detected Anomalies

4.5.5 Review on the library

Following observations have been made for the libray through the implementation:

- The library has very simple implementation and provides a pipeline for anomaly detection of timeseries data.
- Documentation of the library is in detail with example implementation which makes it very easy to understand the pipeline of implementation.
- The library also has implementation to detect anomalous sequence in timeseries which was not tested in this work.
- The libray also provides visualization module which makes it easier to analyze the results.

5

Conclusions

Electricity is integral part of both industrial and household basic necessities, to a extend that our daily routine gets hindered in absence of the supply. To ensure continuous power supply and detect the faults in the supply systems, monitoring the system is essential. Monitoring is generally done by the sensors which generate huge data that needs to be analyzed. With introduction of deep-learning algorithms detecting faults in the data has become very easy as it does not require manual labeling of data. This thesis aims to study electrical consumption data collected for small group of customers in one of the regions in Portugal and detect faults in consumption signal.

The data does not have any undefined or missing values instead for the customers which the data in not available is marked zero. The data was filtered for noise and the dates where there are changes due to Daylight saving were handled. A deep analysis on data was done to understand the behavior of the data and select appropriate methods to be implemented to it for detecting anomalies. Through this analysis it was found that the data consumption is more during night weekends especially Fridays. According to the seasonal change, the consumption in summers is more than winters. There is difference in trend of each clients electrical consumption which is obvious as the different sources have different patterns of electrical consumption.

First method was implemented to test the suitability of torch time library for electrical load data. The library is build for classification datasets but the electrical load dataset does not have any labels thus it was labeled based on the assumption that weekends are high consumption days. The model gives 96 percent accuracy on test set which means that it overfits the data. The possible reasons for this might be incorrect labeling of data and the high invariance of the data. Overall the library was easy to implement and some of the bugs were fixed during the experimentation of the inception module. The code is very well written, it is clear and easy to understand.

To know if the data labeling is correct or not LSTM model that are commonly used for learning, sequential data learning was implemented. This network is shallow that is does not have many layers to avoid overfitting problem. The model achieves 90 percent accuracy but through the ROC plot and F1 score it was observed that the model does not perform very well on to the dataset and overfits the data. Possible reasons might be incorrect labeling of data.

To investigate further an unsupervised approach is used for classification of anomalies using LSTM autoencoder. The model achieves 74 percent accuracy on test set but looking at the F1 score and the

ROC curve, the model performs well and does not randomly classify the data which was seen in the methods mentioned above.

Next step was to build algorithm that detects anomalies in individual electrical consumption. This approach was decided onto the findings of the data that each of the customers have different trends of electrical consumption so the anomalies in each of them might differ and it cannot be blindly assumed that each customer has high consumption only on weekends. Thus a shallow LSTM autoencoder in comparison to the above model was used. It was noticed that the anomalies are around 5:00pm in peak of Winters and Summers. The reason behind this might be sudden usage of electrical appliances due to extreme weather conditions on those days. The reason behind the spikes at exactly 5 might be due to the fact that it is time around which a person returns home.

Similar results were obtained using DeepAnt library. This library also detects anomalies at 5:00pm in extreme weather conditions. So similar deductions can be made for this too. The fact that both the library and the implemented algorithm have similar anomalies, shows that the algorithms are performing well on the data. This library was easy to implement and is well documented.

Overall the supervised methods do not perform very well due to the label assumption in comparison to unsupervised methods.

5.1 Contributions

The contribution of this work are as follows:

- Survey of the supervised and unsupervised state of the classification and anomaly detection algorithms for time-series.
- In depth data analysis of Electrical Load dataset.
- Data transformation and labeling to be suitable for input to algorithms.
- Implementation of time-series classification by usage of torch time library and library review based on the suitable evaluation metrics.
- Implementation of LSTMs for time-series classification.
- Implementation of unsupervised classification using exsisting LSTM Autoencoder network.
- Implementation of LSTM autoencoders for anomaly detection in timeseries.
- Implementation of time-series anomaly detection by usage of DeepAnt library and library review based on the suitable evaluation metrics.
- Comparative evaluation of the anomaly detection and classification techniques based on suitable metrics.

5.2 Lessons learned

- Unsupervised anomaly detection methods perform better on the time series data in comparison of supervised methods.
- The labeling of data collected for any real time variable depends on various factors and is not easy to assume and requires in depth knowledge of the domain.
- To analyze the anomaly detection more information on the environment of the data collected is required.
- There is still scope of improvement in the implementation of torchtime libray by fine-tuning the model.

5.3 Future work

The future scope of this work is as follows:

- Experimentation to fine tune the inception time model of torchtime library.
- The DeepAnt library implements detection of anomalous sequence in addition to the point anomaly detection applied in this thesis. This method can be explored to get insight onto the time period of anomaly as the consumption can be high for particular period of time.
- Comparison of the implementation of the anomaly detection algorithms for different customers to get a better view on the problem.

A

Design Details

Following is the structure of the implementation:

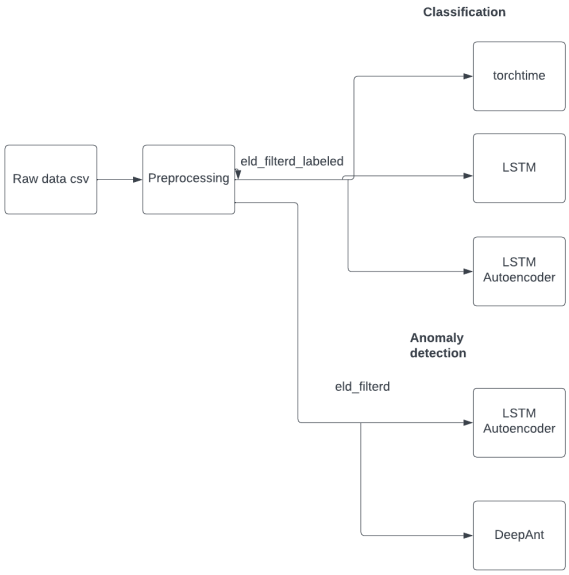


Figure A.1: An overview of implementation

References

[1] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.

[2] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2003, pp. 2–11.

[3] "Plotting a spectrogram using python and matplotlib," https://pythontic.com/visualization/signals/spectrogram, accessed: 2022-02-02.

[4] T. Chen, X. Liu, B. Xia, W. Wang, and Y. Lai, "Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder," *IEEE Access*, vol. PP, pp. 1–1, 03 2020.

[5] C. Cassisi, P. Montalto, M. Aliotta, A. Cannata, and A. Pulvirenti, *Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining*, 09 2012.

[6] H. F. Nweke, Y. W. Teh, M. A. Al-Garadi, and U. R. Alo, "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges," *Expert Systems with Applications*, vol. 105, pp. 233–261, 2018.

[7] H. Obeid, H. Khettab, L. Marais, M. Hallab, S. Laurent, and P. Boutouyrie, "Evaluation of arterial stiffness by finger–toe pulse wave velocity: optimization of signal processing and clinical validation," *Journal of hypertension*, vol. 35, no. 8, pp. 1618–1625, 2017.

[8] "A beginner's guide to lstms and recurrent neural networks," https://wiki.pathmind.com/lstm.

[9] "Autoencoders in deep learning: Tutorial and use cases," https://www.v7labs.com/blog/autoencoders-guide.

[10] "Step-by-step understanding lstm autoencoder layers," ttps://towardsdatascience.com/step-by-step-understanding-lstm-autoencoder-layers-ffab055b6352.

[11] K. Berk, *Modeling and forecasting electricity demand: a risk management perspective*. Springer, 2015.

[12] G. Tang, K. Wu, J. Lei, Z. Bi, and J. Tang, "From landscape to portrait: a new approach for outlier detection in load curve data," *IEEE Transactions on Smart Grid*, vol. 5, no. 4, pp. 1764–1773, 2014.

[13] M. Murugappan and S. Murugappan, "Human emotion recognition through short time electroencephalogram (eeg) signals using fast fourier transform (fft)," in *2013 IEEE 9th International Colloquium on Signal Processing and its Applications*. IEEE, 2013, pp. 289–294.

[14] N. Saravanan and K. Ramachandran, "Incipient gear box fault diagnosis using discrete wavelet transform (dwt) for feature extraction and classification using artificial neural network (ann)," *Expert systems with applications*, vol. 37, no. 6, pp. 4168–4181, 2010.

[15] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.

[16] A. Supratak, L. Li, and Y. Guo, "Feature extraction with stacked autoencoders for epileptic seizure detection," in *2014 36th Annual international conference of the IEEE engineering in medicine and biology society*. IEEE, 2014, pp. 4184–4187.

[17] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: a comprehensive evaluation," *Proceedings of the VLDB Endowment*, vol. 15, no. 9, pp. 1779–1797, 2022.

[18] E. Keogh, J. Lin, S.-H. Lee, and H. V. Herle, "Finding the most unusual time series subsequence: algorithms and applications," *Knowledge and Information Systems*, vol. 11, pp. 1–27, 2007.

[19] X. Wang and C. Wang, "Time series data cleaning with re lar and irregular time intervals," *arXiv preprint arXiv:2004.08284*, 2020.

[20] T. M. Alarcon Falconi, B. Estrella, F. Sempértegui, and E. N. Naumova, "Effects of data aggregation on time series analysis of seasonal infections," *International journal of environmental research and public health*, vol. 17, no. 16, p. 5887, 2020.

[21] A. Marcet, "Temporal aggregation of economic time series," in *Rational Expectations Econometrics*. CRC Press, 2019, pp. 237–282.

[22] J. R. Freeman, "Systematic sampling, temporal aggregation, and the study of political relationships," *Political analysis*, vol. 1, pp. 61–98, 1989.

[23] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Locally adaptive dimensionality reduction for indexing large time series databases," in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001, pp. 151–162.

[24] P. Chaudhari, D. P. Rana, R. G. Mehta, N. J. Mistry, and M. M. Raghuwanshi, "Discretization of temporal data: a survey," *arXiv preprint arXiv:1402.4283*, 2014.

[25] E. Keogh, S. Lonardi, and B.-c. Chiu, "Finding surprising patterns in a time series database in linear time and space," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 550–556.

[26] W. I. D. Mining, "Data mining: Concepts and techniques," *Morgan Kaufinann*, vol. 10, pp. 559–569, 2006.

[27] R. Dash, R. L. Paramguru, and R. Dash, "Comparative analysis of supervised and unsupervised discretization techniques," *International Journal of Advances in Science and Technology*, vol. 2, no. 3, pp. 29–37, 2011.

[28] A. M. Ahmed, A. A. Bakar, and A. R. Hamdan, "Dynamic data discretization technique based on frequency and k-nearest neighbour algorithm," in *2009 2nd Conference on Data Mining and Optimization*. IEEE, 2009, pp. 10–14.

[29] T.-c. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011.

[30] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and knowledge discovery*, vol. 15, pp. 107–144, 2007.

[31] J. MacQueen, "Classification and analysis of multivariate observations," in *5th Berkeley Symp. Math. Statist. Probability*. University of California Los Angeles LA USA, 1967, pp. 281–297.

[32] A. Gupta, K. G. Mehrotra, and C. Mohan, "A clustering-based discretization for supervised learning," *Statistics & probability letters*, vol. 80, no. 9-10, pp. 816–824, 2010.

[33] X.-y. Chen and Y.-y. Zhan, "Multi-scale anomaly detection algorithm based on infrequent pattern of time series," *Journal of Computational and Applied Mathematics*, vol. 214, no. 1, pp. 227–237, 2008.

[34] J. Zhang, F.-C. Tsui, M. M. Wagner, and W. R. Hogan, "Detection of outbreaks from time series data using wavelet transform," in *AMIA Annual Symposium Proceedings*, vol. 2003. American Medical Informatics Association, 2003, p. 748.

[35] C. Shahabi, X. Tian, and W. Zhao, "Tsa-tree: A wavelet-based approach to improve the efficiency of multi-level surprise and trend queries on time-series data," in *Proceedings. 12th International Conference on Scientific and Statistica Database Management*. IEEE, 2000, pp. 55–68.

[36] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 3. IEEE, 2003, pp. 1741–1745.

[37] V. Chandola, D. Cheboli, and V. Kumar, "Detecting anomalies in a time series database," 2009.

[38] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multi-dimensional time-series with support for multiple distance measures," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 216–225.

[39] P. Protopapas, J. Giammarco, L. Faccioli, M. Struble, R. Dave, and C. Alcock, "Finding outlier light curves in catalogues of periodic variable stars," *Monthly Notices of the Royal Astronomical Society*, vol. 369, no. 2, pp. 677–696, 2006.

[40] C. Chatfield, *The analysis of time series: an introduction*. Chapman and hall/CRC, 2003.

[41] R. Fujimaki, T. Yairi, and K. Machida, "An anomaly detection method for spacecraft using relevance vector learning," in *Advances in Knowledge Discovery and Data Mining: 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005. Proceedings 9*. Springer, 2005, pp. 785–790.

[42] B. Pincombe, "Anomaly detection in time series of graphs using arma processes," *Asor Bulletin*, vol. 24, no. 4, p. 2, 2005.

[43] H. Z. Moayedi and M. Masnadi-Shirazi, "Arima model for network traffic prediction and anomaly detection," in *2008 international symposium on information technology*, vol. 4. IEEE, 2008, pp. 1–6.

[44] F. Knorn and D. J. Leith, "Adaptive kalman filtering for anomaly detection in software appliances," in *IEEE INFOCOM Workshops 2008*. IEEE, 2008, pp. 1–6.

[45] Y. Bengio, Y. LeCun *et al.*, "Scaling learning algorithms towards ai," *Large-scale kernel machines*, vol. 34, no. 5, pp. 1–41, 2007.

[46] V. Kuznetsov, V. Moskalenko, and N. Y. Zolotykh, "Electrocardiogram generation and feature extraction using a variational autoencoder," *arXiv preprint arXiv:2002.00254*, 2020.

[47] S. Mehtab and J. Sen, "Analysis and forecasting of financial time series using cnn and lstm-based deep learning models," in *Advances in Distributed Computing and Machine Learning*. Springer, 2022, pp. 405–423.

[48] A. Sagheer and M. Kotb, "Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems," *Scientific reports*, vol. 9, no. 1, pp. 1–16, 2019.

[49] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[50] J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[51] "Electricityloaddiagrams20112014 data set," http://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014.

[52] "Torchtime," https://github.com/VincentSch4rf/torchtime.

[53] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[54] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep neural network ensembles for time series classification," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–6.

[55] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "Deepant: A deep learning approach for unsupervised anomaly detection in time series," *Ieee Access*, vol. 7, pp. 1991–2005, 2018.