REAL-TIME TIME SERIES ERROR-BASED DATA REDUCTION FOR INTERNET-OF-THINGS APPLICATIONS

WONG SIAW LING

MASTER OF SCIENCE (COMPUTER SCIENCE)

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TUNKU ABDUL RAHMAN
DECEMBER 2018

REAL-TIME TIME SERIES ERROR-BASED DATA REDUCTION FOR INTERNET-OF-THINGS APPLICATIONS

By

WONG SIAW LING

A dissertation submitted to the Department of Computer and Communication Technology,
Faculty of Information and Communication Technology,
Universiti Tunku Abdul Rahman,
in partial fulfillment of the requirements for the degree of
Master of Science (Computer Science) in
December 2018

ABSTRACT

REAL-TIME TIME SERIES ERROR-BASED DATA REDUCTION FOR INTERNET-OF-THINGS APPLICATIONS

WONG SIAW LING

There are many time series data reduction methods, ranging from primitive data aggregation such as Rate of Change to sophisticated compression algorithms. Unfortunately, many of these existing algorithms are limited to work in offline mode only, data can only be reduced after a certain amount of data is collected. Such offline mode is not suitable for IoT applications such as monitoring, surveillance and alert system which needs to detect events at real-time.

On the other hand, existing real-time time series data reduction techniques often require manual configuration and adaption to intended applications and hardware like IoT gateway. Such requirements prevent effective deployments of data reduction techniques.

This work is inspired by Perceptually Important Points (PIP) data reduction algorithm due to its superior data reduction ability. This work differs from existing PIP in the sense that, we have devised a real-time data reduction algorithm namely error-based PIP Data Reduction (PIPE), that operates with a single value configuration; error rate, which can be used with various sensor data without any priori analysis required. In additional to that, PIPE is simple to the extent that it can be deployed at the sensor node as well.

Through 7 different time series datasets and by comparing the result against the existing data reduction techniques such as GZIP, Real-Time PIP and Rate of Acceleration threshold-based data reduction, the experimental results are promising, the evaluation shows that it is possible that by only forwarding 10% of data, the reduced data produced by PIPE can be used to reconstruct the time series with an accuracy of 0.98 in real-time.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my main supervisor, Dr. Ooi Boon Yaik for all his guidance and encouragement throughout my journey of pursuing Master degree. The completion of this research work and dissertation will not be possible without his consistent motivation and inspiration to me. Besides, sincerely thank you to my co-supervisor, Dr. Liew Soung Yue for his constructive feedbacks and advices on the research methodology as well as this dissertation.

Furthermore, I would like to thank my employer, Hilti Group as well as both of my managers, Mr. Ng Eng Siong and Dr. Christoph Baeck for providing tremendous support and considerate to me being a part-time postgraduate student, allowing me taking time off for my studies and providing feedbacks on my work from the industry perspective.

Finally, my heartfelt gratitude dedicated to my parents, my love, family and friends who being so supportive all the time.

APPROVAL SHEET

This dissertation/thesis entitled "**REAL-TIME TIME SERIES ERROR-BASED DATA REDUCTION FOR INTERNET-OF-THINGS APPLICATIONS"** was prepared by WONG SIAW LING and submitted as partial fulfillment of the requirements for the degree of Master of Computer Science at Universiti Tunku Abdul Rahman.

Approved by:

_____

(Dr. Ooi Boon Yaik) Date: _____
Main Supervisor
Department of Computer Science
Faculty of Information and Communication Technology
Universiti Tunku Abdul Rahman

_____

(Dr. Liew Soung Yue) Date: _____
Co-supervisor
Department of Computer and Communication Technology
Faculty of Information and Communication Technology
Universiti Tunku Abdul Rahman

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: _____

SUBMISSION OF DISSERTATION

It is hereby certified that Wong Siaw Ling (ID No: 15ACM06529) has completed this dissertation entitled "REAL-TIME TIME SERIES ERROR-BASED DATA REDUCTION FOR INTERNET-OF-THINGS APPLICATIONS" under the supervision of Dr. Ooi Boon Yaik (Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology, and Dr. Liew Soung Yue (Co-Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my dissertation in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

_____
(*Wong Siaw Ling*)

DECLARATION

I, Wong Siaw Ling hereby declare that the dissertation is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

_____

(WONG SIAW LING)

Date: _____

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

IoT	Internet of Things
PIP	Perceptually Important Points
PIPE	Error-based PIP Data Reduction
DFT	Discrete Fourier Transform
SVD	Singular Value Decomposition
CS	Compressed Sensing
DTW	Dynamic Time Wrapping
CLIP	Clinical Speech Processing Chain
AMI	Advance Metering Infrastructure
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
MAPE	Mean Absolute Percentage Error
MQTT	Message Queuing Telemetry Transport

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Internet-of-Things (IoT) collects, transmits and analyses data from a wide range of connected devices. Among the popular use cases are environmental monitoring, biometrics data for healthcare, smart homes and cities initiatives. Depending on the sampling rate and type of applications, sensors can generate an enormous amount of data over short timespans. For instance, an accelerometer sensor that are deployed to monitor machinery vibration can generate up to hundreds of reading in a second. As the adoption of IoT grows, transmission of such amount of data becoming challenging because energy, network bandwidth and storage space of sensor nodes are often limited (Papageorgiou et al., 2015a). To overcome the obstacles, multiple works have suggested adopting time series data reduction to reduce the amount of data that need to be processed and transmitted (Papageorgiou et al., 2015a), (Fathy et al., 2018), (Mohamed et al., 2018) & (Feng et al., 2017).

Many time series data reductions are designed for time series data mining years before the emergence of IoT. In general, those existing techniques can be categorized into lossless and lossy data reduction. However, it is difficult to employ those techniques directly with IoT applications which operates with real-time sensing.

Firstly, some of the existing data reduction algorithms, like GZIP compression (P. Deutsch, 1996), time-domain transformation (Agrawal et al., 1993) & (Chan and Fu, 1999) and Perceptually Important Points (PIP) (Chung et al., 2001) can only act upon the entire datasets but not real-time processing. Real-time data collection is critical for IoT use cases like monitoring, surveillance and alerting system to ensure a timely response. For instance, abrupt fluctuation of reading, sudden change and peaks. Therefore, IoT needs a time series data reduction solution which is capable to perform data reduction and transmission in a real-time fashion to ensure essential information is delivered to users in a timely manner.

There are data reduction algorithms like sampling, data filtering and compressed sensing (Rani et al., 2018) can act upon data points at real-time. However, those techniques required threshold setting. Often, the threshold setting is not adaptive to IoT data from different type of sensors and offline analysis is required to set the optimum threshold. For instance, sparsity needs to be known before compressed sensing starts reducing data. Sparsity is varied among different time-series. Therefore, offline analysis need to be done as pre-work.

The needs to reduce real-time IoT data is not new and there are many existing works (Papageorgiou et al., 2015a), (Fathy et al., 2018), (Mohamed et al., 2018) & (Feng et al., 2017). Unfortunately, these existing works are often use-case specific. For example, most of the current works focus on processing only biometrics data (Dubey et al., 2015) and designing a solution specific to

IoT gateway implementation (Papageorgiou et al., 2015a) & (Feng et al., 2017). This indicates it is difficult to design a data reduction solution for real-time IoT application without constraint on computation and implementation.

PIP is an algorithm that is used to extract a subset of important points from time series to achieve data reduction. PIP is subsequently modified to achieve real-time data processing (Papageorgiou et al., 2015a). However, the central issues of PIP such as offline processing and difficulty in algorithm configuration still yet to be fully optimized.

To bridge the gaps that mentioned above, in this work, a novel data reduction approach inspired by PIP, namely Error-Based PIP Data Reduction (PIPE) is proposed. PIPE can achieve real-time data reduction with single error-rate configuration, without requirements such as hardware and data-specific.

1.1 Problem Statement

Time series data reduction is essential for real-time IoT applications to relieve demand of limited bandwidth, energy and storage space. There are existing works focus on design a new data reduction solution, or optimizing existing data reduction algorithm to adopt IoT use case, however, there are issues remains unresolved: -

a) Existing data algorithms are mainly processed data in offline mode, or configuration requires offline data analysis. For instance, PIP can only be used to reduce data after collection is completed, and to the best of

our knowledge, there is no solution in configuring PIP to work at optimum level.

b) Existing real-time data reduction techniques focuses on less-constraint implementation. For instance, IoT gateway that generally provides high compute power compare to a sensor, as well as use-case specific like biometrics time series data reduction, giving little or no insight on whether the same technique can be applied with other use-case or environment.

1.2 Objectives

The goal of this work is to devise a novel data reduction technique that can perform real-time time series data reduction for IoT applications without constraint such as offline configuration, hardware requirements or addressing specific use-cases/applications such as biometric data. Hence, the research objectives can be described as below:

a) To devise a new data reduction technique that is: -
    i. Capable to perform data reduction and forwarding at real-time. The definition of real-time processing of this work is the each of data points of time series will be processed and forwarded when necessary, as soon as it is generated by sensing device

    ii. Requires no offline analysis for configuration. In another word, the configuration of data reduction does not require knowledge

or information that only can be derived via offline computation or calculation

iii. Can be deployed at heterogeneous IoT environment including sensors. The data reduction technique needs to be less compute-intensive and complex so that user has the flexibility to deploy the data reduction technique at any tier of IoT such as the IoT gateway and sensor node.

In short, this research focuses on creating a data reduction technique for real-time time series data for IoT applications.

1.3 Research Contribution

The major contributions of this research work are: -

a. An error-based PIP data reduction (PIPE) is devised. PIPE can simultaneously achieve real time data reduction and forwarding with single error rate configuration.

b. Experiments and evaluations have demonstrated that PIPE can achieves consistent and strong performance across every different kind of sample datasets, proven its capabilities working with heterogeneous time series data.

c. PIPE can be implemented at the sensor node, showing the proposed technique does not need high computation power that can only acquire

from hardware such as IoT gateway. Besides, PIPE can be employed to reduce the power consumption of the sensor node by half.

1.4 Dissertation Organization

This dissertation will be organized as follows. Continue with Chapter 2, literature review on related works is presented. In chapter 3, the proposed work, error-based PIP data reduction (PIPE) will be discussed and explained. System implementation will be discussed in Chapter 4. Chapter 5 illustrates the evaluation process and the presentation of results. Finally, we conclude this work with Chapter 6.

CHAPTER 2

LITERATURE REVIEW

2.1 Time Series Data Reduction

Before the rise of IoT, data reduction has been actively utilized in data mining field to reduce the dimension and size of time series data, so that subsequent data analysis can be carried out in a more effective and faster manner. (Fu, 2011). Similarly, in the context of IoT, data reduction is needed to reduce the size of a dataset that will subsequently reduce the consumption of network, power and storage resources, with new requirements such as real-time processing and hardware constraint for sensor node implementation.

Data reduction, sometimes also known as data compression (the word compression and reduction will be used interchangeably under Chapter 2) can be categorized mainly in two form, lossless and lossy compression (Salomon, 2004). Lossy compression achieves data reduction by losing some information whereas lossless compression has no information loss. Lossless compressions, in general, are operating offline, given the fact the compression can only be done effectively after data collection is completed. Therefore, lossless compression is difficult to be adapted to real-time IoT application. In contrary, as lossy data reduction can tolerate a certain degree of information loss, nonessential information can be dropped at real-time to achieve data reduction. Because of that, re-constructability of reduced data from lossy data reduction is an important metric to ensure original information still preserved.

In subsequent sections, details of existing data reduction algorithms and techniques of both lossless and lossy data reduction will be discussed.

2.1.1 Lossless Compression

Lossless compression is a data reduction method whereby the output of de-compressed data is identical to original data compressed by the compression algorithm. (Salomon, 2004). One of the notable and widely used lossless compression is GZIP (P. Deutsch, 1996). GZIP can be used to compress a chunk of data into a single file which is usually smaller in size. By forwarding a single compressed file, we can effectively reduce the consumption of bandwidth and power during the data transit. However, data compression can only be done after data collection is completed. At best case, GZIP can be modified to perform batch processing, however, the smaller the batch size, the less effective of GZIP compression will be in term of reduction rate, a ratio of the size of compressed dataset over the size of the original dataset. This still holds true for other lossless compression technique like ZLIB (Deutsch and J-L. Gailly, 1996), LZ4 (GitHub, 2018), Zstandard (Facebook, n.d.) and SprintZ (Blalock et al., 2018).

2.1.2 Lossy Compression

Lossy Compression achieves data reduction by removing data points from the original time-series sequence. Therefore, the main design consideration of lossy compression technique is how the lossy compression algorithm decides which data to be removed from original data, without losing too much meaningful information.

Sampling (Aström, 1969) is one of the simplest forms of lossy compression. Sampling is performed based on *N* samples of time series at an equal spacing of *h*. However, sampling at an equal spacing can run into the risk of losing important information if events happen in between the spacing.

Simple aggregation-based lossy data reduction like segmented mean (Yi and Faloutsos, 2000) and Piecewise Aggregate Approximation (PAA) (Keogh and Pazzani, 2000) leverage the mean of each segment as the features representation of the original time-series. Segmented Mean and PAA are aggregating the mean result at an equal spacing, important information can be lost if it happens in between the spacing. Therefore, to solve such issue, Adaptive Piecewise Constant Approximation (APCA) (Eamonn Keogh et al., 2001) is proposed, which allows the mean to be calculated with an arbitrary length of a segment. The authors of (E. Keogh et al., 2001) have mentioned, in general, there is three type of time series segmentation approaches which is Sliding Windows, Top-Down and Bottom-Up. The authors were aiming to optimize those algorithms by introducing Sliding Window and Bottom-Up (SWAB) algorithm. Similarly, Schoellhammer et al. (2004) have proposed a technique to represent data with a set of aggregated lines based on error-bound. In general, segmentation-based data reduction techniques leverage aggregated representation such as mean, line segment to present the original time series. Such aggregated data could impose challenges for the data analytics process. Especially when the user is interested to know the actual value of the data points, or critical points that contribute to an event. For instance, it is important to know

the actual heart rate of a person to determine whether it is exceeding a dangerous threshold.

Data filtering is an option to achieve data reduction due to its simplicity in term of implementation. Data filtering decides whether to retain points by comparing the current value to a pre-defined threshold and it can be implemented at real-time. Papageorgiou et al.(2015a) have proposed a time-series forwarding handler to only forward value when it falls under a specific range, greater and lower than a threshold. The authors have devised a decision-making framework which can perform switching between handler dynamically. Similarly, (Toni et al., 2013) has proposed a time series data filtering based on single value, the rate of change of data and the rate of acceleration of data. Besides, the work has also devised a data reconstruction scheme and concluded that pairing the data reconstruction scheme with data filtering based on the rate of acceleration reveals the best results for human movement monitoring. The main challenge of adopting data filtering techniques is to define the right threshold value to produce optimum results. For instance, work by Toni et al. (2012) has concluded different threshold values can create a significant impact to reduction rate results, however authors offer no insight on how to set the optimum threshold. Work (Sarker et al., 2016) has done substantial pre-analysis to compute a threshold value that would fit their proposed data filtering technique, proven that data cannot be reduced and transmitted at real time without such pre-works to set the threshold.

Apart from reducing time series in time domain directly, there are data reduction methods which are applied on frequency domain such as Discrete Fourier Transform (DFT) (Agrawal et al., 1993), Discrete Wavelet transform (Chan and Fu, 1999) and Singular Value Decomposition (SVD) (Cadzow et al., 1983). Data reduction is achieved by converting time domain to space domain. These data reduction techniques can be used to extract features from time series data, however, the process can only be done offline, which means after data collection is completed. Furthermore, time domain transformation usually is compute-intensive.

Compressed Sensing (CS) theory is emerging recently as a domain of data reduction and signal compression. CS can be used for the acquisition of signal that is either sparse or compressible, or in another word, the signal or time series contains small amounts of non-zero or significant data, while the rest are zero or non-important data, which can be discarded. (Rani et al., 2018). By exploiting the sparsity of data, compressed sensing is used to acquire data with a fewer sample and then be reconstructed to recover the original data. A lot of works have been introduced under the domain of compressed sensing. In relation to the IoT applications, work (Li et al., 2013) has proposed a CS framework for IoT deployment. CS requires sparsity of data to be known in advance, therefore ,work (Chen et al., 2012) has proposed an optimized CS for IoT without the need of pre-knowledge of sparsity. Besides, authors of (Zhang et al., 2018) has proposed a method that is stable and robust in recovering from the compressed signal using compressed sensing, for ECG application. Not all IoT application generates sparse data, for instance, acceleration sensor that

attached to a speeding car will generate dense data. (Amarlingam et al., 2016). Therefore, Compressed Sensing in limited to certain use-case only.

Perceptually Important Points (PIP) (Chung et al., 2001) achieves data reduction by acquiring important points and discard the rest. PIP algorithm is less compute intensive compared to other data reduction techniques like GZIP compression and signal transformation. Due to its simplicity, PIP has been repeatedly modified and optimized for different use-cases. For instance, the authors of (Zaib et al., 2004) utilized the PIP framework to devise a pattern recognition technique, work of (Tsinaslanidis and Kugiumtzis, 2014) has devised a time series prediction scheme based on PIP and Dynamic Time Wrapping. Phetking et al. (2009) use PIP to devise a method to index financial time series data and Fu et al. (2017) has optimized the PIP algorithm to adapt with use-case such as big data and data mining analytics. When speaking about IoT context, PIP cames with two central issues that hinder people to adapt it with IoT applications and reduce sensor data. Firstly, data can only be processed by PIP after collection is completed, PIP can only act upon the entire datasets. Second, configuration needs to be done to control the amount of data required to be reduced. There is no guidelines or frameworks which helps in setting the optimum configuration. Hence, user is required to determine the configuration on their own and very often, the configuration will be different based on different use-case or requirements. The authors of (Papageorgiou et al., 2015a) has proposed a real-time PIP algorithm based on caching for IoT gateway, performs data reduction for IoT applications. Feng et al. (2017) enable real-time time-series data reduction based on PIP by introducing multi-tiers processing at

IoT gateway and edge device. Both works offer real-time processing based on PIP, however, maybe due to compute-intensive, the implementation requires powerful device like IoT gateway and not for sensor node.

With this research work, we decided to design a new data reduction algorithm to fulfil our design objectives which are real-time data reduction and forwarding, no offline analysis for configuration and can used for heterogeneous hardware deployment and IoT sensors data.

2.1.3 Perceptually Important Points (PIP)

The fundamental concept of PIP is to identify important points from original data and discard the rest. By preserving a subset of important points, PIP retains the information presented in the data and the set of important points can be used to recover the original data as well. The PIP identification process is first proposed by (Chung et al., 2001) for the use of pattern matching for financial analysis purpose.

The PIP identification process can be explained as followed: -

a) Assume time series $C = \{C_1 \dots C_n\}$ has in total n number of data points. The first two PIPs are C_1 and C_n.

b) To identify the third PIP, C_1 and C_n will be connected to form a line. The point that is furthest from the line will then be the third PIP.

c) To continue to identify the fourth PIP, a line will be formed between adjacent PIPs, the point that furthest away from the line will then be elected the fourth PIP.

d) The previous step will be repeated to identify subsequent PIP until the number of required PIPs has reached, otherwise, the process will only be ended until no remaining data points to be processed.

Figure 2.1 – 2.4 has summarized the entire PIP identification process of 4 PIPs.

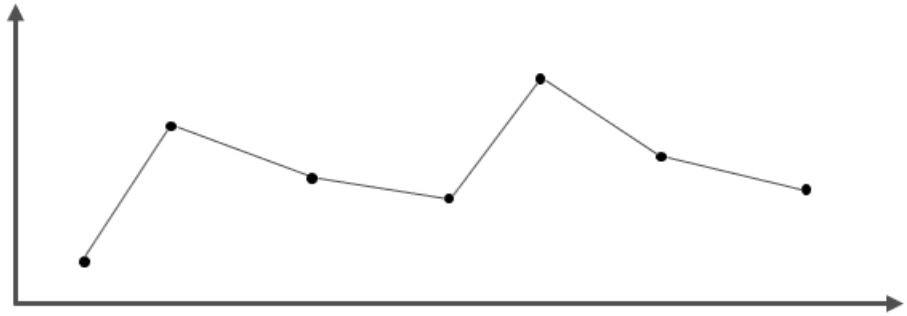

Figure 2.1: The Graphical Model of Time Series C

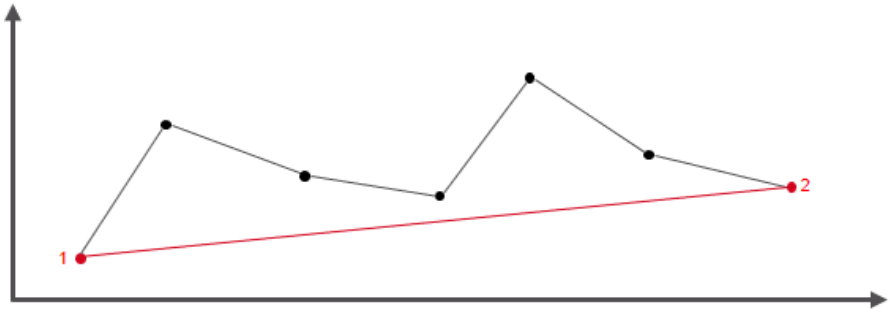

Figure 2.2: Identification of First Two PIPs

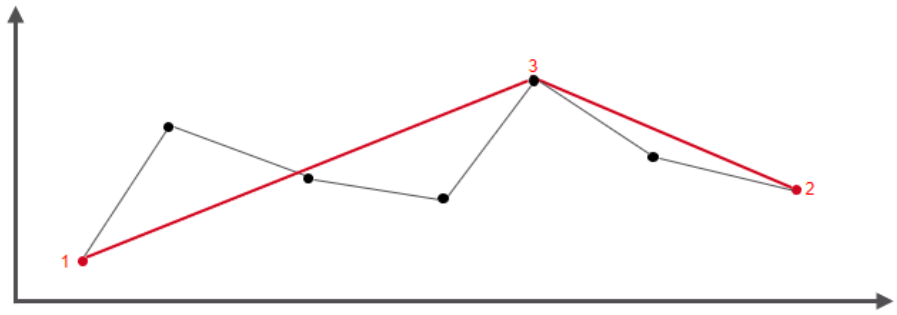

Figure 2.3: Identification Process of Third PIPs

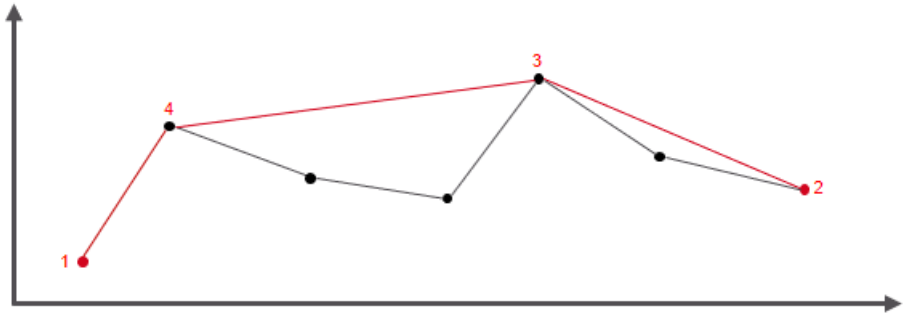

Figure 2.4: Identification Process of Fourth PIPs

As shown, all data points have to be collected and known before the PIP identification. Therefore, the process is inherently offline. Besides, unless the user explicitly configuration the number of iteration, PIP identification will be iterated until all points are indexed based on its importance order. Therefore, such configuration is crucial and can have a significant impact to data reduction performance. This research work will focus on solving these two issues by optimizing PIP algorithm, and employ the optimized PIP to devise real-time data reduction for IoT applications.

2.2 Existing Data Reduction Solutions for IoT

Several works have been proposed to solve data reduction challenges for IoT applications. Work by Papageorgiou et al. (2015a) has proposed a data reduction framework for Edge and IoT gateway implementation namely NECtar,

that automates the switching between different data reduction algorithms includes Sampling, Piecewise Approximation, thresholding filtering, change detection and real-time PIP. The authors focus on illustrate the optimization made on the PIP algorithm and concluded by only forwarding 1/3 of data items, real-time PIP can achieve accuracies between 0.76 and 0.94. To enable real-time PIP, caching is used to store the past processed data points and project the future points based on history. PIP identification is executed upon every incoming data point. The result shows caching more items like 500 data points yields a better result. The processing of each data points is $O(n^2)$. It is compute-intensive and the result has shown with a light resources gateway, the compute process can take up to 1 sec for 400 cache size. Hence, the proposed technique is designed for more powerful hardware like IoT gateway.

The same authors further optimize their solution with work (Papageorgiou et al., 2015b). A feature call reconstruct-ability table is added to identify the best data reduction strategy based on its reconstruct-ability. However, the computational complexity of the enhanced technique is not simplified. The solution still not suitable to deploy at sensor nodes.

Work of (Feng et al., 2017) focuses on enhancing the PIP algorithm by introducing several methods like interval restriction, dynamic caching and weighted sequence selection. These methods are deployed separately in both gateway and edge tiers.

Dubey et al. (2015) strive to solve the challenge of collecting healthcare data with Fog Computing. They have proposed a IoT gateway data reduction framework that includes Dynamic Time Wrapping (DTW), Clinical Speech Processing Chain (CLIP), Fundamental Frequency as well as compression. The framework is deployed with an Intel Edison as the IoT Gateway.

Alduais et al. (2016) have proposed an IoT-based data collection method that aims to reduce the number of transmitted messages via a sink node, which can be defined as IoT Gateway. To reduce the number of transmissions, the authors proposed to detect and send only rare events based on absolute differential values, with a threshold value. The authors did not mention the strategy of defining the threshold value.

The authors of (Fathy et al., 2018) proposed an adaptive method to minimize the data transmissions between the sink and sensor nodes. When sensed values deviate significantly with a pre-defined threshold, it will be transmitted, otherwise, the value will be discarded. The proposed methods operate with two tiers with sink and sensor nodes.

Mohamed et al. (2018) have proposed an adaptive framework for real-time data reduction in Advanced Metering Infrastructure (AMI). The data reduction is done based on forecasts, when the smart meter reading is close to the forecasted value, it will be discarded, otherwise, it will be transmitted. To ensure the framework is adaptive to the pattern of smart meter data, the framework allows switching between forecasting models to ensure the

reduction scheme is most suitable one to the current data pattern. The framework consists of a total 5 components. Each component consists of different calculation that has to be deployed in different tiers of AMI.

Last but not least, Maschi et al. (2018) have proposed a real-time data summarization concept at sensor node level based on parameters. Parameters are stored and updated in a database that is located at Cloud. Sensors can perform a local decision whether to transmit data to the server by communicating and acquiring the parameter from the cloud database.

In summary, based on our definition of real-time, many of the existing works have yet achieved real-time data reduction and forwarding, requires no offline analysis for configuration, capable to reduce heterogeneous IoT data and can be deployed in heterogenous IoT hardware including sensor nodes simultaneously. Most of the works focus on solving one or the other requirements. For instance, some works focus only on real-time IoT gateway data reduction, while the other focus on solving specific use-cases like biometrics time series data reduction.

2.3 Measurement of Error

This research work aims to devise a real-time error-based data reduction technique. Therefore, error estimation will be the crucial component of the proposed technique. The design goal is to utilize error estimation as the gauge of whether data need to be discarded or forward. There are different variants of statistical error measurement is available, the discussion is followed.

One of the straight-forward error measures is Euclidean distance. The Euclidean distance of each data points can be calculated via the formula of 2.1.

$$dist((x,y),(a,b)) = \sqrt{(x-a)^2 + (y-b)^2} \tag{2.1}$$

Similarly, Dynamic Time Wrapping (DTW) is proposed by Keogh and Ratanamahatana (2005) to serve as a distance measure, which can be operated with time series data that comes with different lengths and sizes. Both techniques offer fine-grain comparisons, between each pair of actual and forecasted/recovered data points, but has no unified or aggregated result when compare between two datasets where length is more one.

Apart from that, there are many other error measurements that can be used to estimate error for time series data. Shcherbakov et al. (2013) have done a comprehensive review on each of the error measurement. The authors have mentioned there are multiple variants of error measurement. Among the widely used techniques are absolute forecasting error and measures based on errors such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). Each of these techniques came with different limitations. MAE and RMSE produce errors that can only be associated with the specific time series data that used for calculation, the error reading cannot be used to associate with other time series data. For instance, the error of temperature reading cannot be interpreted together or compare with the error of humidity reading. This issue does not apply to MAPE, which measures

error based on percentage. However, MAPE cannot process time series that contains value zero due to division by zero.

Jaccard Coefficient is introduced by Jaccard (1901). For two set of time series data X, Y, Jaccard Distance between the two time-series is $D(X,Y) = 1 - J(X,Y)$. To adapt Jaccard Distance to vectors as well, the Weighted Jaccard Distance is then introduced (Chierichetti et al., 2010). The Weighted Jaccard Distance formula is defined as below: -

$$J(X,Y) = \begin{cases} 1 - \dfrac{\sum_{i=1}^{n} \min(X_i, Y_i)}{\sum_{i=1}^{n} \max(X_i, Y_i)} & if \sum_{i=1}^{n} \max(X_i, Y_i) > 0, \\ 0 & if \sum_{i=1}^{n} \max(X_i, Y_i) = 0 \end{cases} \quad (2.2)$$

Weighted Jaccard Distance is a ratio-based calculation. It can be used to measure the dissimilar between two time series data and the ratio-based result is easy to interpret and make comparisons between different applications that collects different time-series data (Peng et al., 2016). This research work will incorporate Weighted Jaccard Distance with the proposed technique PIPE.

CHAPTER 3

Proposed Solution

3.1 Time Series Data Reduction Design Goal

The concept of Internet-of-Things is not only about sensing the physical environment but includes using the sensed data to react with events at real-time. For instance, turn on light based on human movement, raise the emergency alarm if machinery is vibrating at an unusual threshold. Because of that, in the context of IoT, the data reduction technique need to process and forward data at real-time.

Reduction rate is one of the keys and most frequently used metrics to measure the performance of a data reduction technique. However, it is important to note retaining important message or information from the original time series is more crucial especially for alerting application that cannot tolerate missing event. In this work, one of the design goals of the data reduction technique is then to prioritize correctness of more reduction rate. When the time series contains a lot of events or patterns, the information should be retained rather than discarded to achieve high reduction ratio.

In chapter 2, we have done a throughout review on existing data reduction techniques, including works that focus on solving IoT challenges. One of the common problems among the existing works is non-adaptive configuration. For instance, threshold filtering requires users to define threshold based on the statistical characteristic of the time series; compressed sensing

requires the sparsity of data to be known in advance; PIP requires users to define the number of PIP identification iteration. Such requirements will hinder large-scale general purpose IoT deployment since human intervention is required. This work aims to minimize the need for configuring threshold values for data reduction processing, especially the threshold value can only be computed offline.

Besides, most of the existing works are designed to reduce data at the IoT gateway level. While this may due to the needs of computational power, this imposes a challenge for some use0case that has only two tiers setup: sensor and endpoint. Therefore, the second aim of this work is to design an algorithm that is less-compute intensive that can be deployed with sensor nodes, as compared to existing work that reduces data at the gateway level.

In summary, the design goals of the proposed work Error-Based PIP Data reduction (PIPE) are: -

a) Real-time data processing and forwarding
b) Prioritize correctness over reduction rate
c) A Single configuration that can be applied with heterogeneous sensors data, which required no offline analysis configuration setting.
d) Less compute-intensive and can be deployed at sensor nodes.

3.2 PIPE Workflow Overview

Figure 3.1 shows an overview of the proposed data reduction technique, Error-Based PIP Data Reduction (PIPE). PIPE operates with an error rate. Error rate indicates the degree of missing information that user can tolerate. This is the only parameter needed and it works across all type of sensor data.

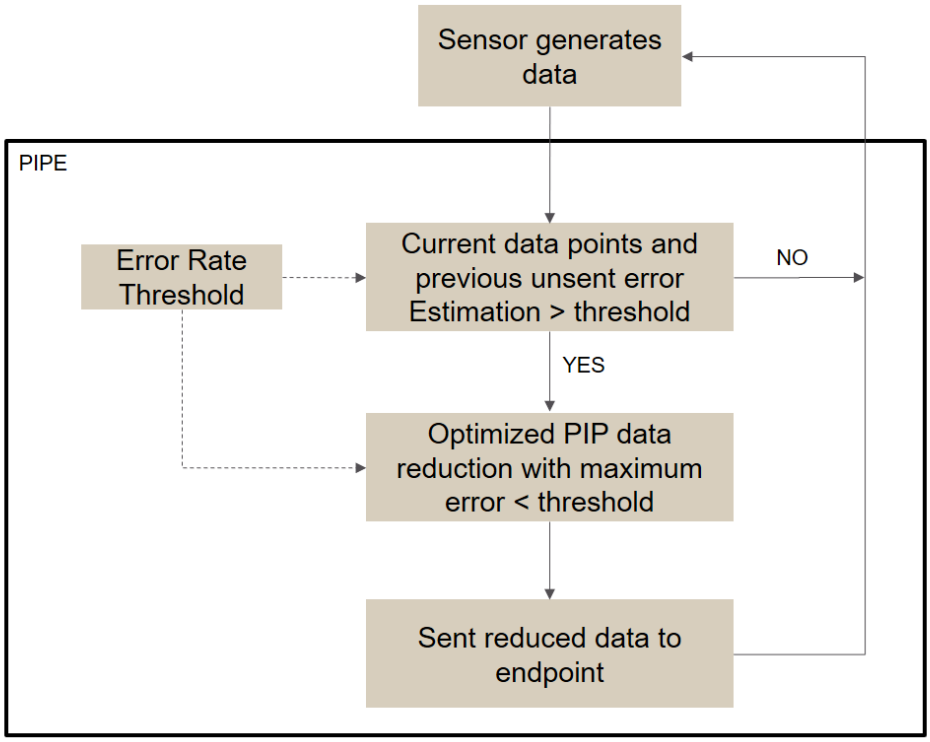

Figure 3.1: High Level PIPE Workflow

Each data points, that is generated by sensors will then be processed with PIPE. The workflow first starts with estimating the error can be generated by the data point, if it is not sent to the endpoint. Estimated error is computed with the formula of Weighted Jaccard Distance, as shown with equation 2.2. If the estimated error is lesser than the defined threshold, PIPE will continue collecting and processing the subsequent data points. If the error is more than

the defined threshold, PIPE will trigger the data reduction process and extract a subset of important data points using PIP (Fu et al., 2008). This ensures the subset of important data points can be reconstructed with error lesser or equals to the defined threshold.

The extracted important points will then be sent to the user endpoint. The workflow continues with processing the subsequent points generated by the sensors. As such, each data point is processed, reduced and forwarded to the endpoint at real time when the error generated is more than the defined threshold.

3.3 Error Rate

As discussed, accuracy is crucial, especially for lossy method to ensure information loss is kept minimal. Therefore, one of the design goals of PIPE is to prioritize accuracy over data reduction rate. To ensure accuracy is consistently kept at desire level, we need impose a parameter for PIPE to control the overall information loss or in another word, error that is generated by the data reduction process. In this work, we call such parameter as the error rate threshold. Error rate threshold will be used to monitor the error generated as if the current and previous unsent data points are discarded from the original dataset. If the error exceeded the threshold, the optimized PIP will be employed to extract a set of important points, whereby the set of important points can be used to recover the original datasets. The recovered datasets will have smaller or equal error to the error rate threshold.

There is another design goal: a single configuration that can interoperate with heterogeneous sensors data. This need to be reflected in the error rate threshold. To fulfil this requirement, we have done studies on existing error measurement method in section 2.3.

With the information available, we have concluded that we will employ Weighted Jaccard distance, a ratio-based error calculation as the error rate threshold.

The Weighted Jaccard Distance reading falls between 0 and 1. 0 indicates both datasets are identical and 1 indicates both dataset is completely different. The reading in between can be used to indicates the error of the approximated datasets vs the original datasets. As it is ratio-based, it can be adapted for any time series data generates by different sensors, regardless of the range of number that the sensor operates on.

As mentioned, the error rate threshold can accept values ranging from 0 to 1. One of the reference settings that users can be adopted is the sensor margin of error. For instance, Schoellhammer et al. (2004), who proposed an aggregation-based data compression has explained that sensors, in general, came with a hardware-specified margin of error. As a result, sensors generate noise within the margin of error and therefore it can be used as an indicator to remove noises from original datasets, and achieve data reduction. For example, if a temperature sensor has a margin error of 2%, the error rate threshold can then be configured at 0.02.

3.4 Error Estimation for Current and Previous Unsent Data Points

The process of error estimation can be divided into two main operations: estimator construction and error computation. The workflow is depicted as Figure 3.2.

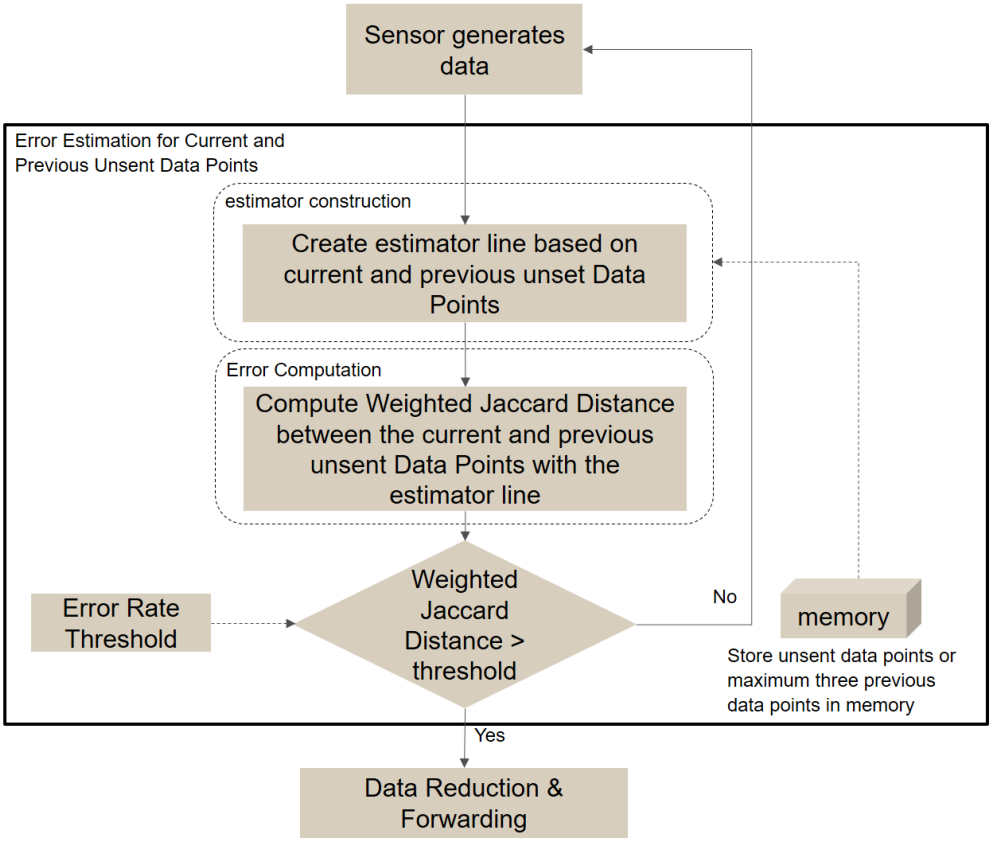

Figure 3.2: Error Estimation Workflow

To estimate the error can be generated by current and previous data points is unsent or discarded from original datasets, we need to create an estimator that we can use to make comparisons with. In this proposed work, the

estimator will be a linear line that is connecting the first and the last points of current and previous unsent data points. This estimator - linear line will then be used to compute the Weighted Jaccard Distance.

If the Weighted Jaccard Distance is more than the configured error rate threshold, the process will be continued by data reduction and forwarding, else the current data points will be stored in the memory as unsent data points for the subsequent processing.

Assuming the time series sequence generated by sensor is $C_o = \{C_{o1}, C_{o2}, \dots, C_{on}\}$. $C_{o1..4}$ will be collected unconditionally prior any processing. 4 points are required to ensure PIP is producing meaningful results.

In order to construct a linear line as an estimator, which represented as C', first we use C_{o1} and C_{o4} to estimate the 5th point C'_{o5} using two-point form (Weisstein, n.d.). The Two-point form is often used to find a point on a line or the slope of the line. The formula of two-point form is given as below: -

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \tag{3.2}$$

For time series data, the combination of (x, y) can be defined as x is the position of the data points within the datasets, and y is the data point.

Therefore, C'_{o5} is estimated with the equation below: -

$$C'_{on+1} = \frac{C_{on} - C_{o1}}{n-1}(n) - C_{o1} \tag{3.3}$$

Once the C'_{o5} is known, the estimator linear line C' will be interpolated between given the point of C_{o1} and C'_{o5}, by using the same two-point form depicted with Equation 3.3. When C' is ready, the error will be calculated by computing the Weighted Jaccard Distance between C' and C. The equation is shown below.

$$J(C_{on}, C'_{on}) = 1 - \frac{\sum_{i=1}^{n} \min(C_{on}, C'_{on})}{\sum_{i=1}^{n} \max(C_{on}, C'_{on})} \tag{3.4}$$

If the Weighted Jaccard Distance does not exceed the configured error rate threshold, the data points will be stored in the memory for subsequent iteration of estimation construction and error computation. Otherwise, if the error exceeds the threshold, C_o will be reduced and forwarded to endpoint, and then purge from memory except the last three data points. Three data points is kept for the next iteration of error estimation.

Figure 3.3(a)-(f) has shown the chronological order of error estimation processing. The steps can be explained as: -

a. The process is started with at least 4 points is collected unconditionally. With C_{o1} and C_{o4}, the subsequent point C'_{o5} is predicted to aid in the upcoming error computation

b. The red estimator linear line is interpolated, between C_{o1} and C'_{o5} and Weighted Jaccard Distance is computed between the red estimator line and black line that is representing the original datasets.

c. As the Weighted Jaccard Distance in computed is lower than configured threshold, the process is continued with the subsequent incoming points C_{o5}.

d. Step b is repeated. This time, the Weighted Jaccard Distance has exceeded the configured error rate threshold, data of C_{o1} to C_{o5} is reduced and forwarded to endpoint. The data is purged from memory after the data forwarding is completed, except the last three points.

e. As the last three points is retained, the processing is continued with new incoming point C_{o6}.

f. Step b is repeated.

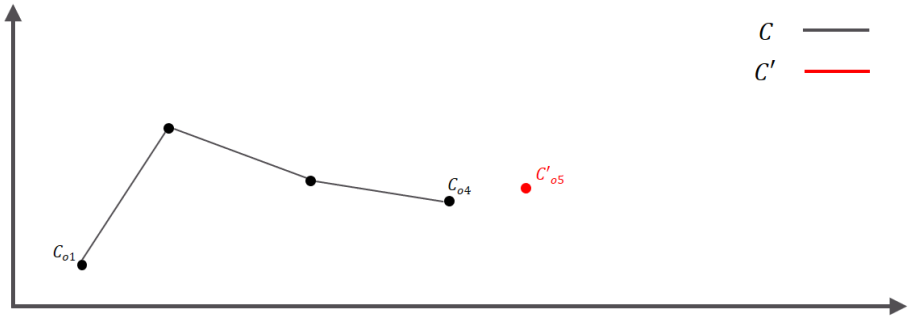

Figure 3.3 (a)

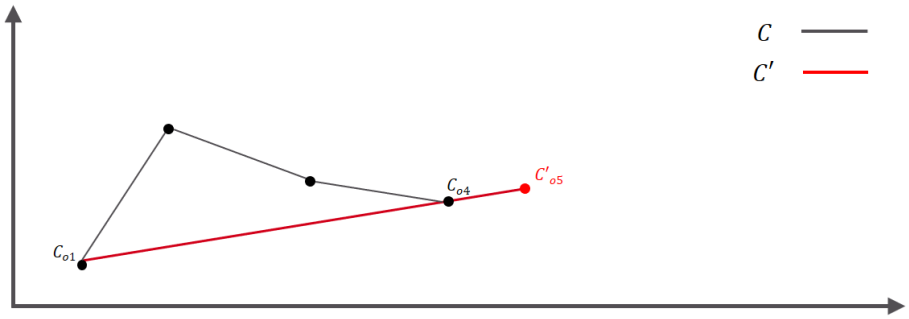

Figure 3.3 (b)

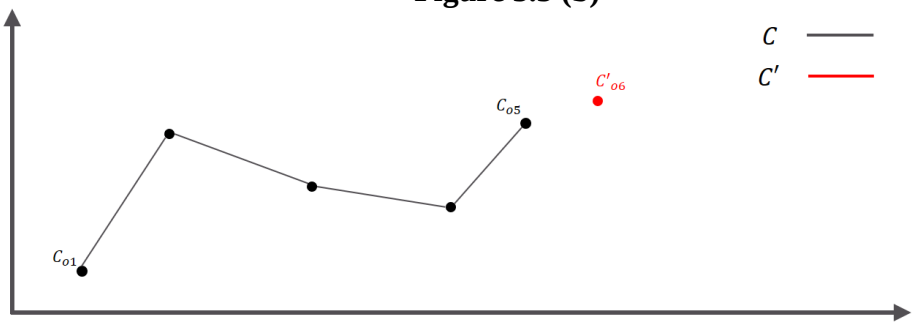

Figure 3.3 (c)

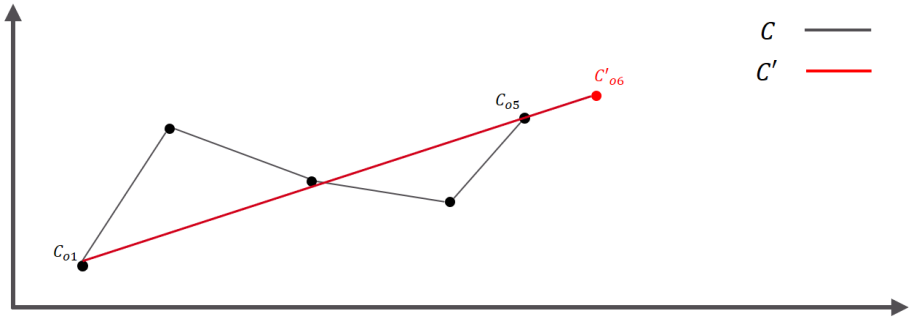

Figure 3.3 (d)

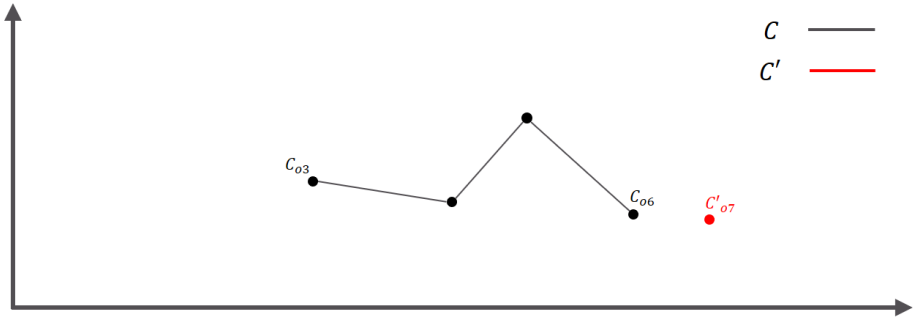

Figure 3.3 (e)

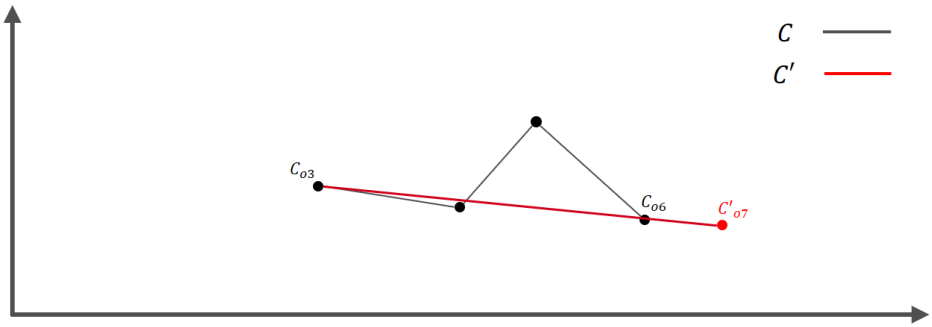

Figure 3.3 (f)

3.5 Optimized PIP Data Reduction and Forwarding

Figure 3.4 is showing the workflow of Optimized PIP Data Reduction and Forwarding process.

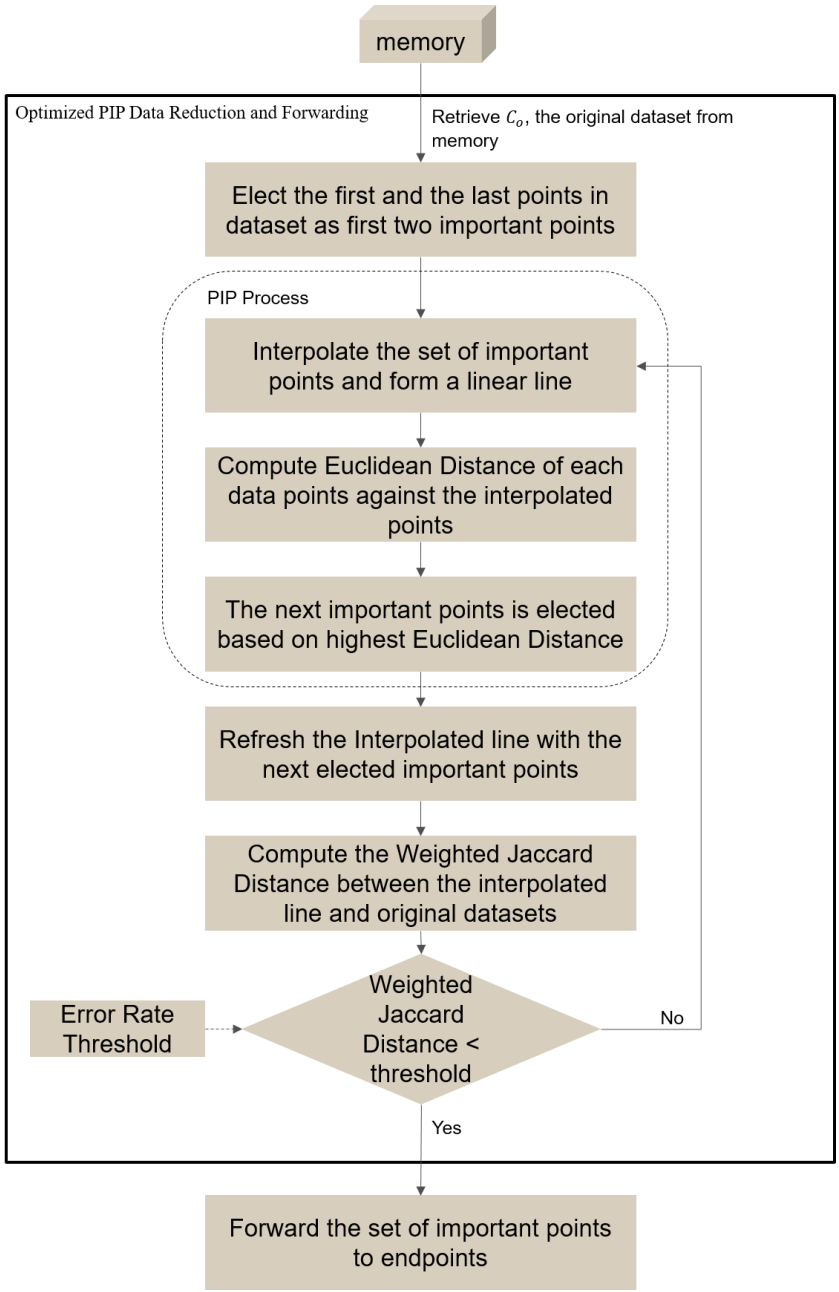

Figure 3.4 Optimized PIP Data Reduction Workflow

When the error for not sending current and previous unsent data points is exceeding the configured error rate threshold, the current and previous data

points will be reduced with optimized PIP, and the subset of important points will be forwarded to the endpoints.

The process starts with retrieving the datasets stored in memory C_o. The result of the optimized PIP data reduction is a set of important points, extracted from the original time series. The set of important points is depicted as $C_r = \{C_{r1}, C_{r2}, \dots, C_{rm}\}, m < n, C_r \subset C_o$.

The PIP data reduction starts with elect the first and the last points as the two important points, C_{r1} and C_{r2}. To identify C_{r3}, C_{r1} and C_{r2} will be connected as a line with interpolation. C_{r3} will be the furthest from the interpolated line. The distance between the original datasets and the interpolated line is computed with Euclidean Distance.

$$D((X_r, Y_r), (X_o, Y_o)) = \sqrt{(X_r - X_o)^2 + (Y_r - Y_o)^2} \tag{3.5}$$

When a new important point is selected, the previous interpolated line will be refreshed by making the connection to the new important points. Weighted Jaccard distance is computed again to examine if the set of important points is sufficient to reconstruct the original datasets with the error rate lower of equal to the configured threshold.

If the Weighted Jaccard Distance higher than the configured error rate threshold, optimized PIP iteration will be continued to search the next important point. If the Weighted Jaccard Distance is lower than the configured error rate,

the PIP process will be stopped and the set of extracted important points will be forwarded to the endpoint, and the rest will be discarded, except the last three points for subsequent processing.

Figure 3.5(a)-(d) illustrates the chronological order of optimized PIP data reduction.

a) The optimized PIP process starts with electing the first and the last points as the first two important points.

b) The first two important points, C_{r1} and C_{r2} will be connected. The connected line in red will be used to search for the third important points.

c) The third important points will be the furthest away from the connected lines. The distance is expressed in Euclidean Distance. As soon as C_{r3} is identified, the connected line will be refreshed to connect with C_{r3}. The Weighted Jaccard Distance between the red line connecting important points and the black line that is connecting the original datasets will be computed. In this case, the Jaccard Distance is not lesser or equal to configure error rate threshold. Therefore, the PIP iteration is continued.

d) The fourth important point, C_{r4} is identified by repeating the step c. In this case, the Weighted Jaccard Distance is lesser than the configure error rate threshold. Therefore, the PIP iteration is stopped and $C_{r1..4}$ is forwarded to endpoint.

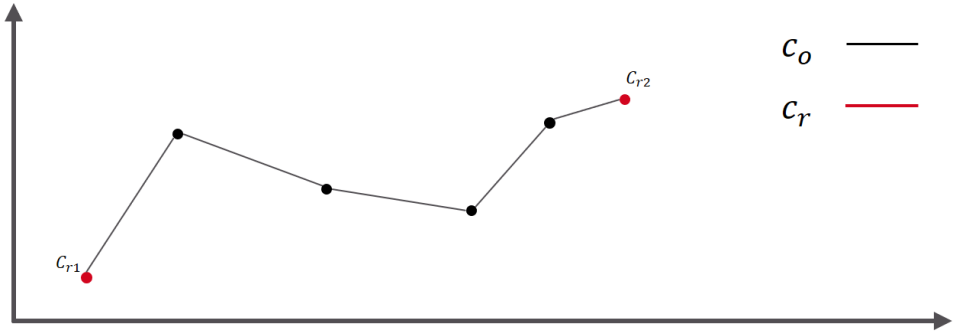

Figure 3.5 (a)

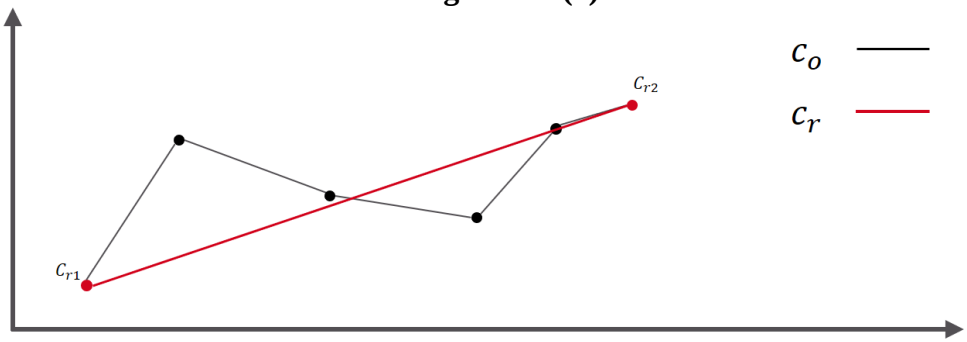

Figure 3.4 (b)

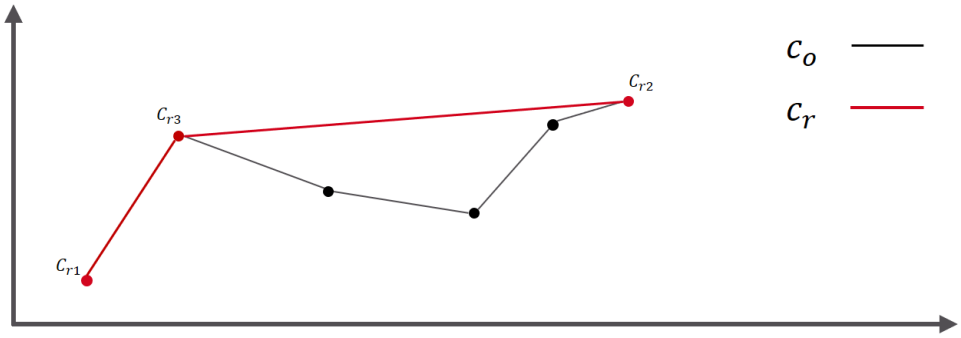

Figure 3.4 (c)

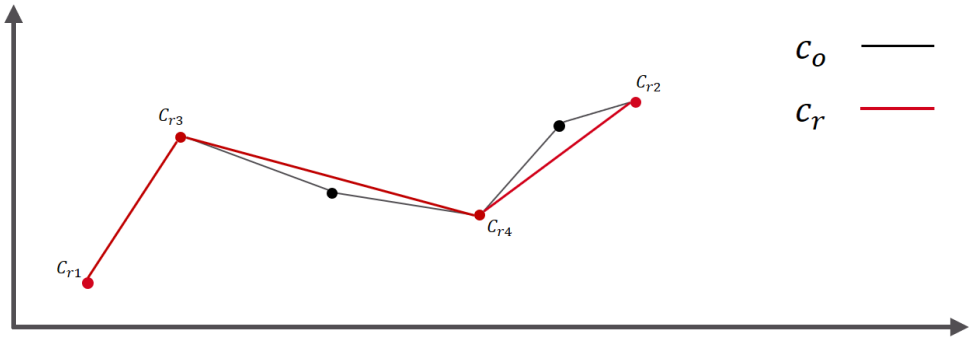

Figure 3.4 (d)

3.6 Design Goal Revisit

Section 3.2 to section 3.5 has illustrates the design and workflow details of our proposed work PIPE. By introducing error rate threshold and error estimation, we have successfully devised a data reduction technique which are: -

a. Data is processed and forwarded at real-time whenever the error generated by not sending data points is exceeding the configured error rate threshold.

b. The error rate threshold controls the accuracy of the reduced datasets. The optimized PIP iteration continues to search more important points to ensure the set of important points can be used to reconstruct original datasets with error rate lesser or equal to the error rate.

c. The error rate threshold can be adapted to any time-series sensor. As discussed in section 3.3. The sensor margin error can be referenced as the error rate threshold setting.

d. The devised algorithm is not complex and can be implemented at any tiers of IoT application, including sensors or microcontroller.

Together with research objectives, these design goals, especially point b. and d. will be verified and validated in Chapter 5.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 Introduction

We have discussed the concept of error estimation and optimized PIP data reduction in Chapter 3. To validate and evaluate the concept of PIPE, we implemented the complete PIPE concept using Arduino Framework (Kravets, n.d.). Arduino Framework is chosen because it is currently one of the most widely-used open source electronic platform for IoT applications. Arduino application can be deployed with any Arduino micro-controllers. Many sensors and libraries are supporting Arduino deployment due to its simplicity and inexpensive in term of cost, though it might not fit for some complex applications since its computing capacity and storage limited. (Barbon et al., 2016).

The program is written in C++ as it is one the supported language by Arduino Framework. In term of data collection, there are two different implementations has been devised: Simulation and physical implementation. A different implementation is needed for simulation because there will be no sensor generating actual data. Instead, sample dataset that has all the data points available in advance will be used. Therefore, a special program that takes in sample datasets as input and mimics the process of the sensor generating data is needed.

Besides, an actual implementation of sensor node with PIPE based on vibration sensor will be described and elaborated as well under this chapter.

4.2 PIPE Implementation

As discussed in Section 3.2, there are two main components of PIPE, error estimation and optimized PIP data reduction with forwarding. Both components will be implemented as two individuals function before adapting it to the main application. Figure 4.1 is showing the main function of the PIPE application. The application is initiated with collecting the first 3 points unconditionally as one-time initialization. When the fourth point and onwards is collected, the error_estimation function is triggered to estimate the error generated if the current and previous data points not forwarded. If the result is more than the error rate threshold, the pip_reduce_forward() will be triggered, else the process will then continue receiving the next incoming point.

```
Variables:error rate threshold, error_threshold
          Original time series array  C_o, oriseq
          The newly generated data points, cur_point

Output:   NULL

Procedure:

//one-time initialization
FOR (i = 0; i++; i<3)
     cur_point = sensor generates data points
     oriseq.add(cur_point)
ENDFOR

WHILE
     cur_point = sensor generates data points
     oriseq.add(cur_point)
     IF (error_estimation(oriseq) >
```

```
     error_threshold)
     THEN
          pip_reduce_and_forward(oriseq,
          error_threshold)
     ELSE
          continue
     ENDIF
ENDWHILE
```

Figure 4.1: Pseudocode of the Main Function of PIPE

4.3 The Error Estimation Function Implementation

For error estimation function, a point ahead will be predicted using two-points form. After that, the first points in the original sequence will be connected with the predicted points using interpolation, to form the estimator line. The estimator line is used to compare with the original sequence to compute the Weighted Jaccard Distance, which representing the error as if the original sequence is not forwarded to endpoints. The function ends with returning the error reading to the main function.

```
Variables:Predicted points, p'
          The estimator line, C', estimated_seq

Output:   The Weighted Jaccard Distance that
          representating error, error

Procedure:
function error_estimation(oriseq)
     P' = predict {oriseq.size() +1 } points with
     two points form.
     Estimated_seq = interpolation(oriseq[0], p')
     error = jaccard_distance (estimated_seq,
     oriseq)
     Return error
end function
```

Figure 4.2 Pseudocode of The Main Function of error_estimation

4.4 Optimized PIP Data Reduction and Forwarding Implementation

Figure 4.3 shows the optimized PIP data reduction and forwarding. The function starts with electing the first and last points from original sequences as the 1st and 2nd important points. The process continues with searching the 3rd important points. The 3rd important points will be the furthest away from the line that is connecting the previous two important points.

Once the 3rd important point is identified, Weighted Jaccard Distance will be computed to examine if the set of important point is sufficient to recover the original datasets with error no more than the configured error rate threshold. If no, the process will continue to search more important points. If yes, the important points will be sent to the endpoint by publishing to MQTT topic that is subscribed by user. All the data points stored in the memory will then be purged, except the last 3 points from original datasets.

```
Variables:The set of important points C_r, imp_seq
          The line that is connecting the set of
          important points C_rc,
          connect_seq
          The Euclidean distance, eudist
          The largest Euclidean distance, maxeudist
          The Weighted Jaccard Distance that
          reprsentating  error, error
          The position of the data point within the
          sequence, selection
Output:   NULL

Procedure:
function pip_reduce_and_forward(oriseq, threshold)
     imp_seq.add(oriseq[0])
     imp_seq.add(oriseq(oriseq.size()))
     connect_seq= interpolation(imp_seq())
```

```
WHILE (error < threshold && imp_seq.size() <
oriseq.size())
     eudistance = 0
     maxeudistance = 0
     selection = 0

     FOR (i = 0; i++; i < oriseq.size())
          eudist =
          EuclideanDistCalculate(oriseq[i] –
          connect_seq[i])
          IF eudist > maxeudist THEN
               maxeudist = eudist
               selection = i
          ENDIF
     ENDFOR
     imp_seq.add(oriseq(selection))
     connect_seq= interpolation(imp_seq())
     error = jaccard_distance (connect_seq,
     oriseq)
     IF error < threshold THEN
          break
     ENDIF
ENDWHILE

sort imp_seq based on the original data points
position
mqtt.publish(imp_seq, data)
purge imp_seq and connect_seq
Purge oriseq and retain the last 3 points
```

Figure 4.3 pip_reduce_and_forward Implementation

4.5 Additional Program for Simulation

For evaluation purpose like testing against the existing datasets, an additional program is required to simulate the data generation in time series streaming manner. Figure 4.4 is showing the process of streaming the sample datasets, that is in csv format by publishing to an MQTT topic. The only change of the PIPE function is instead of collecting the data from the sensor, the data points will be collected by subscribing to a MQTT topic, as shown in figure 4.5.

```
Variables :    csv, sample_data
Output:        NULL

Procedure:
WHILE read(sample_data).hasNextItem
     mqtt.publish(current_item, data)
ENDWHILE
```

Figure 4.4 Pseudocode for Sample Data Streamification

```
cur_point = mqtt.subscribe(data)
```

Figure 4.5 Changes of PIPE Main Function for Simulation

4.6 Physical Implementation with Vibration Sensor

As per the pseudocode shown in figure 4.6, data is collected directly from the sensor for physical implementation. Figure 4.7 is showing the actual deployment with Vibration Sensor. The main board is Wemos D1 Mini (esp8266ex) (“D1 mini [WEMOS Electronics],” n.d.) and vibration sensor is accelerometer for vibration data is provided by Analog Device ADXL345 (“ADXL345 Datasheet and Product Info | Analog Devices,” n.d.). The code is baked into the main board. The communication between the board and the sensor is done via i2c. (“Specification,” n.d.)

```
Variables :error-rate threshold, error_threshold
          Original time-series array  C_o, oriseq
          The newly generated data points, cur_point

Output:   NULL

Procedure:

//one-time initialization
FOR (i = 0; i++; i<3)
     cur_point = sensor generates data points
```

```
      oriseq.add(cur_point)
ENDFOR

WHILE
      cur_point = sensor generates data points
      oriseq.add(cur_point)
      IF (error_estimation(oriseq) >
      error_threshold)
      THEN
            pip_reduce_and_forward(oriseq,
            error_threshold)
      ELSE
            continue
      ENDIF
ENDWHILE
```

Figure 4.6 Pseudocode for Physical Implementation

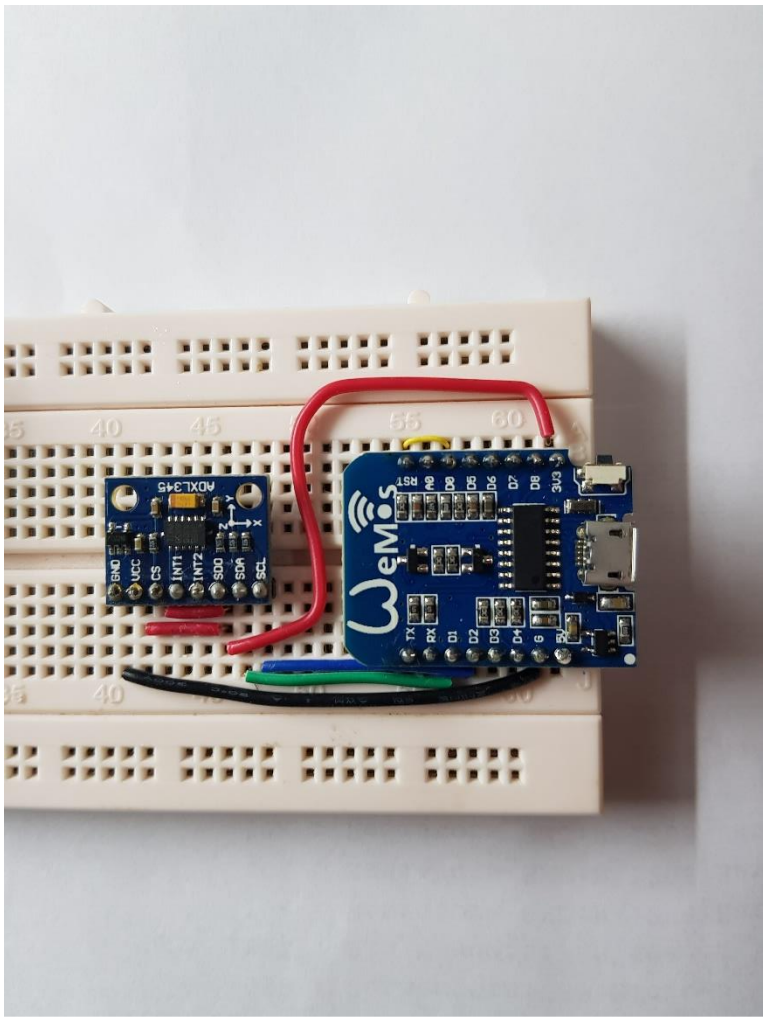

Figure 4.7 Deployment with Vibration Sensor

CHAPTER 5

Evaluation

5.1 Evaluation Objectives

There are two objectives need to be fulfilled under evaluation: -

a) To verify the performance of the proposed PIPE method, based on the design goals described in Chapter 3 Section 1

b) To compare PIPE with other different data reduction methods in relation to reduction rate and accuracy

There are in total three experiments. The overview and the purpose of these experiments are tabulated in Table 5.1 below.

No.	Experiment Setup	Purpose
1.	Utilizing PIP and other existing data reduction methods such as GZIP, Rate of Acceleration (RAC), Real-Time PIP and Original PIP to perform data reduction against 7 selected sample datasets and records the result	To examine the accuracy and reduction rate of PIPE, and compare the PIPE with other existing data reduction methods.

2.	Segmentizes each sample datasets into multiple chunks. Process each chunk with PIPE and record the results.	To examine the consistency of PIPE in term of the performance.
3.	Deploy PIPE into real sensor environment.	To validate the feasibility of deploying PIPE with physical sensor and examine the power consumption.

Table 5.1: Experiment Setups and Purposes

5.2 Key Measurements

To examine the performance of the proposed method, PIPE, and make comparisons against other data reduction techniques, two key measurements are identified and will be used in the evaluation. Firstly, the reduction rate. The reduction rate is used to quantify the degree of reduction or compression achieves by data reduction techniques. The equation of reduction rate is given as below: -

$$Reduction\ Rate = \left(1 - \frac{n\ of\ reduced\ datasets}{n\ of\ original\ datasets}\right) * 100\% \qquad (5.1)$$

Reduction rate is expressed in percentage. For instance, 80% reduction rate indicates the data is reduced by 80%, only 20% of data points remain and forwarded to the endpoint.

Secondly, accuracy indicates to which degree the reduced dataset can be used to reconstruct and restored to the original datasets. To capture the accuracy, linear interpolation between points will be used to recover or reconstruct the missing value from the reduced datasets. With the recovered datasets, accuracy will be computed using Weighted Jaccard Similarity (Chierichetti et al., 2010), in comparison with the original sequences. Weighted Jaccard similarity is the invert of Weighted Jaccard Distance (Chierichetti et al., 2010) . The equation of Weighted Jaccard Similarity is given as follows: -

$$J(C_{ori}, C_{rc}) = \frac{\sum_{i=1}^{n} \min(C_{ori}, C_{rc})}{\sum_{i=1}^{n} \max(C_{ori}, C_{rc})} \quad (5.2)$$

Where C_{ori} is the original datasets, and C_{rc} is the restored dataset. Weighted Jaccard Similarity reading is ranging from 0 to 1. Value 0 represents that the two datasets are completely different while the value 1 represents both datasets are the same.

5.3 Sample Datasets

In total, there are seven datasets used for the first and second evaluation. All datasets can be obtained from (Lichman, 2013), except datasets 7th. The 7th dataset is actual readings obtained from a vibration sensor deployed in a limestone factory. These datasets are selected based on its unique statistical characteristics and time-series pattern. With this, the consistency of the data reduction performance can then be tested.

Each of the datasets plots is shown in figure 5.1 to 5.7 below.

a. Vibration Sensor time series exhibits a trend that is similar to white noises.

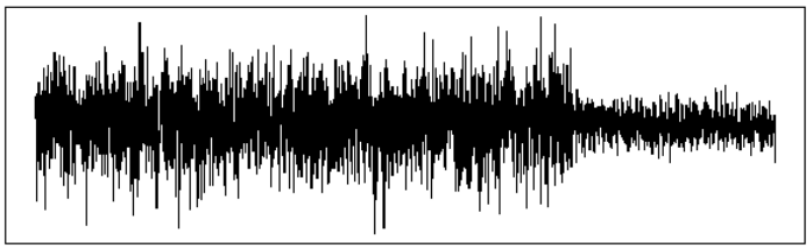

Figure 5.1: Vibration Sensor Time Series Plot

b. Luminosity Sensor. Binary time-series event, zero when the light is off and luminosity reading is recorded when the light is on

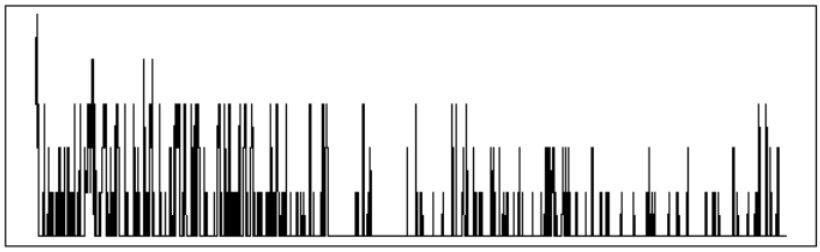

Figure 5.2: Luminosity Sensor Time Series Plot

c. Smart Power Meter Sensor. The time-series starts off with high-frequency event and followed by low frequency reading with spikes occasionally

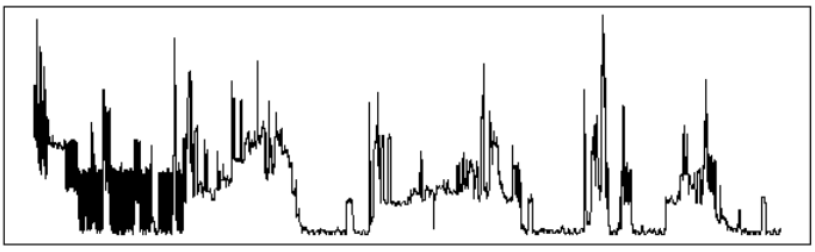

Figure 5.3: Smart Power Meter Sensor Time Series Plot

d. Temperature Sensor shows a slow-moving trend.

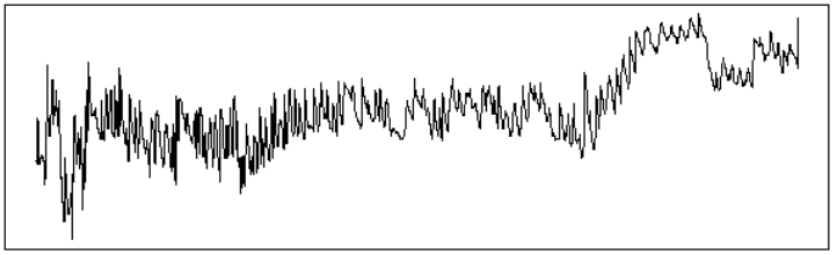

Figure 5.4: Temperature Sensor Time Series Plot

e. Dodger Loop Sensor contains a high frequency of events.

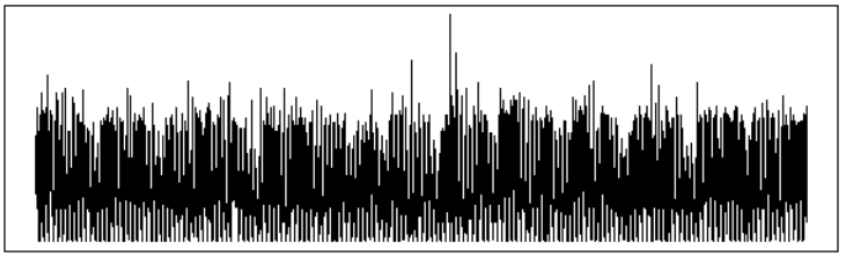

Figure 5.5: Dodger Loop Sensor Time Series Plot

f. Wind Sensor time series is showing a cyclical pattern

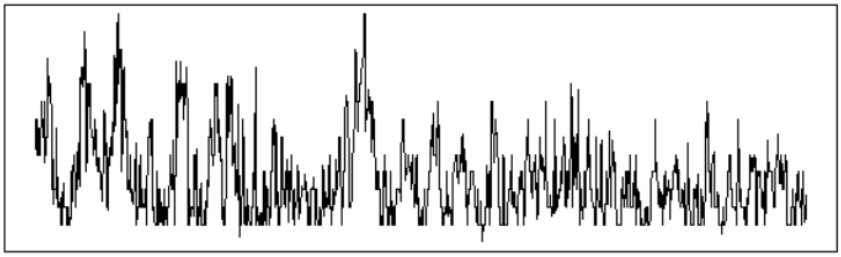

Figure 5.6: Wind Sensor Time Series Plot

g. ECG time series exhibits a recurrent pattern.

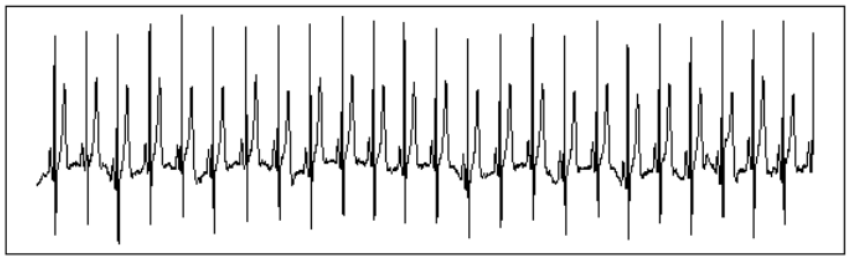

Figure 5.7: ECG Time Series Plot

5.4 Experiment on Comparison between PIPE and Existing Data Reduction Techniques

The objective of this experiment is to examine the performance of the PIPE in comparison to existing data reduction techniques. The existing data reduction techniques are GZIP (P. Deutsch, 1996), Rate of Acceleration (Toni et al., 2013) and Real-Time PIP (Papageorgiou et al., 2015a). Finally, PIPE will be compared against the original PIP proposed in (Chung et al., 2001). PIPE will be implemented based on the specification explained in Chapter 4, and the threshold is set to 0.05, 0.1 and 0.15. To generating streaming time series the sample data using MQTT, implementation based on section 4.4. Therefore, regardless which data reduction techniques, the data collection will be done via subscription to MQTT channel.

For other data reduction techniques, details on implementation will be explained in the following sections.

5.4.1 GZIP Compression

GZIP compression is implemented based on online batch-processing model. For instance, GZIP will perform compression at the interval of every 30 seconds. Figure 5.8 below illustrates the GZIP processing. After the processing is completed, the size of compressed datasets will be obtained and the reduction rate is the ratio of the size of compressed datasets over the size of the original datasets.

For this evaluation, the compression interval is set to 30, 50 and 100 seconds.

```
Variables:interval of compression, int
          Array to store data points, ori_seq
          The newly generated data points, cur_point
          The compressed dataset,
          compressed_dataset
Output:   NULL

Procedure:
cur_point = mqtt.subscribe(data)
for (i < 0; i < int; i++)
     ori_seq.add(cur_point)
ENDFOR
compressed_dataset = gzip(ori_seq)
Forward compressed_dataset
```

Figure 5.8: Pseudocode for GZIP Implementation

5.4.2 Rate of Acceleration (RAC)

Toni et al.(2013) have devised an data filtering algorithm based on rate of acceleration. The algorithm is depicted as the equation below: -

$$C_{forward} = \sqrt{C_i - 2C_{i-1} + C_{tj}} \geq th_{accel} \quad (5.3)$$

Where C_i is the current data points, C_{tj} is the previous forwarded point. th_{accel} is the threshold to determine whether the points need to be forwarded. The authors have not provided the details on setting the threshold. Therefore, in this evaluation, the th_{accel} is calculated based on the average of RAC of the entire datasets. The pseudocode of the RAC data reduction is shown in the table below.

```
Variables:Threshold, th
          Previous points, prev_point
          Previous forwarded point, _prev_fw_point
          The newly generated data points, cur_point
          Rate of Accereleration, RAC
Output:   NULL

Procedure:
cur_point = mqtt.subscribe(data)
rac = square_root(cur_point - 2*prev_point +
      prev_fw_points)
IF rac > th THEN

     Forward cur_point to endpoint
     prev_fw_point = cur_point
ENIF
prev_point = cur_point
```

Figure 5.9: Pseudocode for RAC Implementation

For every data points, RAC will be computed and compared against the configured threshold. IF the RAC reading has exceeded the threshold, the current point will then be forwarded to the endpoint.

5.4.3 Real-Time PIP

Real-time PIP is proposed by Papageorgiou et al. (2015). The proposed algorithm employs caching to execute PIP at real time on every data points. Each of the processed points will be ordered based on its level of importance. If the processed data point falls under top *X*% of the ordered list of importance, it will be forwarded to the endpoint. Users are required to configure the *X* parameter and authors did not explain about any recommended or specific method to configure the parameter. In this evaluation, the importance threshold is set to 0.85, 0.9 and 0.95.

The implementation is done based on the pseudocode provided by author at best effort basis. The pseudocode provided by author included in Appendix A.

5.4.4 The Original PIP

PIP is first proposed by Chung et al. (2001). As discussed in Chapter 2, the two main problems of this PIP version are that the processing is inherently offline and the authors did not mention how to control the reduction rate. However, since PIPE aims to optimize the original PIP, therefore, it is important to understand the performance difference between the optimized version, PIPE, and PIP.

In this evaluation, the error of reconstructed datasets that is reduced by PIPE, will be used as the error rate threshold of PIP. Figure 5.10 below illustrates the pseudocode of the implementation.

```
Variables:The entire sample dataset, data
          The set of important points C_r, imp_seq
          The line that is connecting the set of
          important points C_rc,
          connect_seq
          The Euclidean distance, eudist
          The largest Euclidean distance, maxeudist
          The Jaccard Distance that reprsentating
          error,error
          The position of the data point within the
          sequence, selection
Output:   The set of important points C_r, imp_seq

Procedure:
     data = read_all_data(sample_datasets)_
     imp_seq.add(data[0])
     imp_seq.add(data(data.size()))
     connect_seq= interpolation(imp_seq())

     WHILE (error < threshold && imp_seq.size() <
     data.size())
          eudistance = 0
          maxeudistance = 0
          selection = 0

          FOR (i = 0; i++; i < data.size())

               eudist=EuclideanDistCalculate
               (data[i] – connect_seq[i])
               IF eudist > maxeudist THEN
                    maxeudist = eudist
                    selection = i
               ENDIF
          ENDFOR
          imp_seq.add(data(selection))
          connect_seq= interpolation(imp_seq())
          error = jaccard_distance (connect_seq,
          data)
          IF error < threshold THEN
               break
          ENDIF
     ENDWHILE

     return imp_seq
```

Figure 5.10: Pseudocode for Original PIP Implementation

5.4.5 Experiment Results

The result of each of the datasets discussed individually. The result is tabulated in a quadrant graph, for ease of comparison, the label of each result can be read as {data reduction method | threshold configuration}. Finally, all results will be combined and discussed.

5.4.5.1 Vibration Sensor

Table 5.2 and Figure 5.11 below shows the result of each of the data reduction techniques in respect to accuracy and reduction rate. As GZIP is a lossless data compression, the accuracy is always attaining at 1. However, the reduction rate shows that GZIP yields better performance when the interval is huge. That means GZIP is not suitable for real-time or online processing since the GZIP only perform better with larger data size.

Looking at real-time PIP, although the reduction rate is better than RAC and PIPE, the accuracy result is not on a satisfactory level, which is lower than 0.7. In this work, we believe accuracy is more important than reduction rate to ensure information loss is minimized. RAC achieves similar accuracy and reduction with PIPE, which is around 30%, whereas PIPE result shows PIPE is capable to retain the accuracy based on the error rate threshold, despite the low reduction rate. The reduction rate is decreasing when the error rate threshold is decreasing. This could be due to more points need to be attained to minimize information loss since vibration sensor contains more events than other sample datasets.

			Vibration Sensor	
Method	Configuration		Reduction Rate	Accuracy
GZIP	Interval size	30	87.68%	1
		50	92.39%	1
		100	95.73%	1
RAC	Threshold-based		35.92%	0.954
Real-time PIP	Importance threshold	0.85	78.42%	0.681
		0.9	85.02%	0.630
		0.95	91.96%	0.567
PIPE	Error rate threshold	0.15	41.34%	0.939
		0.1	30.63%	0.964
		0.05	19.75%	0.984

Table 5.2: Vibration Sensor Evaluation Result

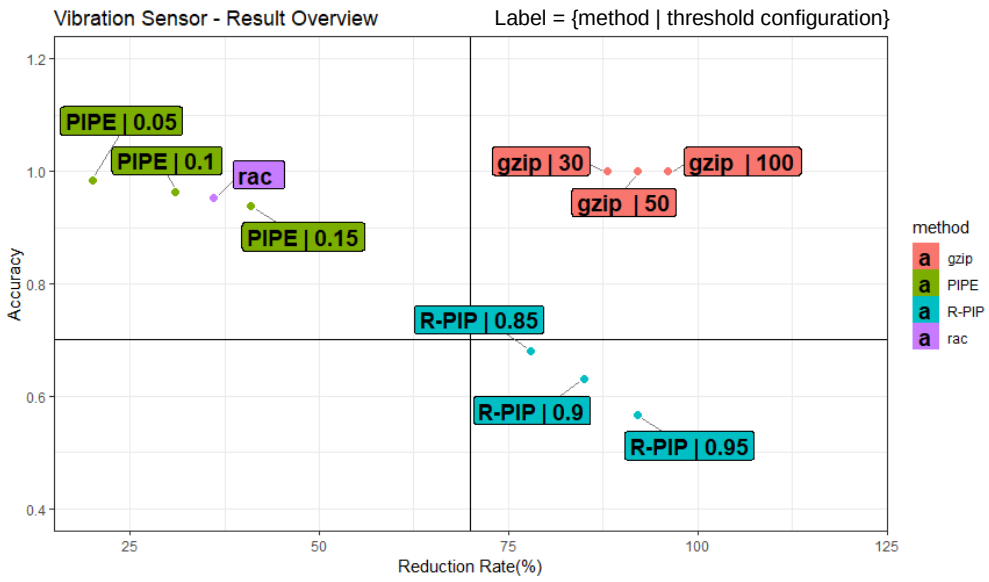

Figure 5.11: Quadrant Chart of Vibration Sensor Result

5.4.5.2 Luminosity Sensor

Similarly, Table 5.3 and Figure 5.12 shows the result for luminosity Sensors. Real-time PIP achieves good reduction rate but not accuracy. The result of PIPE and RAC is very close in term of accuracy, except PIPE has a slightly better reduction rate. Again, PIPE is capable to maintain the accuracy

based on error rate threshold configuration. The reduction rate is better than the vibration sensor.

			Luminosity Sensor	
Method	Configuration		Reduction Rate	Accuracy
GZIP	Interval size	30	85.75%	1
		50	90.82%	1
		100	95.12%	1
RAC	Threshold-based		65.89%	1
Real-time PIP	Importance threshold	0.85	75.45%	0.428
		0.9	81.79%	0.384
		0.95	89.51%	0.296
PIPE	Error rate threshold	0.15	70.03%	0.965
		0.1	68.52%	0.979
		0.05	67.45%	0.998

Table 5.3: Luminosity Sensor Evaluation Result

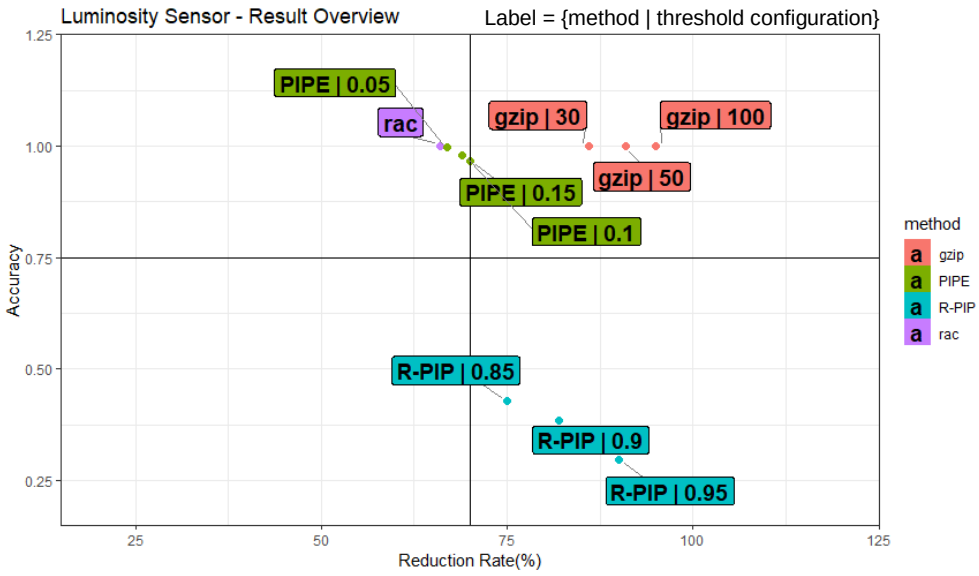

Figure 5.12: Quadrant Chart of Luminosity Sensor Result

5.4.5.3 Smart Power Meter Sensor

Table 5.4 and figure 5.13 are showing the result for smart power meter sensors, Real-Time PIP accuracy results are better than the previous datasets, though, in term of reduction rate, PIPE can achieve higher reduction rate with

better accuracy. RAC maintain its consistency in term producing similar results with PIPE.

			Smart Power Meter Sensor	
Method	Configuration		Reduction Rate	Accuracy
GZIP	Interval size	30	87.49%	1
		50	92.27%	1
		100	95.69%	1
RAC	Threshold-based		65.89%	0.969
Real-time PIP	Importance threshold	0.85	44.95%	0.880
		0.9	56.46%	0.838
		0.95	69.08%	0.771
PIPE	Error rate threshold	0.15	77.26%	0.931
		0.1	70.75%	0.962
		0.05	60.87%	0.981

Table 5.4: Smart Power Meter Sensor Evaluation Result

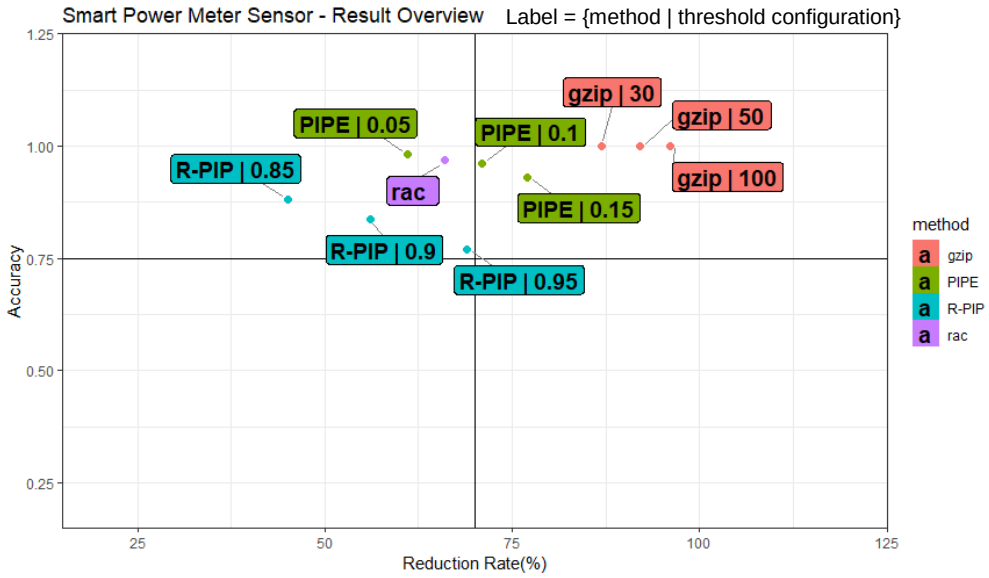

Figure 5.13: Quadrant Chart of Smart Power Meter Sensor Result

5.4.5.4 Temperature Sensor

Looking at table 5.5 and figure 4.14, it is obvious that all data reduction techniques are capable to achieve satisfactory accuracy reading. PIPE can achieve reduction rate that is better than GZIP at a slight deterioration of

accuracy. Note that PIPE has only result for 0.05 error rate configuration because for this temperature datasets, PIPE cannot produce accuracy reading that is lesser than 0.095.

			Temperature Sensor	
Method	Configuration		Reduction Rate	Accuracy
GZIP	Interval size	30	88.37%	1
		50	92.65%	1
		100	95.63%	1
RAC	Threshold-based		40.29%	0.999
Real-time PIP	Importance threshold	0.85	28.28%	0.997
		0.9	45.78%	0.991
		0.95	73.02%	0.981
PIPE	Error rate threshold	0.15	98.04%	0.966
		0.1	98.04%	0.966
		0.05	98.04%	0.966

Table 5.5: Temperature Sensor Evaluation Result

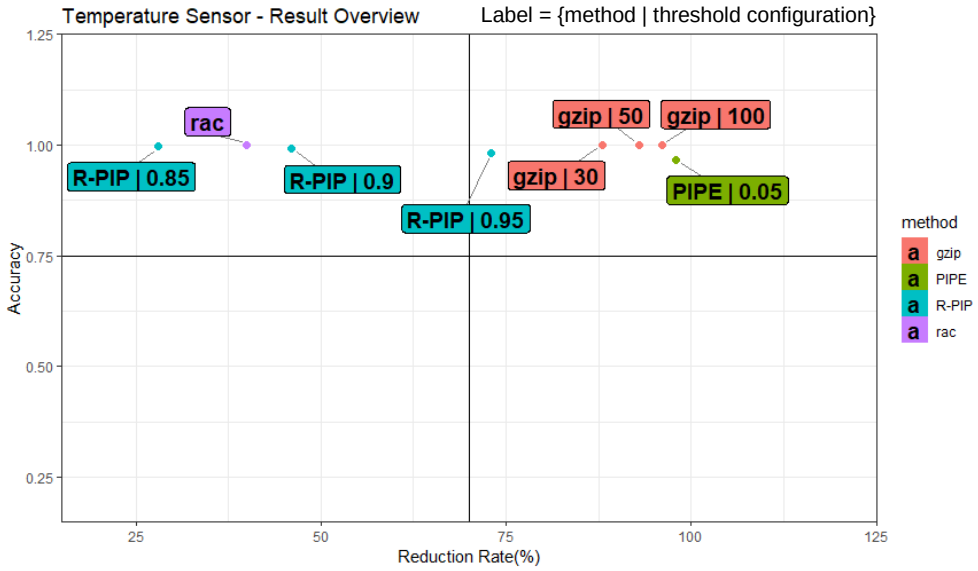

Figure 5.14: Quadrant Chart of Temperature Sensor Result

5.4.5.5 Dodger Loop Sensor

Referring to table 5.6 and figure 5.15, the result shows similar behaviours with vibration sensor. PIPE has a poor reduction rate, together with

RAC. Dodger loop sensor dataset exhibits a high frequency of events. Therefore, this has further strengthened the hypothesis made in section 5.4.4.1, low reduction rate is due to more points is required to minimize information loss for datasets that contains more events. On the other hand, Real-Time PIP produces a satisfactory result as well and exhibits the same behaviours with RAC and PIPE, which is a low reduction rate.

			Dodger Loop Sensor	
Method	Configuration		Reduction Rate	Accuracy
GZIP	Interval size	30	86.34%	1
		50	91.27%	1
		100	95.28%	1
RAC	Threshold-based		32.03%	0.970
Real-time PIP	Importance threshold	0.85	47.17%	0.860
		0.9	57.09%	0.830
		0.95	69.19%	0.795
PIPE	Error rate threshold	0.15	51.63%	0.919
		0.1	34.66%	0.960
		0.05	20.12%	0.982

Table 5.6: Dodger Loop Sensor Evaluation Result

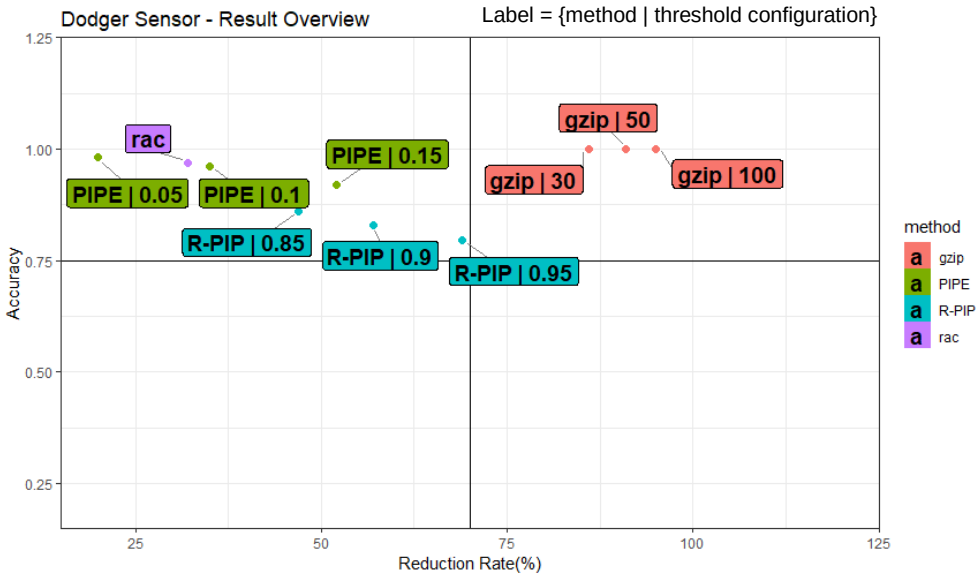

Figure 5.15: Quadrant Chart of Dodger Loop Sensor Result

5.4.5.6 Wind Sensor

The result for wind sensor is shown with Table 5.7 and Figure 5.16 below. Results are similar across all the data reduction technique except GZIP. When the accuracy reading is lower, the reduction rate is higher. PIPE attains the accuracy reading based on the error rate threshold.

			Wind Sensor	
Method	Configuration		Reduction Rate	Accuracy
GZIP	Interval size	30	88.70%	1
		50	92.98%	1
		100	96.00%	1
RAC	Threshold-based		38.46%	0.985
Real-time PIP	Importance threshold	0.85	44.86%	0.921
		0.9	57.40%	0.88
		0.95	77.55%	0.756
PIPE	Error rate threshold	0.15	70.74%	0.933
		0.1	60.93%	0.959
		0.05	45.29%	0.986

Table 5.7: Wind Sensor Evaluation Result

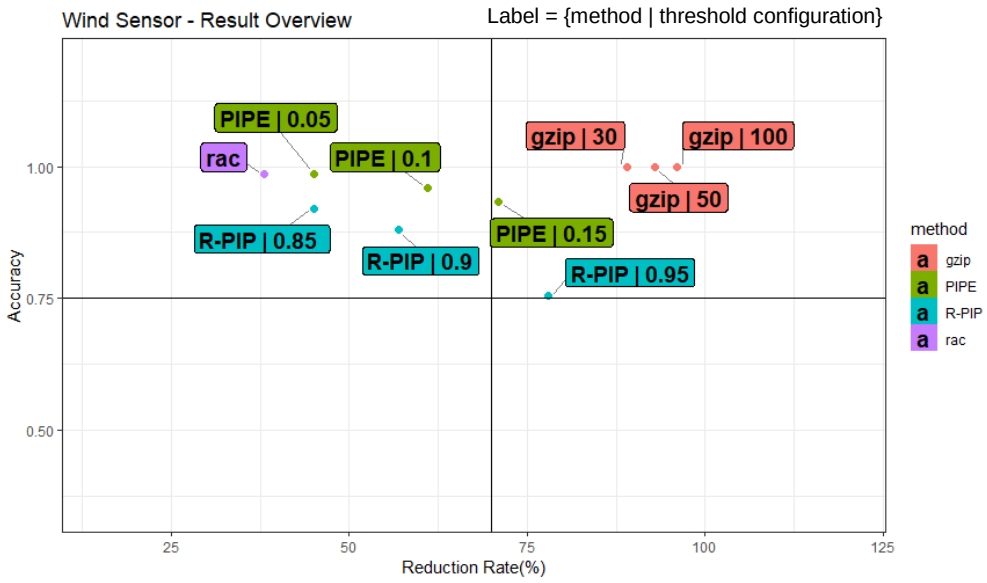

Figure 5.16: Quadrant Chart of Wind Sensor Result

5.4.5.7 ECG

Referring to Table 5.8 and Figure 5.17, all data reduction techniques achieve near to 1 accuracy. Real-time PIP achieves near 1 accuracy with lower reduction compare to RAC and PIPE. With marginally lesser accuracy than 1, PIPE achieves good reduction rate, which is almost close to 100.

			ECG	
Method	Configuration		Reduction Rate	Accuracy
GZIP	Interval size	30	88.05%	1
		50	92.68%	1
		100	95.92%	1
RAC	Threshold-based		62.58%	0.995
Real-time PIP	Importance threshold	0.85	18.10%	0.999
		0.9	32.76%	0.992
		0.95	50.50%	0.975
PIPE	Error rate threshold	0.15	96.32%	0.946
		0.1	95.42%	0.958
		0.05	90.85%	0.980

Table 5.8: ECG Evaluation Result

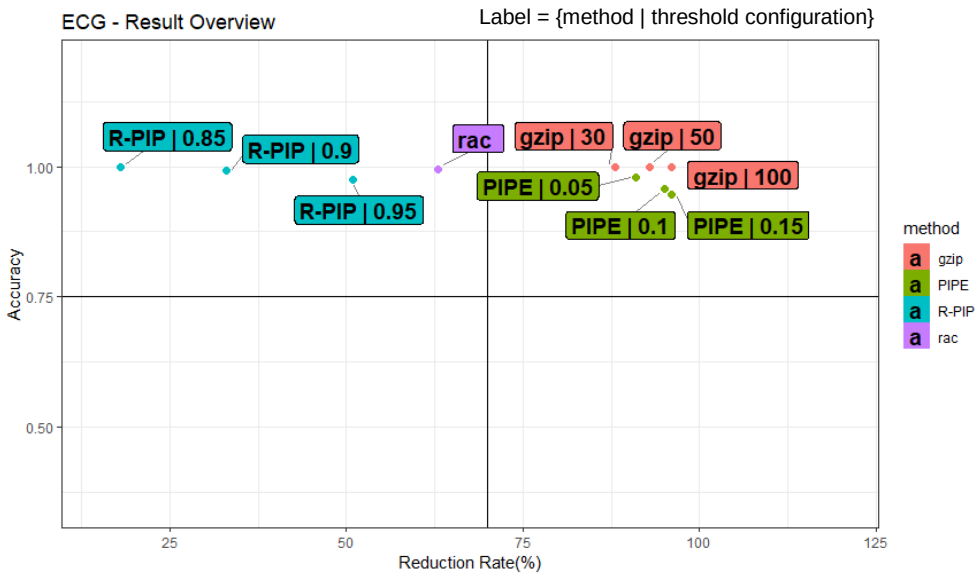

Figure 5.17: Quadrant Chart of ECG Result

5.4.5.8 Summary

Figure 5.18 illustrates the summary of the results of the 7 sample datasets. As GZIP compression result is consistent and similar across all datasets, GZIP compression is not shown to reduce the complexity of diagram.

Real-Time PIP results are not consistent across the datasets. Some datasets have low accuracy reading, which is not favourable as it indicates information loss.

RAC attains high accuracy reading but its reduction rates vary between 20% to 70%. This can be explained by some datasets contain more events, therefore more points are preserved to minimize information loss. Although result is satisfactory for RAC, the threshold setting for RAC is not adaptive and requires offline analysis to compute the optimum threshold for each dataset.

The experiment results have proven that PIPE has achieved the design goal set in chapter 3. With an error rate threshold setting, PIPE attains the accuracy level for all different sample datasets despite their statistical differences. In another word, the error rate threshold is adaptive to time-series data reduction usage, without the need for offline analysis. Besides, the error rate threshold ensures accuracy is prioritized over the reduction rate. For instance, sample data like dodger loop sensor and vibration sensor, PIPE attains the accuracy by preserving more points. For time series that has lesser events like temperature sensor, PIPE achieves high reduction rate like 98% with an accuracy of 0.97.

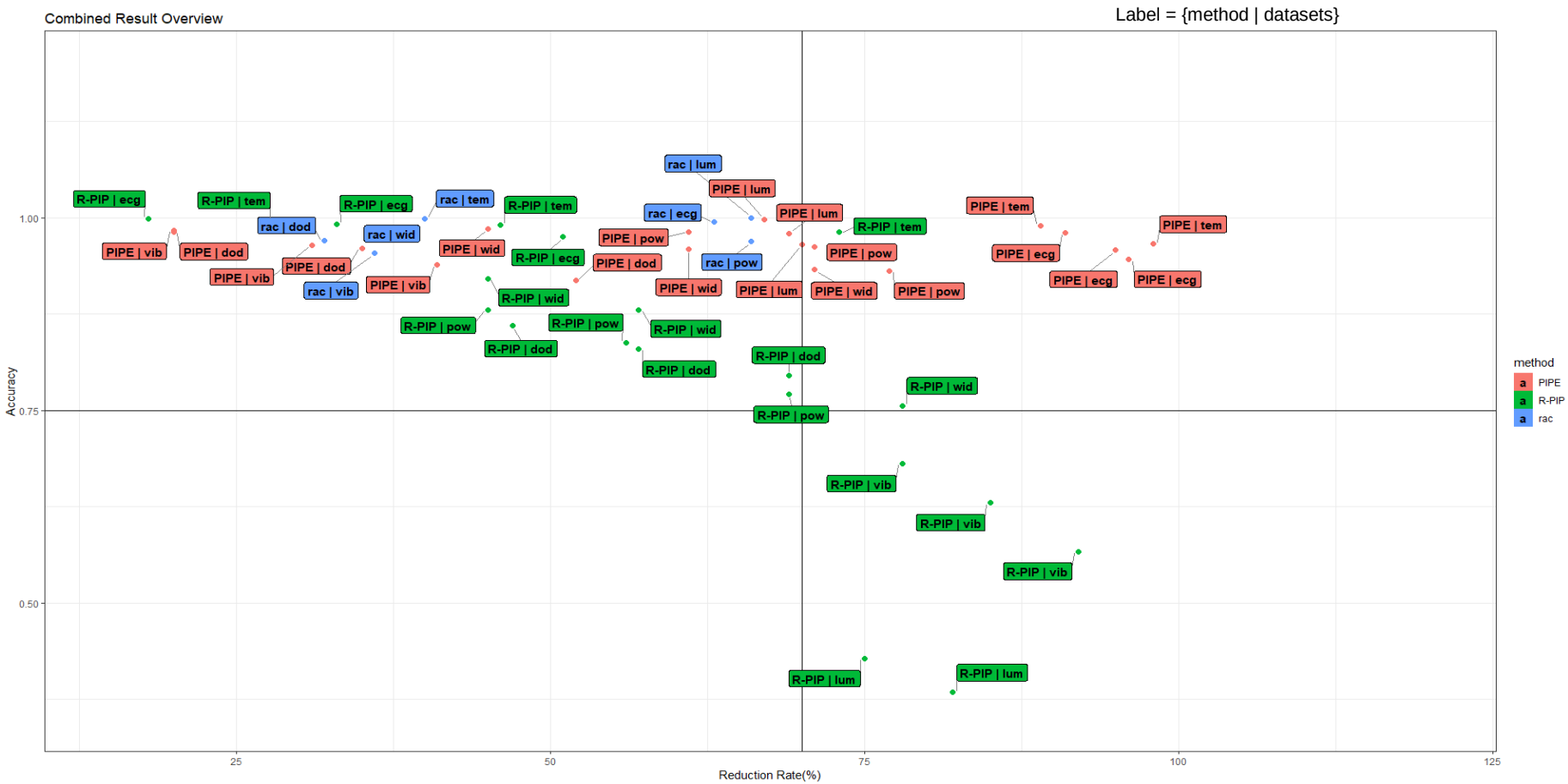
Combined Result Overview
Label = {method | datasets}
Accuracy
1.00
0.75
0.50
25
50
75
100
125
Reduction Rate(%)
method
PIPE
R-PIP
rac
R-PIP | ecg
R-PIP | tem
R-PIP | ecg
rac | tem
R-PIP | tem
rac | lum
PIPE | lum
rac | ecg
PIPE | lum
PIPE | tem
PIPE | tem
rac | dod
rac | wid
PIPE | pow
R-PIP | tem
PIPE | tem
PIPE | vib
PIPE | dod
PIPE | wid
R-PIP | ecg
PIPE | dod
rac | pow
PIPE | pow
PIPE | ecg
PIPE | tem
PIPE | dod
PIPE | vib
PIPE | wid
PIPE | lum
PIPE | wid
PIPE | pow
PIPE | ecg
PIPE | ecg
rac | vib
PIPE | vib
R-PIP | wid
R-PIP | pow
R-PIP | pow
R-PIP | wid
R-PIP | dod
R-PIP | dod
R-PIP | dod
R-PIP | wid
R-PIP | pow
R-PIP | vib
R-PIP | vib
R-PIP | vib
R-PIP | lum
R-PIP | lum

Figure 5.18: Quadrant Chart of the Combined Results

5.4.5.9 Comparison between PIPE and Original PIP

PIPE has employed and optimized PIP to devise a novel data reduction technique that is aligned to the research objectives and design goal. It is essential to understand the performance of PIPE compared PIP. The evaluation is done based on implementation method in 5.4.4, with error rate of PIP is set to the error of reconstructed dataset that is reduced by PIPE. Figure 5.16 (a) – (g) shows the result of PIPE vs PIP. The same colour code indicates the accuracy result of both PIP and PIPE is the same.

		PIPE		Original PIP	
Datasets	Error Rate (only for PIPE)	Reduction Rate	Accuracy	Reduction Rate	Accuracy
Vibration	0.05	41.34%	0.939	55.04%	0.939
	0.1	30.63%	0.964	44.52%	0.964
	0.15	19.75%	0.984	31.58%	0.984
Luminosity	0.05	70.03%	0.965	75.93%	0.965
	0.1	68.52%	0.979	74.98%	0.979
	0.15	67.45%	0.998	73.64%	0.998
Smart Meter	0.05	77.26%	0.931	84.70%	0.931
	0.1	70.75%	0.962	78.42%	0.962
	0.15	60.87%	0.981	71.02%	0.981
Temperature	0.05	98.04%	0.966	99.15%	0.966
Dodger Loop	0.05	51.63%	0.919	68.67%	0.919
	0.1	34.66%	0.960	50.80%	0.96
	0.15	20.12%	0.982	35.07%	0.982
Wind	0.05	70.74%	0.933	81.99%	0.933
	0.1	60.93%	0.959	74.98%	0.959
	0.15	45.29%	0.986	62.49%	0.986
ECG	0.05	96.32%	0.946	97.17%	0.946
	0.1	95.42%	0.958	97.05%	0.958
	0.15	90.85%	0.980	94.00%	0.98

Table 5.9: Results of PIPE vs Original PIP

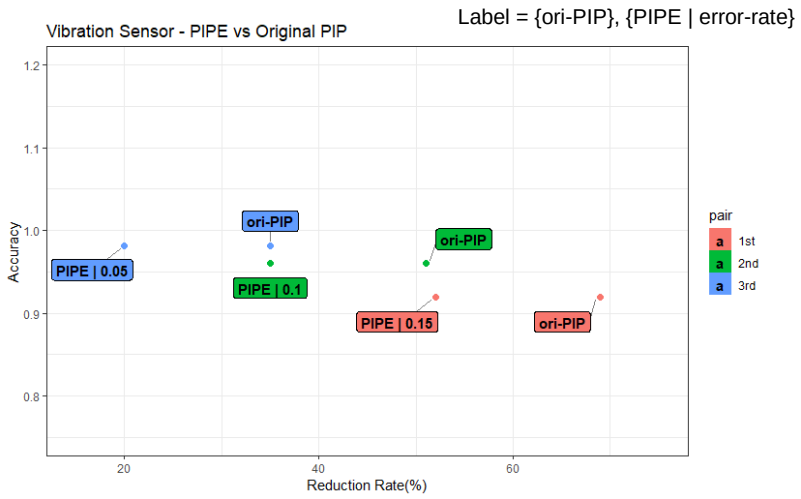

Figure 5.19 (a): Quadrant Chart of Vibration Sensor PIPE vs PIP Results

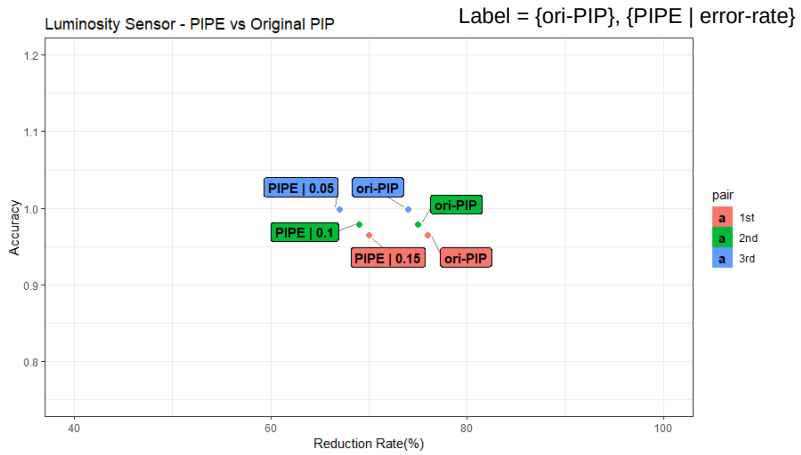

Figure 5.19 (b): Quadrant Chart of Luminosity Sensor PIPE vs PIP Results

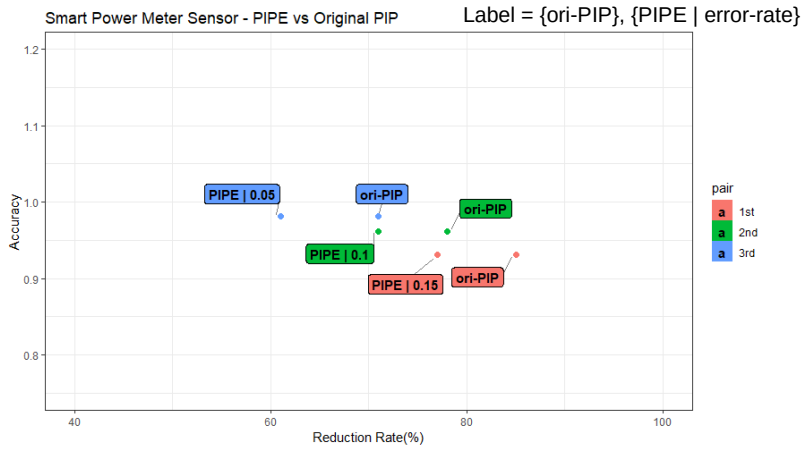

Figure 5.19 (c): Quadrant Chart of Smart Power Sensor PIPE vs PIP Results

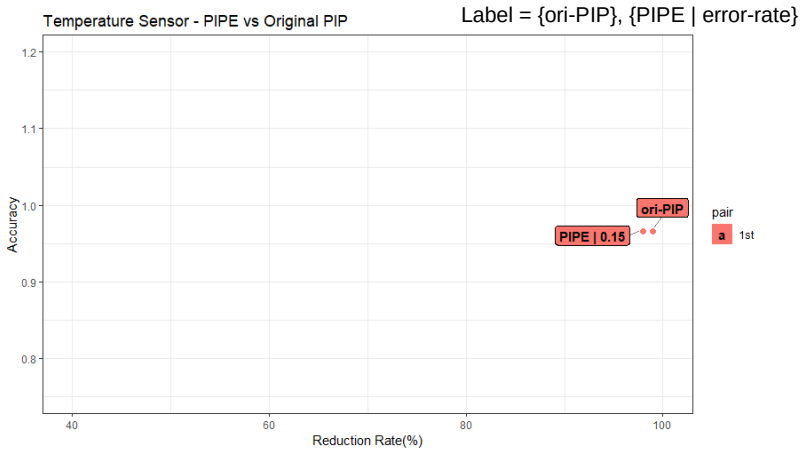

Figure 5.19 (d): Quadrant Chart of Temperature Sensor PIPE vs PIP Results

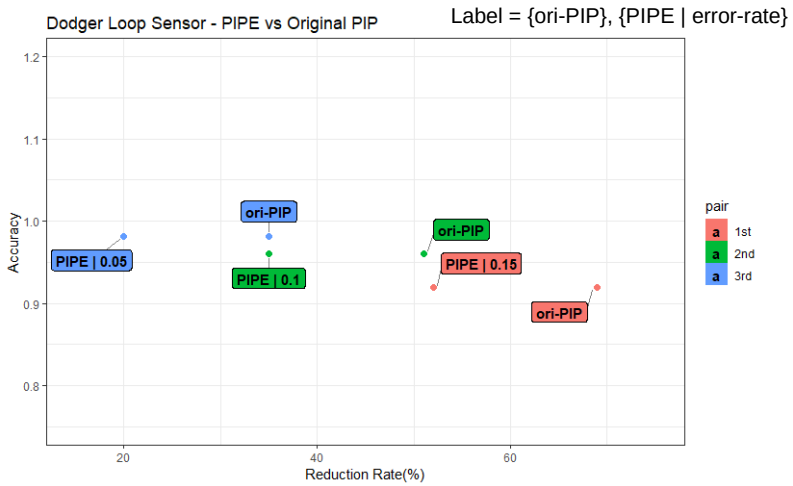

Figure 5.19 (e): Quadrant Chart of Dodger Loop Sensor PIPE vs PIP Results

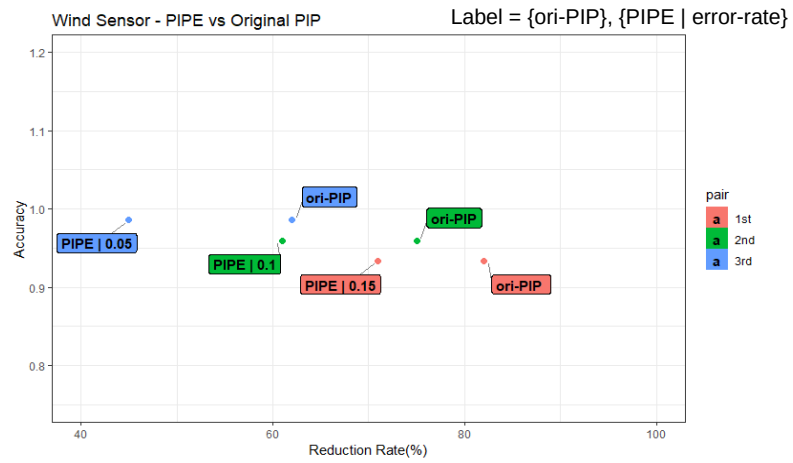

Figure 5.19 (f): Quadrant Chart of Wind Sensor PIPE vs PIP Results

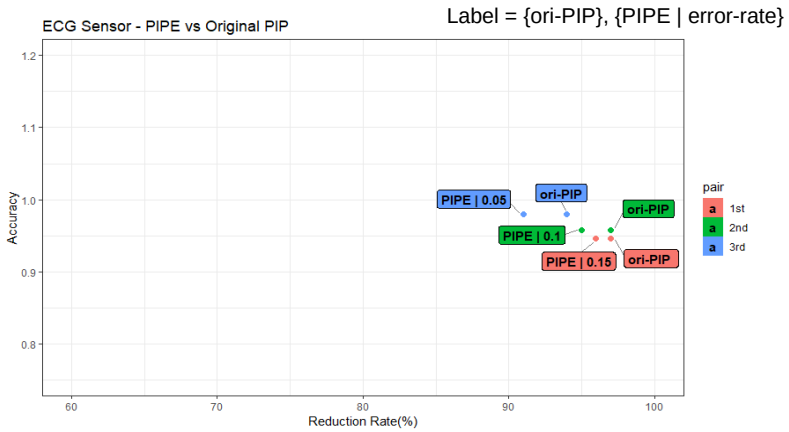

Figure 5.19 (g): Quadrant Chart of ECG PIPE vs PIP Results

Figure 5.19 (a) – (g) shows the original PIP has better performance in term of reduction rate compare to PIPE. For instance, around 20% of difference is observed for Dodger Loop Sensor and Vibration Sensor datasets. In contrary, for datasets like ECG and Temperatures, the different in term of reduction rate is minimal. The could be due to the error estimation prone to high frequency of events, therefore, more PIP iteration is triggered For Dodger Loop Sensor and Vibration Sensors.

As compared to PIPE, the original PIP has the full knowledge of every data points in the sample datasets. Therefore, it is expected that the original PIP can extract important points more effective then PIPE, which results in better reduction rate. In contrary, though PIPE has no full knowledge over the entire datasets, it is still capable to produce satisfactory and consistent data reduction result at real time processing.

5.5 Experiment on PIPE Performance Consistency with Segmented Datasets

Each of the sample dataset consists of 3000 data points. It is important to note for the entire datasets, different events or frequency of event occurrences can be different in random sub-sections of the datasets.

The result of experiment 5.4 has shown that when event frequency is higher, PIPE produces lower reduction rate. To examine whether such variants happens within the same datasets and impacting the performance of PIPE in term of both reduction rate and accuracy, each sample datasets is divided into sub-section with size of 100. PIPE is executed on each sub-section, with error-rate threshold 0.05. The result is aggregated with the table below.

	Reduction Rate (%)		Accuracy	
Data Set	Average	Standard Deviation	Average	Standard Deviation
Vibration	19.8	0.053	0.98	0.0025
Luminosity	63.5	0.181	0.99	0.0025
Smart Meter	60.1	0.196	0.98	0.0059
Temperature	95.5	0.031	0.97	0.0094
Dodger	19.9	0.047	0.98	0.0041
Wind	44.6	0.077	0.99	0.0046
ECG	89.2	0.022	0.98	0.0046

Table 5.10: Results of PIPE on Segmented Datasets

Referring to the table 5.10, the standard deviation of reduction is varying between different datasets. For example, low standard deviation is observed for sample datasets like Temperature and ECG, whereas high standard deviation

reading for Luminosity and Smart Meter Sensors. Whereas, the accuracy reading is maintained at 0.95 and above with low standard deviation reading, reflecting the error rate threshold successfully control the error generated of the entire datasets despite of different segments, and producing consistent results in term of accuracy.

The result has further strengthened the claims of PIPE prioritizes correctness over reduction rate, as well as error-rate threshold can be used to control the error of the reduced datasets effectively.

5.6 Experiment on Physical Deployment

Chapter 4 has mentioned the PIPE is implemented based on C++ Arduino framework. In this section, physical implementation and evaluation will be carried out to measure the power consumption, to examine whether it is feasible to implement PIP at sensor node level. In fact, this work has more ambitious goal, which is to reduce the power consumption by reducing the number of communication.

PIPE is deployed with WeMos D1 ESP8266 WiFi Board ("D1 mini [WEMOS Electronics]," n.d.) with ADXL345 accelerometer ("ADXL345 Datasheet and Product Info | Analog Devices," n.d.). ESP8266 is one of the most widely used microcontrollers for IoT application due to the inexpensive cost. (Abdel-Basset et al., 2018).

However, high power consumption is one of the biggest challenges to implement ESP8266 as compared to other Bluetooh-based microcontrollers because the WiFi communication module consumes substantial amount of power. (Skraba et al., 2016). By default, PIPE does not send data regularly, only when error estimation exceeds error rate threshold. Therefore, incorporate with ESP8266 implementation, the WiFi module is switched off at all time unless data forwarding is required. An experiment is conducted to examine the possibility of power saving with such deployment.

In term of the deployment details, the sensor is recording the vibration axis at the frequency of every second. With a default implementation, the sensor is sending data to an MQTT broker at every second as soon as the data is collected; In contrary, PIPE implementation will send data on-demand basis. The key measurement of this experiment is power consumption.

To examine the energy efficiency of PIPE, the power consumption over a period of 60 sec is recorded by oscilloscope for two scenarios: -

a) Default implementation. There is no data reduction processing. Sensing and data forwarding to MQTT broker is done every second. WiFi connection is maintained all the time.

b) PIPE implementation, with error rate threshold set to 0.1. WiFi connection will only initiated to facilitate on-demand PIPE data forwarding. Otherwise, it will be turned off.

The key measurement is power consumption and total energy consumption. Power consumption refers to electrical energy per unit time, it displayed as a graph at unit time of 5ms with Figure 5.20. The total energy consumption will be calculated as the area under the graph. Figure 5.20 has shown the power consumption for default implementation. The power consumption is maintained at minimum 200mW per unit over 60 secs. The total energy consumption is 3.0kJ.

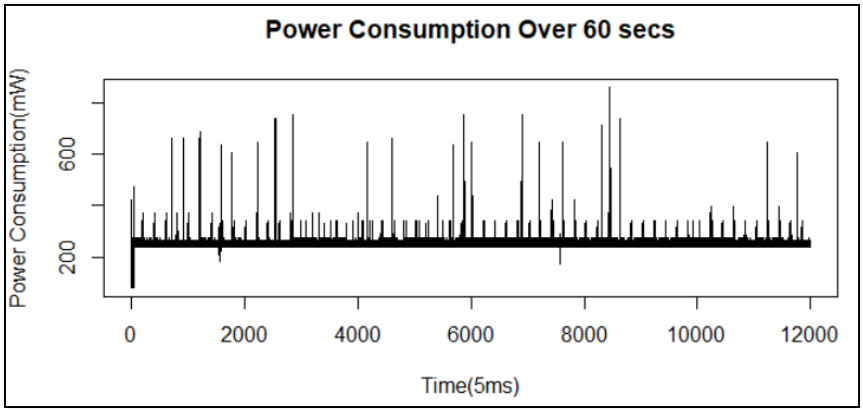

Figure 5.20: Power Consumption Over 60 Sec for based implementation

The power consumption of PIPE implementation over 60 secs is shown in Figure 5.21. When there is no data forwarding, the power consumption is negligible compares to Figure 5.20, even though there is PIPE processing for each data points. There are several on-demand data forwarding is triggered by PIPE along the 60 secs. During this period, the power consumption is increased and maintained at more than 20mW. Once forwarding is completed, the power consumption is again dropped to minimal. The total energy consumption for PIPE implementation over 60 secs is 1.56kJ.

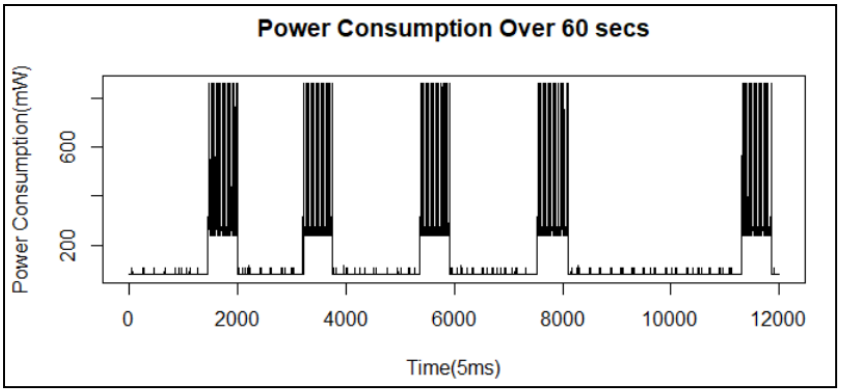

Figure 5.21: Power Consumption Over 60 Sec for PIPE implementation

In summary, the experiments can be concluded with the following points: -

a) PIPE can be implemented at sensor node level.

b) By switching on the WiFi module only for on-demand data forwarding, PIPE helps to save up to 50% of energy consumption of ESP8266.

CHAPTER 6

Conclusion and Future Work

6.1 Revisiting the Objectives

The objective of this research work is to devise a novel data reduction approach to can perform real-time processing with no offline analysis for configuration, and can be deployed and used with heterogeneous environment and time-series data. In chapter 2, state-of-the-art is discussed and concluded that, although there are existing works have solved problem like real-time processing, those works come with constraint like use-case specific. For instance, some works focus on processing biometrics data while others focus on IoT gateway implementation. Hence, the motivation to devise a novel data reduction technique that achieves all defined objectives at the same time is formed in this work.

We opined that accuracy is more crucial then reduction rate for a data reduction technique. Therefore, the first step taken in designing the data reduction technique was introducing error rate threshold. Error rate is used to control the accuracy of the reduced datasets. The same error rate can be used for any time series data despite the difference in term of statistical or pattern.

Secondly, to enable read-time processing, error estimation is introduced. Error estimation process every data points at real time. When the estimated error is exceeding the error rate threshold, the optimized PIP will be triggered, extracts a subset of important points and send to endpoint at real time.

To ensure the objectives and designed goal are realized, PIPE is evaluated with three experiments. First, PIPE performance is examined in comparison with other data reduction techniques. The results have indicated the PIPE exhibits consistent performance among other data reduction in term of accuracy across all datasets, proven error rate threshold can be used for heterogeneous datasets. Second experiment aims to test the performance consistency on segmented datasets. The results reveal PIPE consistently prioritizes accuracy over performance based on error-rate threshold. For the last experiment, PIPE is deployed with ESP8266 microcontroller, benefit from the computation logic, PIPE is capable to achieve up to approximate 50% of energy saving depending on the amount of reducible data. Therefore, it is feasible to deploy PIPE at sensor node. As conclusion, the objectives of this work are achieved.

6.2 Future Work

PIPE requires unsent data points to be stored in the memory before it get processed. PIPE can be further optimized so that there will be no need of storing data points at memory.

Besides, error rate threshold can be further improved to be dynamically configured based on the statistical characteristic of time-series, without any human intervention. Addition to that, the algorithm can further be improved to produce reduced data that have the error rate equivalent to error rate threshold. Currently the reduced data error rate is always higher than error-threshold, based on the evaluation results.

REFERENCES

Abdel-Basset, M., Manogaran, G., Mohamed, M., 2018. Internet of Things (IoT) and its impact on supply chain: A framework for building smart, secure and efficient systems. Future Generation Computer Systems. https://doi.org/10.1016/j.future.2018.04.051

ADXL345 Datasheet and Product Info | Analog Devices [WWW Document], n.d. URL http://www.analog.com/en/products/adxl345.html (accessed 9.24.18).

Agrawal, R., Faloutsos, C., Swami, A., 1993. Efficient similarity search in sequence databases. Springer.

Alduais, N.A.M., Abdullah, J., Jamil, A., Audah, L., 2016. An efficient data collection and dissemination for IOT based WSN, in: Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016 IEEE 7th Annual. IEEE, pp. 1–6.

Amarlingam, M., Mishra, P.K., Prasad, K.D. and Rajalakshmi, P., 2016, December. Compressed sensing for different sensors: A real scenario for WSN and IoT. In Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on (pp. 289-294). IEEE.

Aström, K.J., 1969. On the choice of sampling rates in parametric identification of time series. Information Sciences 1, 273–278.

Barbon, G., Margolis, M., Palumbo, F., Raimondi, F., Weldin, N., 2016. Taking Arduino to the Internet of Things: The ASIP programming model. Computer Communications 89–90, 128–140.

Blalock, D., Madden, S. and Guttag, J., 2018. Sprintz: Time Series Compression for the Internet of Things. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2(3), p.93.

Cadzow, J.A., Baseghi, B., Hsu, T., 1983. Singular-value decomposition approach to time series modelling, in: IEE Proceedings F (Communications, Radar and Signal Processing). IET, pp. 202–210.

Chan, K.-P., Fu, A.W.-C., 1999. Efficient time series matching by wavelets, in: Data Engineering, 1999. Proceedings., 15th International Conference On. IEEE, pp. 126–133.

Chen, G., Yang, H. and Huang, L., 2012. Optimizing compressive sensing in the internet of things. In Future Wireless Networks and Information Systems (pp. 253-262). Springer, Berlin, Heidelberg.

Chierichetti, F., Kumar, R., Pandey, S. and Vassilvitskii, S., 2010, January. Finding the jaccard median. In Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms (pp. 293-311). Society for Industrial and Applied Mathematics.

Chung, F.-L., Fu, T.-C., Luk, R., Ng, V., 2001. Flexible time series pattern matching based on perceptually important points. International joint conference on artificial intelligence workshop on learning from temporal and spatial data 1–7.

D1 mini [WEMOS Electronics] [WWW Document], n.d. URL https://wiki.wemos.cc/products:d1:d1_mini (accessed 9.24.18).

Deutsch, P. and Gailly, J.L., 1996. Zlib compressed data format specification version 3.3 (No. RFC 1950).

Dubey, H., Yang, J., Constant, N., Amiri, A.M., Yang, Q., Makodiya, K., 2015. Fog Data: Enhancing Telehealth Big Data Through Fog Computing, in: Proceedings of the ASE BigData & SocialInformatics 2015. ACM, Kaohsiung, Taiwan, pp. 1–6.

Facebook, n.d. Zstandard - Real-time data compression algorithm [WWW Document]. URL https://facebook.github.io/zstd/ (accessed 8.31.18).

Fathy, Y., Barnaghi, P. and Tafazolli, R., 2018, May. An Adaptive Method for Data Reduction in the Internet of Things. In Proceedings of IEEE 4th World Forum on Internet of Things. IEEE.

Feng, L., Kortoçi, P. and Liu, Y., 2017, October. A multi-tier data reduction mechanism for IoT sensors. In Proceedings of the Seventh International Conference on the Internet of Things (p. 6). ACM.

Fu, T., 2011. A review on time series data mining. Engineering Applications of Artificial Intelligence 24, 164–181. https://doi.org/10.1016/j.engappai.2010.09.007

Fu, T.C., 2011. A review on time series data mining. Engineering Applications of Artificial Intelligence, 24(1), pp.164-181.

Fu, T.C., Hung, Y.K. and Chung, F.L., 2017, November. Improvement algorithms of perceptually important point identification for time series data mining. In Soft Computing & Machine Intelligence (ISCMI), 2017 IEEE 4th International Conference on (pp. 11-15). IEEE.

GitHub, 2018. LZ4 Extremely Fast Compression algorithm. lz4.

Jaccard, P., 1901. Distribution de la Flore Alpine dans le Bassin des Dranses et dans quelques régions voisines. Bulletin de la Societe Vaudoise des Sciences Naturelles 37, 241–72.

Keogh, Eamonn, Chakrabarti, K., Pazzani, M., Mehrotra, S., 2001. Locally adaptive dimensionality reduction for indexing large time series databases. ACM SIGMOD Record 30, 151–162.

Keogh, E. and Ratanamahatana, C.A., 2005. Exact indexing of dynamic time warping. Knowledge and information systems, 7(3), pp.358-386.

Keogh, E., Chu, S., Hart, D. and Pazzani, M., 2001. An online algorithm for segmenting time series. In Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on (pp. 289-296). IEEE.

Keogh, E.J., Pazzani, M.J., 2000. Scaling up dynamic time warping for datamining applications, in: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 285–289.

Kravets, I., n.d. PlatformIO: An open source ecosystem for IoT development [WWW Document]. PlatformIO. URL https://platformio.org (accessed 9.9.18).

Li, S., Xu, L.D., Wang, X., 2013. Compressed Sensing Signal and Data Acquisition in Wireless Sensor Networks and Internet of Things 9.

Lichman, M., 2013. UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences.

Maschi, L.F., Pinto, A.S., Meneguette, R.I. and Baldassin, A., 2018. Data Summarization in the Node by Parameters (DSNP): Local Data Fusion in an IoT Environment. Sensors, 18(3), p.799.

Mohamed, M.F., Shabayek, A.E.R., El-Gayyar, M. and Nassar, H., 2018. An adaptive framework for real-time data reduction in AMI. Journal of King Saud University-Computer and Information Sciences.

P. Deutsch, 1996. GZIP file format specification version 4.3 [WWW Document]. URL https://tools.ietf.org/html/rfc1952 (accessed 6.24.18).

Papageorgiou, A., Cheng, B., Kovacs, E., 2015a. Real-time data reduction at the network edge of Internet-of-Things systems, in: Network and Service Management (CNSM), 2015 11th International Conference On. IEEE, pp. 284–291.

Papageorgiou, A., Cheng, B., Kovacs, E., 2015b. Reconstructability-Aware Filtering and Forwarding of Time Series Data in Internet-of-Things Architectures. IEEE, pp. 576–583.

Peng, J., Wang, H., Li, J. and Gao, H., 2016, June. Set-based similarity search for time series. In Proceedings of the 2016 International Conference on Management of Data (pp. 2039-2052). ACM.

Phetking, C., Sap, M.N.M., Selamat, A., 2009. Identifying Zigzag based Perceptually Important Points for indexing financial time series, in: 2009 8th IEEE International Conference on Cognitive Informatics. pp. 295–301.

Rani, M., Dhok, S.B., Deshmukh, R.B., 2018. A Systematic Review of Compressive Sensing: Concepts, Implementations and Applications. IEEE Access 6, 4875–4894.

Salomon, D., 2004. Data compression. Springer, New York.

Sarker, H., Tyburski, M., Rahman, M.M., Hovsepian, K., Sharmin, M., Epstein, D.H., Preston, K.L., Furr-Holden, C.D., Milam, A., Nahum-Shani, I. and Al'Absi, M., 2016, May. Finding significant stress episodes in a discontinuous time series of rapidly varying mobile sensor data. In Proceedings of the 2016 CHI conference on human factors in computing systems (pp. 4489-4501). ACM.

M.M., Hovsepian, K., Sharmin, M., Epstein, D.H., Preston, K.L., Furr-Holden, C.D., Milam, A., 2016. Finding Significant Stress Episodes in a Discontinuous Time Series of Rapidly Varying Mobile Sensor Data. ACM Press, pp. 4489–4501.

Schoellhammer, T., Greenstein, B., Osterweil, E., Wimbrow, M., Estrin, D., 2004. Lightweight Temporal Compression of Microclimate Datasets 9.

Shcherbakov, M.V., Brebels, A., Shcherbakova, N.L., Tyukov, A.P., Janovsky, T.A., Kamaev, V.A., 2013. A survey of forecast error measures. World Applied Sciences Journal 24, 171–176.

Skraba, A., Kolozvari, A., Kofjac, D., Stojanovic, R., Stanovov, V., Semenkin, E., 2016. Streaming pulse data to the cloud with bluetooth LE or NODEMCU ESP8266. IEEE, pp. 428–431.

Specification [WWW Document], n.d. . I2C Bus. URL https://www.i2c-bus.org/specification/ (accessed 9.24.18).

Toni, Goh, H.G., Liew, S.-Y., 2013. Energy Saving Data Abstraction and Reformation Algorithms for Human Movement Monitoring. IEEE, pp. 599–604.

Toni, Goh, H.G., Liew, S.Y., 2012. Performance study of zeroth-, first-and second-order data abstraction and reformation algorithms for wireless sensor networks, in: Wireless Communications and Applications (ICWCA 2012), IET International Conference On. IET, pp. 1–6.

Tsinaslanidis, P.E., Kugiumtzis, D., 2014. A prediction scheme using perceptually important points and dynamic time warping. Expert Systems with Applications 41, 6848–6860.

Weisstein, E.W., n.d. Two-Point Form [WWW Document]. URL http://mathworld.wolfram.com/Two-PointForm.html (accessed 7.2.18).

Yi, B.-K., Faloutsos, C., 2000. Fast time sequence indexing for arbitrary Lp norms. VLDB.

Zaib, G., Ahmed, U., Ali, A., 2004. Pattern recognition through perceptually important points in financial time series 12.

Zhang, J., Yu, Z.L., Gu, Z., Li, Y. and Lin, Z., 2018. Multichannel Electrocardiogram Reconstruction in Wireless Body Sensor Networks Through Weighted ℓ1, 2 Minimization. IEEE Transactions on Instrumentation and Measurement.

APPENDIX A

The pseudocode of the Real-Time PIP implemented by Papageorgiou et al., (2015a)

```
DEFINITIONS

tsName:         The time series name, provided before calling handleData
cache:          The cache, initialized upon instantiation of the data handler
projStrategy    The cache projection strategy (CLONE, TWIN, or AVG)
threshold:      Configuration parameter (between 0.0 and 1.0),
                which implicitly enforces the filtering ratio
-------------------------------------------------------------
// Inputs: value (a value of the time series that has just been received)
//         timestamp (the timestamp for this value, captured by the GW application)
// Output: - (none)

currentItem.name = tsName;
currentItem.value = value;
currentItem.timestamp = timestamp;

cache.add(currItem);

// Create cache projection (NOTE: Maintaining a copy of the cache can be optimized.
       This is skipped here for easier understanding)

for each cacheItem in cache) {
   cacheProjection.add(cacheItem);
         sum = sum + cacheItem.getValue;
}
currItemCopy = currItem;
if (projStrategy == CLONE) {
         cacheProjection.add(currItemCopy);
} else if (projStrategy == TWIN) {
   for (i = 1 : cache.size) {
      cacheProjection.add(cache[i]);
   }
} else if (projStrategy == AVG) {
   currItemCopy.setValue(sum/cache.size);
   cacheProjection.add(currItemCopy);
}

order = [] // list that will store the indices of the cached items in descending
       order of item importance

// Now the first and the last item of cacheProjection can be included as PIPs
order.add(0);
order.add(cacheProjection.size-1);

// iterate as many times as the (rest of the) items that need to be ordered
for (int j=1; j<cacheProjection.size-1; j++) {
   selection = j; // the item currently selected to become the next PIP
   // iterate over all items (ignoring the already selected ones) in order to find
         the one with the maximum distance to its adjacent PIPs
   for (int i=1; i<cacheProjection.size-1; i++) {
      if (order.contains(i)) continue; // item has already been selected as a PIP
            previously. Ignore it.
      tmpItem = cacheProjection[i];
      // get the line to which the distance of the currently examined item should be
            measured, i.e., the line that connects the two closest PIPs between
            which the currently examined item is located
      currItemRelevantLine = getLineOfAdjacentPIPs(tmpItem); // details of this
            subroutine are omitted, functionality is obvious
      distance = currItemRelevantLine.getDistance(tmpItem); // details of this
            subroutine are omitted, functionality is obvious
      if (distance > maxDistance) {
         maxDistance = distance;
         selection = i;
```

```
        }
    }
    // add the item with the maximum distance to its adjacent PIPs to the already
           selected ones, thus making it the next PIP
    order.add(selection);
}

currItemNormalizedImportance = 1.0 - itemPositionInOrder/order.size;

if (currItemNormalizedImportance > threshold) {
    sendToCloud(currItem);
}

if (cache.size == cache.maxSize) {
    cache.remove(0);
}

return;
```