Technical University of Crete

School of Electrical and Computer Engineering

Reversible Data Summaries and their Effect on Mining Sensor Data Streams

Diploma Thesis

Anthony Skevis

Thesis Committee

Asst. Professor Nikos Giatrakos(Supervisor)
Professor Antonios Deligiannakis
Assoc. Professor Vasilis Samoladas

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Αντιστρέψιμες Συνόψεις Δεδομένων και η Επίδρασή τους στην Εξόρυξη Γνώσης από Δεδομένα Αισθητήρων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σκεύης Αντώνης

Εξεταστική Επιτροπή

Επ. Καθηγητής Γιατράκος Νίκος(επιβλέπων)
Καθηγητής Δεληγιαννάκης Αντώνιος
Αναπλ. Καθηγητής Σαμολαδάς Βασίλειος

ACKNOWLEDGEMENTS

I want to express my sincere thanks to Prof. Nikos Giatrakos for his valuable advice, introduction to the topic, and guidance throughout this study. His support has played a crucial role in my understanding of the subject, and I am grateful for the opportunity to work under his supervision.

I would also like to convey my deepest appreciation to my family and friends, whose steadfast support and inspiration have been a constant source of strength over the years.

ABSTRACT

This thesis investigates the effect of reversible data summary methods in the context of mining sensor data streams. The study explores four well-established methods, namely Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), and Piecewise Aggregate Approximation (PAA), alongside a novel Reversible Random Hyperplane Projection method developed within the scope of this thesis.

As part of this research, we apply the aforementioned compression techniques to two datasets that contain sensor measurements. We employ tumbling windows of varying sizes and leverage an array of data mining techniques. These include diverse clustering methods such as K-Means and DBSCAN, regression algorithms like Linear and Logistic Regression, and various classification approaches such as K-NN, SVM, and a Neural Network. We apply these techniques to both raw and compressed datasets, ensuring a comprehensive analysis of the data. The study aims to assess how well each compression method retains the accuracy of results compared to the original, raw, data. By directly comparing these methods, we gain insights into their effectiveness in retaining crucial information for data mining purposes within the compressed representations.

Additionally, the study explores real-world network effects of the involved compression techniques by simulating a sensor network using TOSSIM. This simulation evaluates how data compression affects the number of bits needed to transmit a dataset and the resulting power savings. By gaining insights from these results, the research contributes to understanding how compression techniques could increase the lifetime of sensor networks, an important factor for sustainable and efficient deployments of resources, not only in sensor, but also in broader Internet of Things (IoT) settings.

This detailed analysis has two goals: firstly, to compare how different data compression methods perform, and secondly, to provide practical insights for using them in real-world sensor networks.

CONTENTS

CHAPTER 1: INTRODUCTION

1.1 THESIS MOTIVATION

In today's modern landscape, sensor networks play a crucial role in the collection and transmission of data. The efficiency of these networks is intricately tied to the amount of data they handle. As data volumes rise, so do the energy demands of these networks, ultimately shortening their operational lifespan as interruptions in connectivity between sensor nodes become more frequent when the residual energy in their battery is depleted.

Our objective is to extend the lifespan of these networks by minimizing the energy needed for data transmission while ensuring the preservation of valuable information when performing a variety of data mining tasks over the data collected by sensors. To achieve this goal, we study the impact and effectiveness of four prominent reversible data compression techniques found in the literature and an additional method introduced in this thesis.

1.2 METHODOLOGY

In order to evaluate and compare the compression techniques, we took sensor datasets frequently used in the related work and divided them into data windows of 16, 32, 64, and 128 data points. Subsequently, we applied reversible compression techniques to each of these individual windows and we used the reverse data summary to perform a variety of standard data mining tasks. This research aims to:

- Assess the effectiveness of established compression techniques, including Discrete Fourier Transform (DFT)[1], Discrete Wavelet Transform (DWT)[2], Discrete Cosine Transform (DCT)[3], and Piecewise Aggregate Approximation (PAA)[4] in the context of mining sensor data. The focus is on evaluating the data mining task accuracy versus network bandwidth consumption in the following way: (i) compressed sensor measurements are communicated across the sensor network, (ii) the reversibility of the compressed data is exploited at a base station to

re-generate approximations of the original measurements. Subsequently, these approximations are used to perform a variety of data mining tasks.

- Introduce a novel data compression technique that takes advantage of Random Hyperplane Projections to achieve data size reduction while preserving information integrity with quality guarantees.

- Examine the impact of these compression methods on data mining tasks, over a variety of algorithms per task, including:

    - **Clustering algorithms:** K-means[5], DBscan[6]

    - **Regression algorithms:** Linear[7] and Logistic[8] regression algorithms

    - **Classification algorithms:** K-nearest neighbors[9], Support Vector Machine[10], and a Neural Network[11].

- Investigate the implications of these compression techniques through evaluation using various metrics, specific to each task and algorithm.

- Implement and analyze the real-world applicability of these compression techniques in a simulated wireless sensor network using TOSSIM[12], focusing on data transfer and energy consumption in a controlled environment.

1.3 CHAPTER OUTLINE

This thesis is structured as follows:

- **Chapter 2: Related Work on Reversible Data Compression Methods**
  In this chapter, we delve into established compression techniques, specifically examining the Discrete Fourier Transform (DFT)[1], Discrete Cosine Transform (DCT)[3], Discrete Wavelet Transform (DWT)[2], and Piecewise Aggregate Approximation (PAA)[4]. Our emphasis lies in comprehending their practical applications

and assessing their efficacy in managing sensor network data. We will explore the mathematical foundations and real-world uses of each technique, laying the groundwork for subsequent discussions on compression methodologies.

- **Chapter 3: Our Proposed Reversible Data Compression Method**
  This chapter introduces our sensor data compression technique based on Random Hyperplane Projections[13]. We outline the fundamental principles behind this method, emphasising its role in reducing storage and transmission requirements while preserving the relative similarity of sensor data. The chapter covers both the compression phase and the subsequent decompression phase. This chapter offers an outline of our method, presenting a practical solution for handling sensor data in real-world scenarios.

- **Chapter 4: TOSSIM Showcase**
  The fourth chapter presents the TOSSIM simulation environment, its uses, and how we used it to evaluate the performance of compressed and uncompressed data in a realistically simulated wireless sensor network.

- **Chapter 5: Datasets, Algorithms, and Evaluation Metrics**
  Here we provide a detailed exploration of the datasets, machine learning algorithms, and evaluation metrics used in our experiments. We discuss two datasets, one provided by IntelLabData[14] and the other comprised of measurements from 51 water pump sensors[15], along with the associated data preprocessing steps. The chapter introduces the machine learning algorithms we used, such as K-means, DBSCAN, Linear Regression, Logistic Regression, KNN, SVM, and Neural Networks, each tailored to specific research objectives. We explain the application of these algorithms and the process of selecting their parameters.

  We conclude with a discussion of evaluation metrics, including Silhouette Score, Adjusted Rand Index, R-squared, Root Mean

Squared Error, Accuracy, and F1 Score, adding a 95% confidence interval to expose the effectiveness of the respective techniques beyond their mere average behavior. This chapter sets the stage for presenting and interpreting experimental results.

- **Chapter 6: Performance Comparison and Results**
  This chapter reports on our experiments, the results obtained, and the implications of our findings regarding energy efficiency, data accuracy, and the performance of the compression algorithms.

By investigating these aspects, this research contributes to the broader understanding of energy-efficient data transmission in sensor networks, providing insights into the optimization of sensor networks for diverse applications.

CHAPTER 2: RELATED WORK ON REVERSIBLE DATA COMPRESSION METHODS

This chapter delves into four significant techniques that are available in the literature: the Discrete Fourier Transform (DFT)[1], the Discrete Wavelet Transform (DWT)[2], the Discrete Cosine Transform (DCT)[3], and the Piecewise Aggregate Approximation (PAA)[4]. While seemingly different in their approaches, these techniques share a common objective: the compression with the ability of decompression and manipulation of (mainly time series) data to not only extract valuable information via mining but also apply them for ultimately serving a multitude of applications.

Our goal in this chapter is to explore the aforementioned techniques, laying the foundation for our later experimental comparisons. Through an examination of DFT, DWT, DCT, and PAA, we will unravel the inner workings of each technique, highlight their respective strengths, and showcase their practical applications.

2.1 THE DISCRETE FOURIER TRANSFORM (DFT)

2.1.1 Introduction to DFT

The Discrete Fourier Transform (DFT) is a mathematical operation used to change a set of equally spaced samples from a function into another set of same-length and equally spaced samples, representing the discrete-time Fourier transform. The inverse DFT (IDFT) can then be used to recover the original data from these frequency components. It is most often used in signal processing and data analysis applications. It has recently seen extensive use as an image compression technique[16] where it is used in a similar way as our own applications.

The application of DFT is most often performed by making use of the fast Fourier transform (FFT)[17], which we also use in our experiments. Mathematically DFT is expressed as:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi kn/N}$$, k = 1,2,...,N [7]

Where:

- x_n represents the initial input sequence.
- X_k represents the frequency components of the input sequence.

- The term $e^{-j2\pi kn/N}$ follows the form of Euler's formula. It decomposes the signal into its constituent sinusoidal components. The DFT computes the coefficients of these exponentials.

- N is the number of data points in the sequence.

By using Parseval's theorem, which states that the energy of a signal x(t) in the time domain (E(x)) is equal to the energy of its Fourier transform X(f) (E(X)), we can show that the Euclidean distance between two signals in the time domain is the same as their distance in the frequency domain[18]:

$$D(x, y) = \sqrt{(E(x - y))} = \sqrt{(E(X - Y))} = D(X, Y)$$

Compression is achieved by sorting the resulted coefficients in descending order and keeping (transmitting in our setup) the most significant ones throughout the network. Because of this compression scheme, we essentially omit positive additive terms from the D(X,Y) distance calculation and, therefore, for the compressed series: $D(x,y) \geq D(X,Y)$ holds.

The reverse data summary is derived by using only these most significant coefficients and inverse DFT expressed as:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{+j2\pi kn/N}$$, k = 1,2,...,N [7]

In the inverse DFT formula depicted above, x_n, X_k and N correspond to the same variables mentioned in the DFT formula. The term $e^{+j2\pi kn/N}$ follows the form of Euler's formula, serving to reconstruct the signal.

2.1.2 DFT Applications

DFT has a multitude of applications some of which include:

Audio and Speech Processing

Facilitating spectral analysis, pitch detection, and noise reduction, DFT-based algorithms play a pivotal role in applications such as speech recognition and audio compression[19].

Image Processing

In image processing, the DFT has multiple applications, including filtering, image compression, and feature extraction. By converting images to the frequency domain it enables noise reduction, feature enhancement[20], and the utilization of a multitude of compression techniques.

Telecommunications

Within telecommunications, the DFT can be applied for modulation[21], where it transforms digital data into analog waveforms, and also, in demodulation, for the reconstruction of the original data at the receiving end. Techniques like Quadrature Amplitude Modulation (QAM) and Frequency-Shift Keying (FSK) rely on the DFT[22]. Moreover, the DFT is a pivotal tool used in channel equalization[23], where it addresses signal distortion during transmission.

Scientific Research

- *Medical Imaging*
  The DFT processes MRI and CT scan data[24], extracting and analyzing image data, contributing to diagnoses and medical research.

- *Astronomy*
  DFT aids astronomers in analyzing space signals, including radio emissions from celestial bodies, facilitating comprehension of their characteristics and properties[25].

- *Spectroscopy*
  In spectroscopy, DFT is applied to assess the spectra of different materials[26], supporting chemical analysis and material characterization.

- *Seismology*
  Seismologists rely on the DFT to assess seismic signals, decipher earthquake traits, and comprehend ground movements[27]. This understanding of seismic wave frequencies is vital for earthquake

prediction and structural engineering.

Engineering
DFT can be used as a method to analyze structural vibrations, enabling the detection of defects and the assessment of the health of infrastructure like bridges and buildings[28].

The broad spectrum of applications for the Discrete Fourier Transform (DFT) demonstrates its fundamental importance in an array of scientific, technological, and research domains. The examples discussed here merely scratch the surface of its utility. Beyond the highlighted applications, the DFT finds its place in fields such as radar technology, weather forecasting, financial analysis, and many others.

2.1.3 DFT Data Compression

To better understand DFT, we can take a closer look at its application in data compression:

- Initially, we need a dataset sampled from various sources(such as sensor data, audio recordings, or digital images) at regular intervals to create a sequence of data points.

- We partition the dataset into windows of different sizes (16, 32, 64, and 128). For instance, in a dataset with 48 rows, with each representing a sensor and containing 624 temperature measurements, selecting a window size of 16 results in dividing each row into 39 data windows, each containing 16 data points(see Fig.1).

- For each data window, we apply the DFT formula and calculate their complex coefficients, which are comprised of a magnitude(real part) and a phase(imaginary part).

- We arrange the coefficients in descending order of magnitude and retain only a portion of the larger coefficients, with the proportion determined by the desired level of compression.

- After compression, we can make use of the inverse DFT (IDFT) to recover an approximation of the original data. Zero-padding can be applied for additional frequency components in the DFT spectrum.

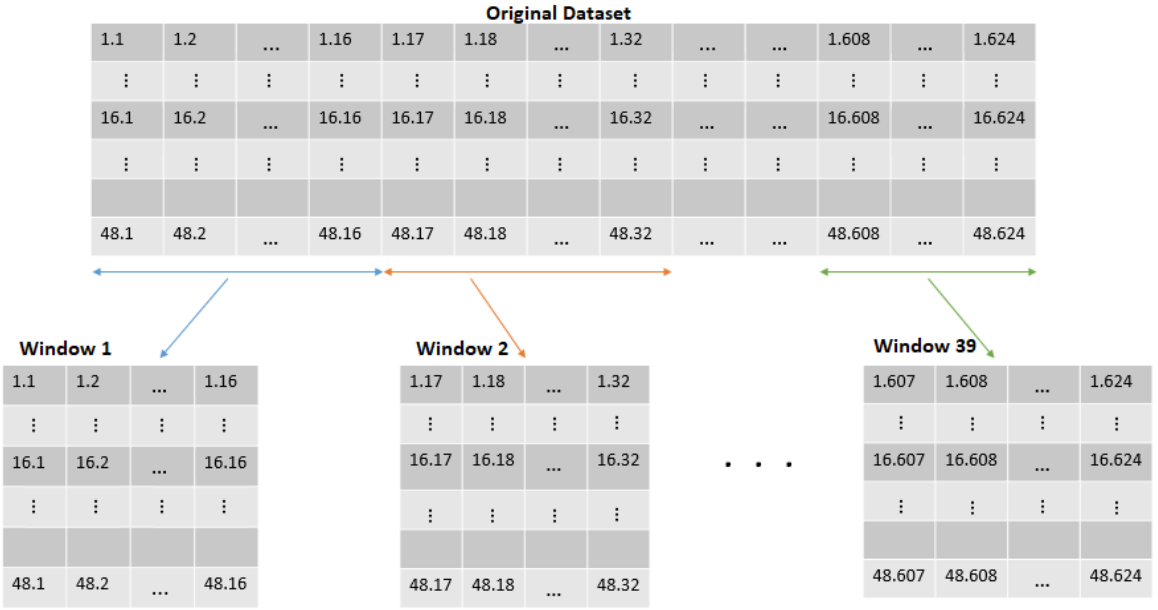

Figure 1. Example of splitting a dataset into windows of 16 data.

In the realm of data analysis and compression techniques, the DFT offers an effective way to reduce data size while preserving critical information. It can be accomplished by singling out the highest-magnitude frequencies, typically corresponding to the most significant ones[29]. This is based on the principle that, when it comes to real-world datasets, most of the important information is concentrated within a small number of dominant frequency components. DFT-based compression takes advantage of this by identifying and isolating the aforementioned dominant components from the less significant ones.

DFT compression enables data dimensionality reduction[30], simplifying data while conserving storage space, computational resources, and also, upon application on sensors, residual energy resources. The DFT's utility extends to feature extraction where, by isolating key frequency components, it is particularly beneficial for tasks like pattern recognition and anomaly detection[31]. Additionally, by distinguishing and eliminating undesired noise elements, it serves as a proficient tool for noise mitigation, ensuring the retention of essential frequency data[32].

In summary, the Discrete Fourier Transform (DFT) is a versatile mathematical tool with a wide array of applications across multiple domains. Its ability to transform the input data and represent them in the frequency domain makes it an indispensable tool for tasks ranging from data compression and feature extraction to signal processing and scientific research. The DFT's impact extends from everyday technology like audio and image compression to critical applications in fields such as healthcare, telecommunications, and environmental monitoring.

2.2 THE DISCRETE COSINE TRANSFORM (DCT)

2.2.1 Introduction to DCT

The Discrete Cosine Transform (DCT)[3] is closely related to the Discrete Fourier Transform (DFT) but operates solely with real numbers. Unlike the DFT, which deals with periodically extended sequences, the DCT is associated with Fourier series coefficients from symmetrically and periodically extended sequences[33]. DCTs can be seen as an extension of DFTs, typically involving real data, as the Fourier transform of a real function is also real.

Of the eight standard variants of the Discrete Cosine Transform (DCT), four of them are the most widely used. Among these variants, the type-II DCT, often referred to simply as the DCT, stands out as the most prevalent. Originally introduced by Nasir Ahmed[3], this DCT variant serves as an effective tool for translating data into the frequency domain by representing it as a combination of cosine functions at varying frequencies[33]. The type-III DCT, commonly known as the Inverse DCT (IDCT), assists in reconstructing the original data from these frequency components. The DCT's adaptability and utility extend to a broad spectrum of applications, including signal processing, data compression, and image and video coding.

Mathematically type-II and type-III DCT are expressed as[33]:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right], \quad k = 0,\ldots,N-1$$

[***DCT (type-II)***]

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(k + \frac{1}{2}\right)n\right], \quad k = 0,\ldots,N-1$$

[***IDCT (type-III)***]

Where:

- N is the number of data points in the sequence.
- x_o-x_{N-1} represents the real numbers to be transformed.
- X_k represents the DCT or IDCT output.

Assuming the row and column vectors that compose each discrete cosine transform kernel matrix define a set of orthogonal basis functions with C denote the (n x m) kernel matrix of a given cosine transform, the Parseval Theorem for DCT is expressed by this formula[34]:

$$\sum_{m=0}^{N-1} |X(m)|^2 = ||X||^2 = X^T X = x^T C^T C x = x^T x = ||x||^2 = \sum_{i=0}^{N-1} |x(i)|^2$$

Where $C^T = C^{-1}$ and $C^T C = I$, since the column vectors of C are orthogonal to each other, and as a result, the transpose of C is also its inverse.

In the above equation, both the left and the right sides are the sum of squares. By truncating coefficients, the sum of squares in the DCT decreases, thus the expression $\sum_{i=0}^{N-1} |x(i)|^2 \geq \sum_{m=0}^{N-1} |X(m)|^2$ is valid.

2.2.2 DCT Applications

The DCT proves its potency in the analysis and manipulation of data across one-dimensional and two-dimensional domains. Below we highlight some of its key applications.

Digital media technologies

DCT serves as the foundation for prevalent compression standards like JPEG (Joint Photographic Experts Group)[35] in the case of images, and MPEG (Moving Picture Experts Group)[36] in the case of video and audio, where it is used to shrink file sizes while upholding satisfactory quality. It efficiently

reduces memory and bandwidth requirements, achieving compression ratios from 8:1 to 100:1. Consequently, DCT standards are integral in various digital media technologies, including digital images, photos, video, streaming, and high-definition content[19].

Biometrics

By extracting distinctive features and patterns from biometric data, DCT can be used in authenticating and identifying individuals, thus finding applications in biometric fields like facial recognition and fingerprint analysis.[37].

Encryption and Steganography

By employing DCT-based data-hiding techniques, information can be subtly concealed within images and other media, rendering it imperceptible to human observation. This approach is commonly employed for watermarking and steganography, allowing the covert inclusion of secret messages or metadata within digital content[38].

Forgery Detection

DCT is often used as a tool for the analysis of coefficients linked to image frequency, spotting unusual patterns that indicate tampering. It is also often integrated into forgery detection algorithms to compare or break down images or videos.[39].

Although the scope of applications for the Discrete Cosine Transform (DCT) extends far beyond the ones we mentioned, its significance and utility become particularly evident in our constantly evolving digital landscape.

2.2.3 DCT Data Compression

The DCT compression process involves several key steps, in our application these steps include:

- In a manner similar to DFT-based compression, we commence with a discrete dataset sourced from various origins such as images, audio recordings, or, in our case, sensor data, all of which comprise of regularly sampled data points.

- We split the dataset into windows of varying sizes (16, 32, 64 and 128).
- We apply the DCT formula to each data window. Unlike the DFT, the DCT is a real-to-real transform and computes a set of coefficients that represent the signal in terms of its cosine components.
- Similar to DFT-based compression, we arrange the DCT coefficients in descending order of magnitude. To achieve compression, we retain only a portion of the larger coefficients. The proportion of the retained coefficients is determined by the desired level of compression. Higher compression ratios involve discarding more coefficients.
- The inverse DCT (IDCT) can be used to recover an approximation of the original data. Zero-padding can be applied to acquire additional spatial components in the DCT spectrum, just as in DFT-based compression.

Within the domain of data analysis and compression techniques, the Discrete Cosine Transform presents a proficient approach to reducing data size while retaining important information. Similarly to the DFT compression approach, DCT compression takes advantage of the presumption that crucial data is usually predominantly concentrated within a limited set of dominant frequencies, which are effectively isolated through the application of thresholding techniques.

Similar to its other transform-based counterparts, DCT compression offers substantial assistance in reducing data dimensionality by simplifying complex and large datasets[40]. In applications like image and video compression, this method is particularly helpful, as it facilitates the efficient storage and transmission of multimedia data without substantial compromise to quality.

In summary, the Discrete Cosine Transform (DCT) has wide-ranging applications in data analysis, compression, and signal processing. Developed by Nasir Ahmed, T. Natarajan, and K. R. Rao in the early 1970s, the DCT has proven instrumental in various fields, including digital media technologies, biometrics, encryption, and forgery detection. Its ability to efficiently compress data while maintaining essential information, thanks to its real-to-real transformation and cosine-based representation, has had a profound impact on data compression. As our digital landscape continues to evolve, the DCT

remains a versatile and essential tool in modern data processing and communication technologies.

2.3 THE DISCRETE WAVELET TRANSFORM (DWT)

2.3.1 Introduction to DWT

The Discrete Wavelet Transform (DWT) is an essential tool that emerged as a powerful alternative to traditional Fourier-based techniques. In this chapter, we will explore the history, mathematical foundations, various applications, and key variants of the DWT.

The development of the DWT involved the independent contributions of several researchers with Ingrid Daubechies, Yves Meyer, and Stéphane Mallat being some of the most prominent figures. Their work[41] revolutionized signal processing, especially image and data compression.

The DWT, like the other transform-based techniques discussed in our thesis, operates in the frequency domain. However, it distinguishes itself through the use of wavelets, a distinct mathematical foundation. The DWT decomposes data into wavelets, which are localized both in time and frequency. These wavelets are derived from a mother wavelet function and possess the unique property of capturing high and low-frequency components in data effectively. There are several wavelet variants of the DWT, each with its unique properties and applications. Some of the most commonly used variants include Haar[42], Daubechies[43], Coiflets[44], and Symlets[45]. Depending on the application, there are a multitude of other wavelet variants that can be used.

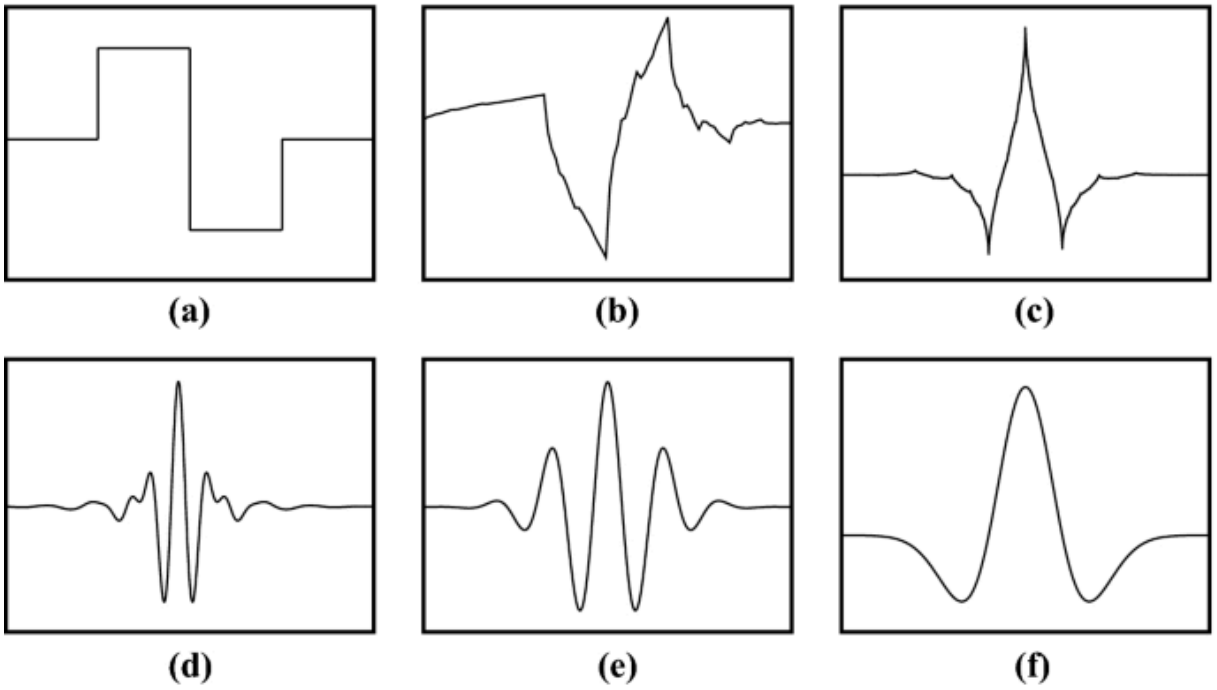

Figure 2. Examples of a few popular wavelets: (a) Haar, (b) Daubechies, (c) Coiflet, (d) Meyer, (e) Morlet, (f) Mexican Hat[26]

The decomposition process using DWT involves passing the data through two filters, one for low frequencies and one for high frequencies. The main signal is embedded within the low frequencies, while the details are embedded within the high frequencies.

To reconstruct the original data we begin by reconstructing the main signal and then incorporate the details to restore the original signal. This is achieved through the use of the inverse DWT. The specific filters and equations may vary based on the chosen wavelet, but this fundamental approach defines the DWT.

The DWT decomposition is given by[46]:

$$\phi_{j,n}[t] = 2^{\frac{j}{2}} \sum_{n} c_{j,n} \phi[2^j t - n]$$

$$\varphi_{j,n}[t] = 2^{\frac{j}{2}} \sum_{n} d_{j,n} \varphi[2^j t - n]$$,where:

- ϕ(t) represents the scaling function used to generate the approximation coefficients
- φ(t) represents the wavelet function used to generate detail coefficients.
- c_j is the scaling coefficient at scale j,
- d_j is the wavelet coefficient at scale j.

Parseval's theorem can be used to explain the relationship between the input signal and each order of wavelet coefficients. With time k and scale j, it can be expressed as:

$$\frac{1}{N}\sum_{t}|X[t]|^2 = \frac{1}{N_j}\sum_{k}|cA_{j,k}|^2 + \sum_{j=1}^{j}\left(\frac{1}{N_j}\sum_{k}|cD_{j,k}|^2\right)$$ [46] , where:

- cA represents the approximated coefficients
- cD represents the detailed coefficients
- N is the sampling period

Similarly to the previously mentioned transform techniques, DWT most commonly performs data compression by truncating a portion of the resulting detail coefficients, that capture high-frequency variations such as noise or outliers. Thus, since Parseval's equation also applies to DWT, for the distance between the original data and the data after DWT is applied, D(x,y) ≥ D(X,Y) holds.

The reverse data summary is derived by the truncated coefficients and inverse DWT is expressed as[47]:

$$cA_{j,k} = \sum_{m} h_{k-2m} cA_{j+1,m} + g_{k-2m} cD_{j+1,m}$$

Where g_k is the wavelet filter and h_k is the scaling filter.

2.3.2 DWT Applications

The DWT plays a pivotal role in several domains, often with specific wavelets tailored to each task.

Audio and Image Processing
In this domain, wavelets such as Daubechies wavelets are commonly used[48]. They enable tasks like denoising[49], image enhancement, and compression through standards like JPEG2000, making them valuable for scientific fields such as medical imaging and geophysics[50].

Biometrics
Wavelets like the Haar wavelet are widely employed for applications such as face recognition and fingerprint analysis[51]. They play a vital role in feature extraction for accurate authentication and identification.

Anomaly Detection
Various wavelets, including the Morlet wavelet, are instrumental in the realm of anomaly detection. They are applied in various fields, such as climate anomaly detection[52] and fortifying network security[53].

Environmental Monitoring
The DWT is also prevalent in the analysis of environmental data. For instance, the Haar wavelet can be applied in seismic signal analysis[54], while the Mexican Hat and Morlet wavelets can be used for weather pattern recognition[55].

These applications illustrate how the DWT remains a powerful mathematical tool, adaptable, and pivotal in driving innovation across various disciplines.

2.3.3 DWT Data Compression

The steps towards using DWT for sensor data compression include:

- Once again we begin with a dataset split into windows of varying sizes.

- We apply the wavelet function of our choice to each data window. The Discrete Wavelet Transform applies high and low-pass filters to break down input data into detail and approximation coefficients respectively. Detail coefficients capture high-frequency variations and fine details, including noise or outliers. In contrast, approximation coefficients represent the essential data, conveying the dataset's core characteristics. This application of a high-pass and a low-pass filter is a fundamental aspect of DWT's data analysis and compression capabilities.

- Similarly to the other transform-based compression techniques, to achieve compression we begin by sorting the DWT outputs in descending magnitude order and retaining the coefficients with larger magnitudes. The choice between keeping the detail or approximation coefficients depends on the specific application's objectives. In our case, we opt to discard some of the detail coefficients. This approach allows us to achieve compression while prioritizing the retention of the datasets' most essential information. Higher compression ratios involve discarding more coefficients.

- The inverse DWT (IDWT) can be used to recover an approximation of the original data by using the retained approximation and detail coefficients to reconstruct the original data.

DWT compression is able to eliminate non-essential information through high-pass and low-pass filters, which not only enhances data quality but also effectively reduces the dimensionality of the dataset, making it more manageable and efficient for further analysis[30].

In summary, the Discrete Wavelet Transform stands out as a powerful departure from conventional Fourier-based techniques. Its distinctive feature lies in the ability to break down data by making use of various wavelet variants. It has demonstrated its versatility across an array of domains, such as audio and image processing, biometrics, anomaly detection, and environmental monitoring. Furthermore, the DWT has had a major impact on data compression by efficiently preserving essential information while eliminating non-essential details.

2.4 THE PIECEWISE AGGREGATE APPROXIMATION (PAA)

2.4.1 Introduction to PAA

The Piecewise Aggregate Approximation (PAA)[56] is a technique that provides an alternative to traditional transform-based methods. The core concept of PAA entails partitioning input data into segments of equal length and calculating the mean value of each segment. The selection of the segment length depends on the specific application and the desired level of data compression. Shorter segments are better at capturing details, whereas longer segments yield a more generalized perspective of the data.

Assuming a time series X of length n, that is represented in N space by a vector $\overline{X} = \overline{x_1},...,x_N$ .The i^{th}element of $\overline{X}$ is calculated by the following equation[56]:

$$\overline{x_i} = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j$$

To include a guarantee on how PAA performs based on the distance between the original data and the data after PAA, assuming the Euclidean distance between two time series X and Y is given by: $D(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$, we can use this formula to derive a relationship between X and Y after the application of PAA[37]:

$$DR(\overline{X}, \overline{Y}) \equiv \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^{N} (\overline{x_i} - \overline{y_i})^2}$$, where

$$DR(\overline{X}, \overline{Y}) \leq D(X, Y)$$

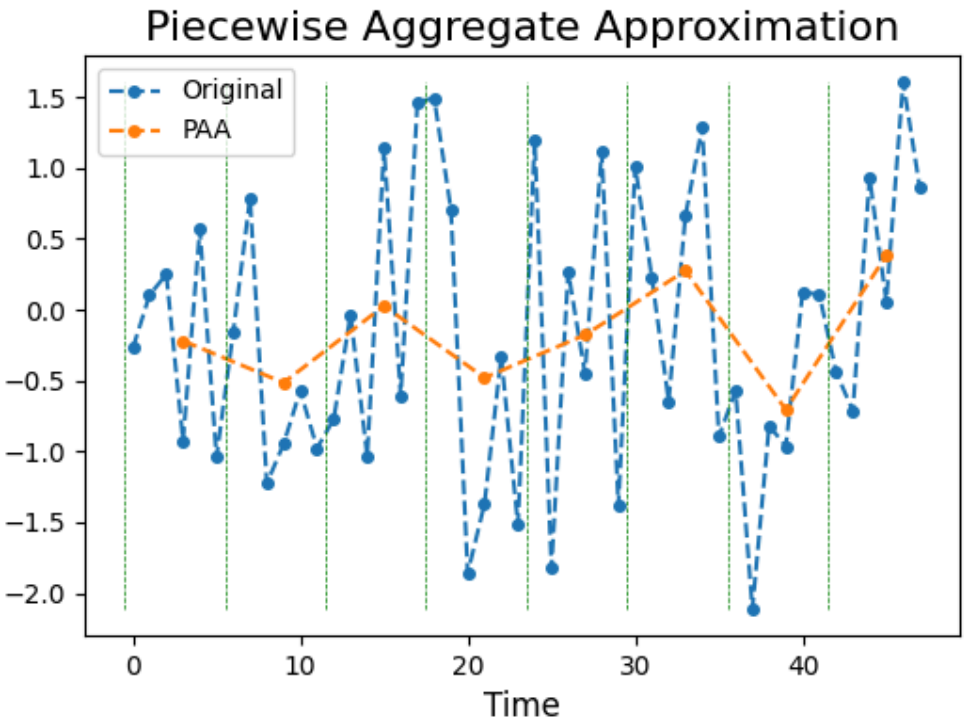

Figure 3. PAA application example[34]

PAA stands out as a relatively straightforward technique when contrasted with more complex methods, yet it possesses remarkable versatility. Its simplicity lends itself to easy modification and adaptation to suit a wide array of applications and requirements. PAA can be tailored by altering segment lengths, experimenting with various aggregation functions, or even integrating PAA with other methods to amplify its performance, making it a flexible tool for addressing diverse data analysis challenges.

2.4.2 PAA Applications

Here, we will explore how PAA plays a pivotal role in several fields, often using specific strategies tailored to the task at hand.

Data Analysis
PAA finds application in data analysis, enabling feature extraction for a range of pattern recognition tasks. For instance, within the financial sector, it can be used as a tool to provide stock market trend analysis[57]. Furthermore, PAA proves its effectiveness in spotting anomalies in various areas, such as optimizing the energy performance of buildings[58].

Eco-Monitoring
Climate and meteorology experts utilize PAA for analyzing data, such as temperature and precipitation records, enabling the detection of heatwaves and other phenomena[59].

Transportation Safety
In the domain of transportation safety, PAA assists in various applications, such as identifying driver behavior patterns to enhance road safety[60].

Healthcare
PAA has a wide range of uses in biomedicine. It can be used to simplify the analysis of Electrocardiogram (ECG) signals, making it easier to spot and diagnose heart conditions[61]. It can also be valuable in diabetes research[62], offering insights into this area of study.

Gesture Recognition
In Gesture Recognition, especially in fields like human-computer interaction and motion analysis, Piecewise Aggregate Approximation (PAA) simplifies and identifies complex hand and body movements[63], making it easier for users to interact with digital systems seamlessly and intuitively.

PAA is an adaptable method across various fields, boasting a multitude of applications, some of which we mentioned here, showcasing its influence on data analysis and pattern recognition.

2.4.3 PAA Data Compression

PAA functions as a compression method by partitioning the data windows into smaller, non-overlapping segments, and subsequently determining the mean of each of these segments. Consequently, the size of each segment determines the level of compression. This process results in a streamlined representation of the original data, effectively diminishing the dataset's dimensionality[64] while retaining its fundamental features.

It's worth mentioning that there an inverse PAA method does not exist, since the mean values used to represent each segment in PAA cannot uniquely determine the original data. PAA is most valuable in scenarios where the primary objective is the preservation of essential information while significantly reducing data size, rather than striving for a close reconstruction of the original dataset.

To summarize, PAA is a versatile data analysis technique known for its use of segment means. It excels in a wide range of applications, including financial trend analysis, climate monitoring, transportation safety, healthcare, and gesture recognition. Moreover, it serves as an efficient data compression method, reducing dataset dimensionality while preserving essential information.

In the preceding chapter, we examined the historical evolution and practical applications of key techniques such as the Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), and Piecewise Aggregate Approximation (PAA). In the upcoming chapter, we will introduce our novel compression approach, based on matrix multiplication with random vectors and random hyperplane projection.

CHAPTER 3: OUR PROPOSED REVERSIBLE DATA COMPRESSION METHOD

In this chapter, we present the sensor data compression and reconstruction technique we developed and tested, which has the potential to significantly reduce the storage and transmission requirements for sensor data while maintaining its essential information. This technique utilizes Random Hyperplane Projections, a concept that can provide multiple advantages in data compression and processing. Before exploring the details of our approach, it is essential to understand the fundamental principles that form the basis of our method.

3.1 Random Hyperplane Projection

Random Hyperplane Projection finds application in various machine learning and data analysis tasks, particularly assisting in dimensionality reduction. It relies on the Johnson-Lindenstrauss lemma, which proposes that sufficiently high-dimensional data points, can be projected into a lower-dimensional space while approximately preserving the distances between them[65]. To achieve this, assuming we want to leverage Random Hyperplane Projection to compress two data vectors u_i and u_j into binary bitmaps Xi and Xj, the relationship between the angular separation of the original vectors (θ(ui, uj)) and the Hamming distance in the compressed space (Dh(Xi, Xj)) is captured by the following formula[66]:

$$\frac{\theta(u_i, u_j)}{\pi} = \frac{D_h(X_i, X_j)}{d}$$

High-dimensional data representation is a common occurrence in real-world applications leading to increased computational needs and transmission costs, noise, and significant challenges for data organization. A random hyperplane in high-dimensional data is a flat, affine subspace represented by vectors, which serves to divide the space into two regions. In various machine learning and data analysis applications, the projection step assigns a binary code (e.g., 0 or 1) to each data point based on its position relative to the hyperplane, effectively encoding its location. This introduces an element of randomness, often employed in algorithms such as random projection and random forests, which can aid in managing computational complexity and improving algorithm performance. The binary codes from multiple random hyperplanes are

combined to create a lower-dimensional representation of the data, which is typically much smaller than the original.

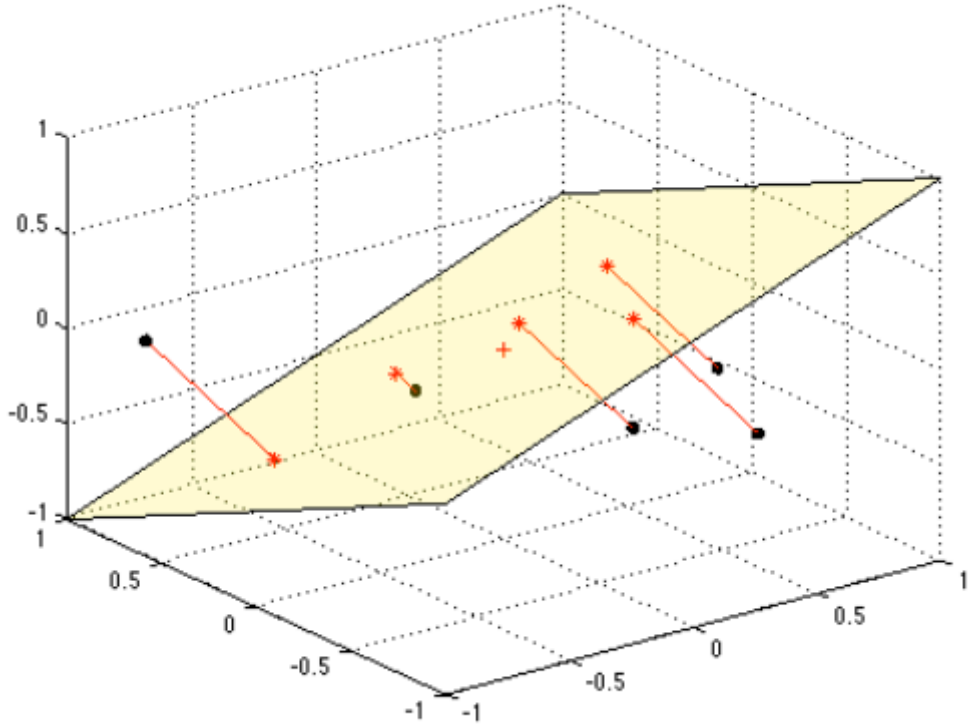

Figure 4. Random Hyperplane Projection[43]

In addition to reducing dimensionality and computational costs, in recent research, Random Hyperplane Projection has been applied for the detection of outliers, as evidenced by studies such as [66], [67], and [68].

By incorporating random hyperplane projection into our data compression and reconstruction technique, we aim to enhance the efficiency and precision of sensor data storage and transmission, especially in scenarios where outlier detection and data integrity hold significant importance. In the following section, we will demonstrate the application of our technique.

3.2 Compression Phase

The initial phase of our methodology centers on data compression through the utilization of Random Hyperplane Projection. Applying a similar approach as we did for the compression techniques we previously mentioned:

- We begin by partitioning the dataset into separate windows, each with a window size chosen from a set of values like 16, 32, 64, and 128, enabling us to work with manageable subsets of data.

- For every row within these data windows, we introduce an element of stochasticity by multiplying the data with a vector array sampled from a standard normal distribution, N(0,1). This vector array is structured with a number of rows equal to the data window size, and it contains D columns. The value of D is calculated as the product of the data points' bit value and the window size divided by the desired compression ratio. The multiplication result is a one-dimensional array of size D.

- In the realm of random hyperplane projection, each value within the 1D array is assigned either a 1 or a 0, depending on whether it falls on the positive or negative side of the randomly generated hyperplane. This binary representation serves as an efficient encoding of the original data, reducing its dimensionality while preserving critical information.

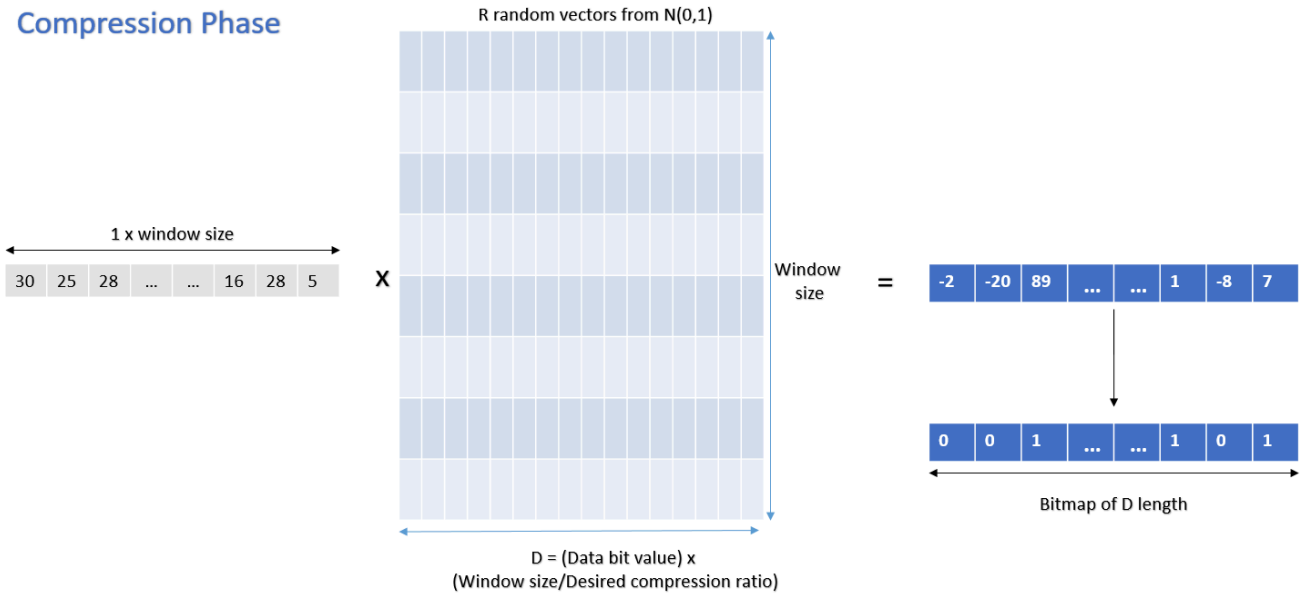

Figure 5. Compression Phase Visual Showcase

3.3 Decompression Phase

In the second phase of our technique, the decompression phase, we aim to reconstruct the compressed data while retaining its essential properties. To achieve this, we utilize the right-inverse, of the D-sized random vector array used in the compression phase.

The process of decompression effectively reverses the compression operation. To reconstruct a close approximation of the original data, we multiply each one-dimensional bitmap created in the compression phase with the right inverse of the D-sized random vector array that was used to create it.

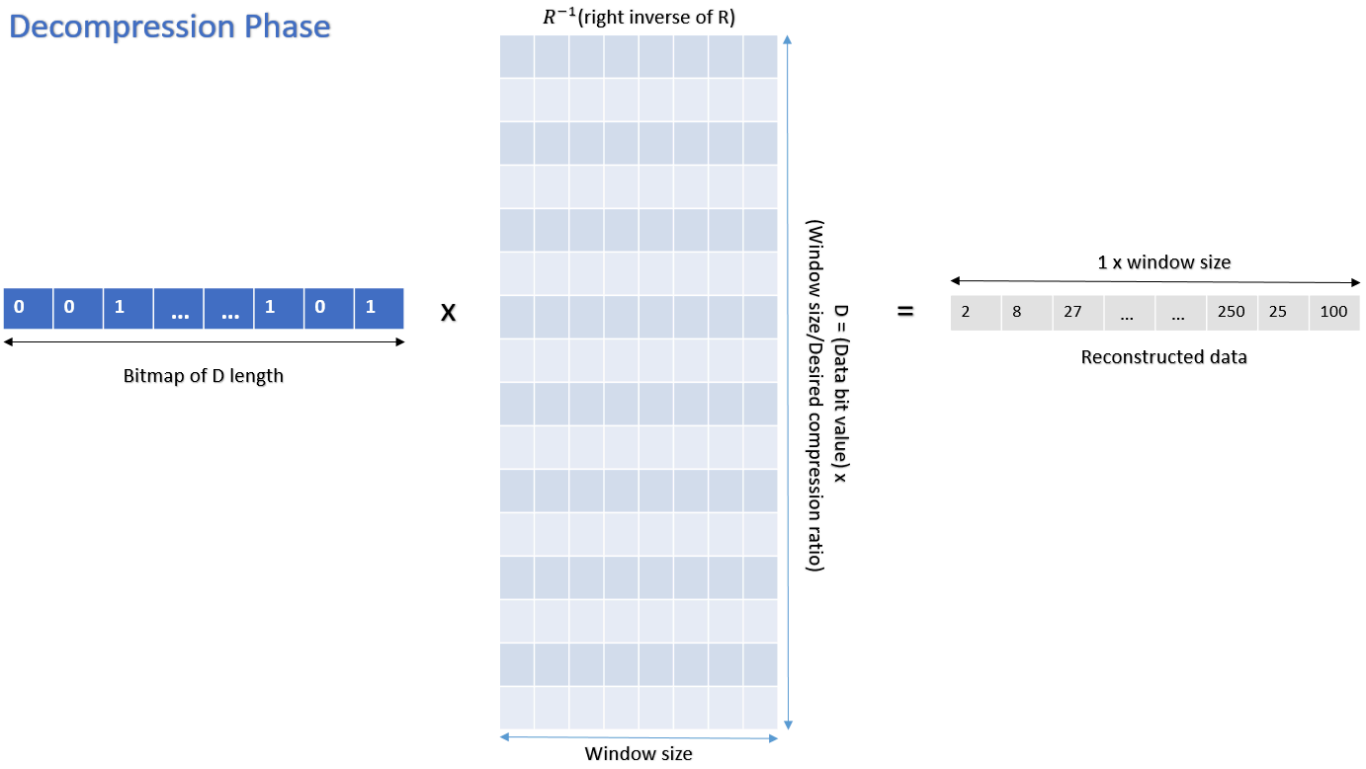

Figure 6. Decompression Phase Visual Showcase

To summarize, the presented sensor data compression and reconstruction technique uses Random Hyperplane Projection to efficiently reduce dimensionality and enhance data storage and transmission. By introducing an element of randomness, this approach not only addresses computational challenges but also opens up possibilities for outlier detection and improved data integrity. Overall, it offers a promising solution for managing high-dimensional sensor data in real-world applications.

CHAPTER 4: TOSSIM SHOWCASE

This chapter introduces TOSSIM, a robust sensor network simulation environment that serves as a valuable platform for testing and experimenting. Within this chapter, we will highlight TOSSIM's capabilities, delve into its various applications, and explain its role in evaluating the performance of both compressed and uncompressed data transmission in a simulated Wireless Sensor Network (WSN) we developed.

4.1 Introducing TOSSIM

TOSSIM, short for TinyOS Simulation, is a discrete event simulator specifically designed for Wireless Sensor Networks. In our research, TOSSIM enables us to replicate the actions of sensor nodes organized in clusters, cluster heads, and a base station in a controlled and repeatable setting. This simulated environment provides a safe and cost-effective way of evaluating different aspects of our sensor network, eliminating the necessity for costly hardware or the intricacies of real-world experimentation.

TOSSIM offers a wide array of features, including:

- Accurate TinyOS-based network simulations that faithfully reproduce the behavior of sensor nodes. This includes realistic modeling of radio communication, energy consumption, and resource limitations.

- Effective scalability, simulating networks of various sizes, from small sensor setups to networks with thousands of nodes.

- Users can exercise precise control over the simulation process using nesC and Python scripts, managing network initialization, data collection, radio channels, network topologies, noise, and much more.

- High repeatability, experiments in TOSSIM allow us to validate our findings and compare results across multiple trials confidently.

4.2 TOSSIM Configuration

Our simulated Wireless Sensor Network (WSN) consists of MicaZ sensor nodes organized into clusters, with each cluster comprising of six sensor nodes. These

clusters serve as the basic building blocks of our network. In this setup, sensor nodes within each cluster communicate with a specific cluster head(each cluster connects with a different, unique, cluster head), which, in turn, forwards the data to a base station.

Our network communication setup ensures that when a transmitted message is not acknowledged by the corresponding cluster head (if the sender is a node) or by the base station (if it's a packet forwarded from a cluster head), the system will automatically retransmit the message until it is successfully received. This functionality mimics the real-world challenges of dealing with an unreliable wireless channel, ensuring dependable data transmission.

To assess the performance of data compression, we conducted the following scenarios:

Scenario A*: *Uncompressed Data
In this scenario, we transmitted raw sensor data without any compression. This served as our baseline for evaluation metrics.

Scenario B*: *Compressed Data
In contrast, we transmitted data packets containing the outputs of our chosen compression techniques, applied to the raw data, intending to reduce the amount of transmitted data packets and conserve network bandwidth and residual energy.

Our aim was to measure three key parameters:

- **Bitwise Comparison**
  We compare the number of bits required to transmit the entire dataset in both scenarios. This comparison is crucial to understanding the potential benefits of data compression, as it directly impacts the efficiency of our network in terms of energy consumption and transmission time.

- **NACKS Cost**
  In wireless networks, packet acknowledgments (ACKs) are essential for ensuring the reliability of data transmission. However, they introduce additional bits needed to resend packets that were not acknowledged

(NACKs). We quantify how many of the bits transmitted are a result of NACKs, shedding light on the trade-off between reliability and network efficiency.

- **Power Consumption Estimation**
  Power consumption is a critical factor that directly influences the longevity of sensor nodes. We calculated the power consumption in both Scenario A (Uncompressed Data) and Scenario B (Compressed Data). By comprehending the power demands in each scenario, we aim to evaluate the energy-saving potential of data compression techniques and their influence on the network's sustainability. This analysis is vital in informing decisions regarding node deployment and network management in practical applications.

In an upcoming chapter, we will present the results obtained from our TOSSIM experiments, offering an in-depth examination of data transmission efficiency within our simulated Wireless Sensor Network (WSN). We will explore the effects of data compression, assess the implications of NACKs, and arrive at insights concerning the balance between data compression and reliability.

CHAPTER 5: DATASETS, ALGORITHMS, AND EVALUATION METRICS

In this chapter, we provide an in-depth exploration of the datasets, machine learning algorithms, and algorithm assessment metrics employed throughout our experiments.

5.1 Datasets and Data Preprocessing

In our experimental procedures, we employed two datasets.

IntelLabData Dataset

The IntelLabData dataset[14] includes measurements from environmental sensors, such as temperature, humidity, voltage, and light measurements. To analyze these measurements, we treated each type of data separately, creating four distinct datasets. We organized the datasets in a way where each row in these datasets represented measurements from one sensor. The data was then divided into windows, with each window containing 16, 32, 64, or 128 data points (columns), as shown in Figure 1. Each row within these windows was used as input for clustering and regression algorithms, with options for either compressed or raw data. Subsequently, metrics were computed for each algorithmic application.

Pump Sensor Data Dataset

The second dataset was sourced from [15], providing a larger number of data points captured at different time intervals. Notably, the dataset includes an additional column labeled "machine status" that describes the network's state at each specific time moment, categorizing it as either "NORMAL," "RECOVERING," or "BROKEN." This extra feature enables the utilization of classification algorithms for this dataset, which was not possible for the IntelLabData one. To make the dataset more practical and manageable in terms of computation time, each time we randomly selected 2000 rows from the dataset, these rows were then shuffled.

When dealing with classification tasks, the "machine_status" label played a central role and was closely tied to specific timestamps. We chose to preserve the original data format to maintain the fidelity of the label information for classification algorithms. In this original format, sensor measurements were arranged column-wise instead of the previous row-wise organization. The data

division in windows of 16, 32, 64 or 128 measurements was also done column-wise before compression. This approach ensured that the connection between sensor data and machine status remained intact, preserving the integrity of the temporal relationships between sensor measurements and machine status, which are critical for classification algorithms.

The preprocessing of the datasets also includes using a MinMaxScaler for feature scaling. This scaler normalizes each feature to a range between 0 and 1, addressing differences in feature scales. This normalization can help algorithms work better and produce more accurate results. It also aids in comparing our suggested compression technique with the other methods, as all methods are now on a consistent scale, making the evaluation more reliable.

5.2 Machine Learning Algorithms

We applied a set of machine learning algorithms to both datasets, each tailored to suit the dataset's specific nature and research objectives:

- **K-means:** K-means[5] is a popular clustering algorithm used in unsupervised machine learning that groups similar data points together. It works by repeatedly assigning data points to clusters based on the average (centroid) of the points in the cluster, and then updating the centroids. This process repeats until convergence, where the assignment of data points to clusters remains stable. K-means is straightforward and efficient, often used for tasks like grouping data and finding patterns. However, it needs the number of groups to be specified beforehand and can be influenced by the initial setup. The number of clusters (K) is the most important parameter and has to be set by the user before the algorithm is applied to the data. To determine an optimal K, we utilized the Elbow Method[69], a graphical approach that helps identify the ideal number of clusters in a dataset by locating the point on a plot where the within-cluster sum of squares shows a distinct "elbow", as showcased in the figure below.

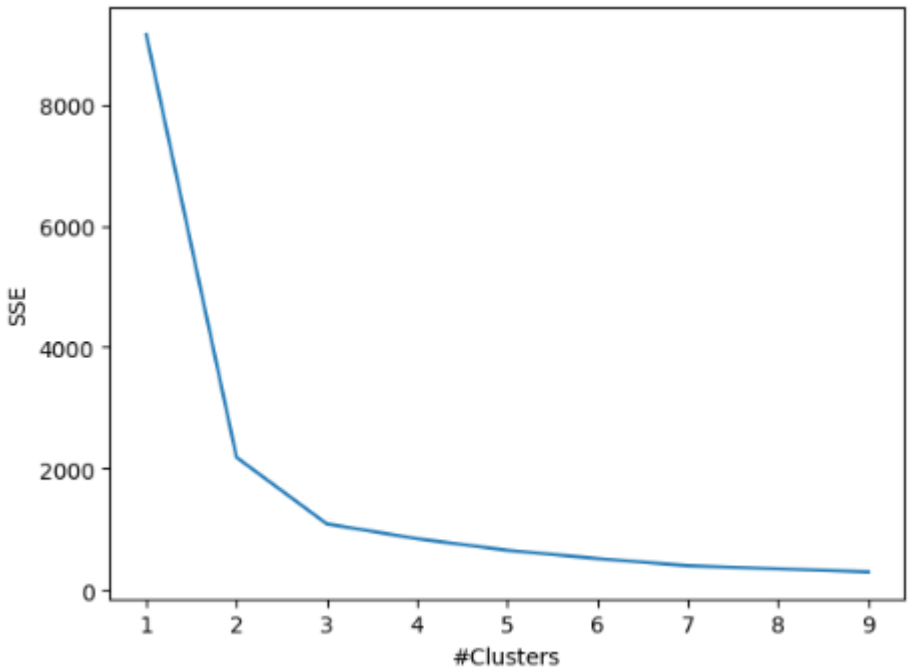

Figure 7. K-means elbow criterion for the Pump Sensor dataset. In this case, k = 2 is the point where SSE noticeably changes.

- **DBSCAN:** DBSCAN[6], which stands for Density-Based Spatial Clustering of Applications with Noise, is an algorithm for clustering that groups data points by considering their proximity and the density of neighboring data points. Data points are categorized as core points, border points, or noise points based on factors such as their neighborhood radius (represented by the epsilon value) and the minimum number of points needed to constitute a dense region.

  To determine the optimal epsilon for DBSCAN, a common approach involves computing the k-distance (typically with k set to 2 or 4) for each data point. After sorting these distances, we can apply the elbow method and pinpoint the "knee" in the resulting plot. This knee point serves as an indication of the optimal epsilon value we can use for DBSCAN.

  The rationale behind this method lies in the observation that points within clusters tend to exhibit smaller k-nearest neighbor distances, indicative of proximity. In contrast, noise points tend to display larger distances, suggesting isolation[70]. By visually identifying the knee in the plot, we effectively select an epsilon value that accurately captures cluster boundaries while excluding noise points. This visual approach provides an intuitive means of determining an appropriate epsilon for DBSCAN, aligning with the inherent characteristics of the data distribution.

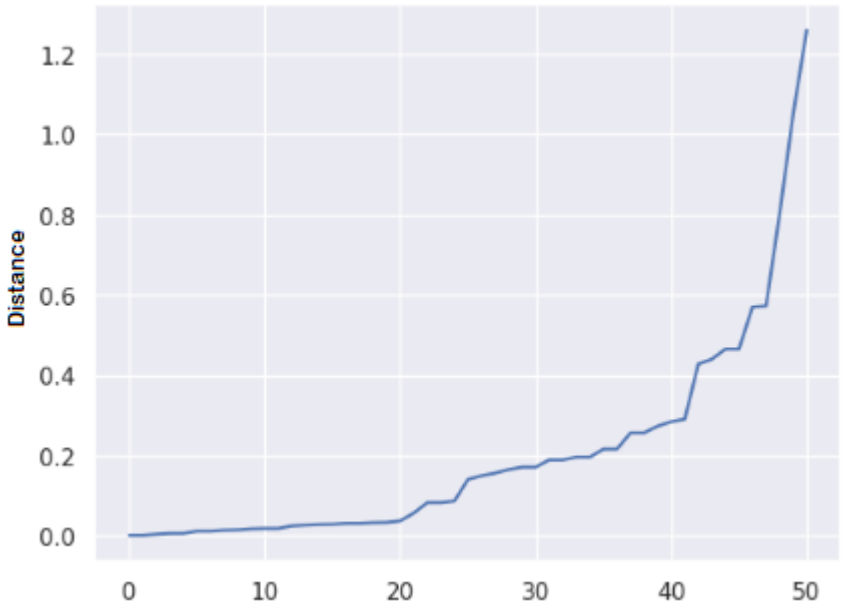

Figure 8. Second nearest neighbor distances for clustering, Pump sensor dataset(for window = 16)..

- **Linear Regression:** Linear regression[7] is a statistical tool that helps us see the link between two or more factors. The main goal is to find the best straight-line equation that describes how a dependent variable (the one we want to predict) relates to one or more independent variables (the ones affecting the prediction). In simpler terms, it's like finding the most fitting line that shows how changes in certain factors connect to changes in the outcome we're interested in. In our linear regression implementation, we employ a 75/25 split for each window before applying linear regression. This entails dividing the data into a 75% training set, used for model training, and a 25% testing set, utilized to assess the model's predictive performance.

  We also adopt a 75/25 split for all classification algorithms and the neural network discussed below. This split ensures that the models are exposed to diverse data during training, providing a reliable assessment of their performance on new, unseen data. This approach enhances their capacity to generalize effectively to new observations.

- **Logistic Regression:** Despite its name, logistic regression[8] is mainly used for classification tasks. Instead of giving exact values, it estimates the likelihood of an event occurring. It uses the logistic function, or sigmoid, to transform input features with weighted coefficients,

generating probabilities between 0 and 1 for class prediction. The model is trained with maximum likelihood estimation, adjusting these weights to make predicted probabilities match actual outcomes.

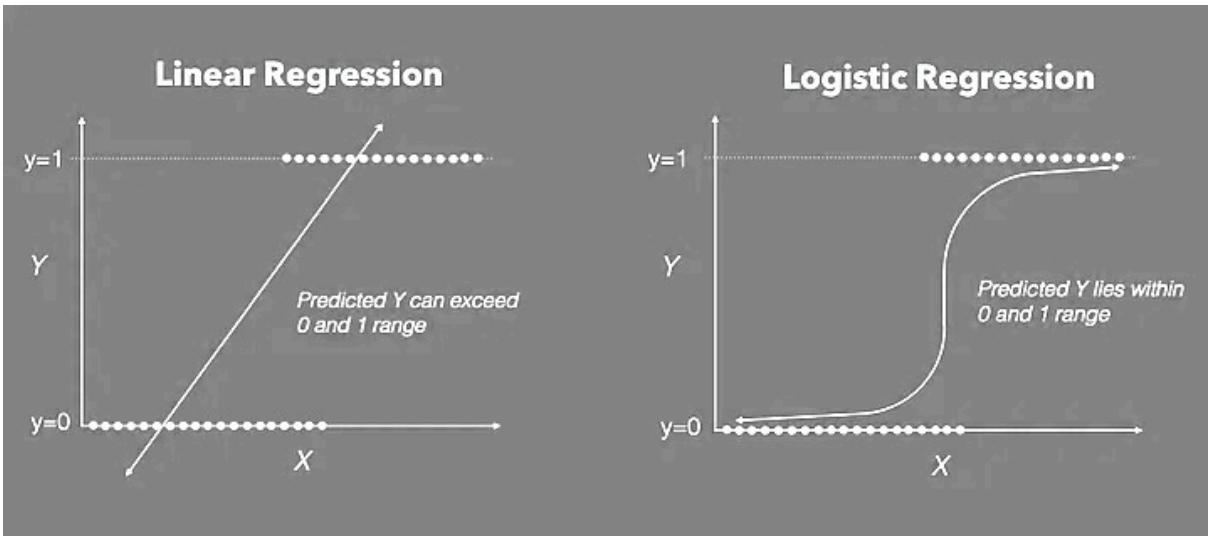

Figure 9. Linear and Logistic Regression visualized.[48]

- **K-Nearest Neighbors (KNN):** K-Nearest Neighbors (KNN)[9] is a straightforward and effective machine learning method for classifying data. It calculates distances between the point to be classified and all others in the training set. The "k" represents the number of neighbors considered. Once distances are calculated, the algorithm looks at the k-nearest neighbors and assigns the most common class among them to the data point, based on the idea that similar points belong to the same class. To select the most effective for our application, we experimented with various values on our uncompressed data and chose the value that resulted in the best metrics.

- **Support Vector Machines (SVM):** SVM[10] is a machine learning algorithm mainly used for data classification and for predicting values within datasets. Conceptually, it operates by finding the hyperplane that maximizes the margin, or the distance between the hyperplane and the data points of each class that are closest to it, known as support vectors. Our approach to selecting the most effective SVM parameters for each of our applications was similar to the KNN approach, where we experimented with various combinations of the raw data and selected the ones that resulted in the best metrics.

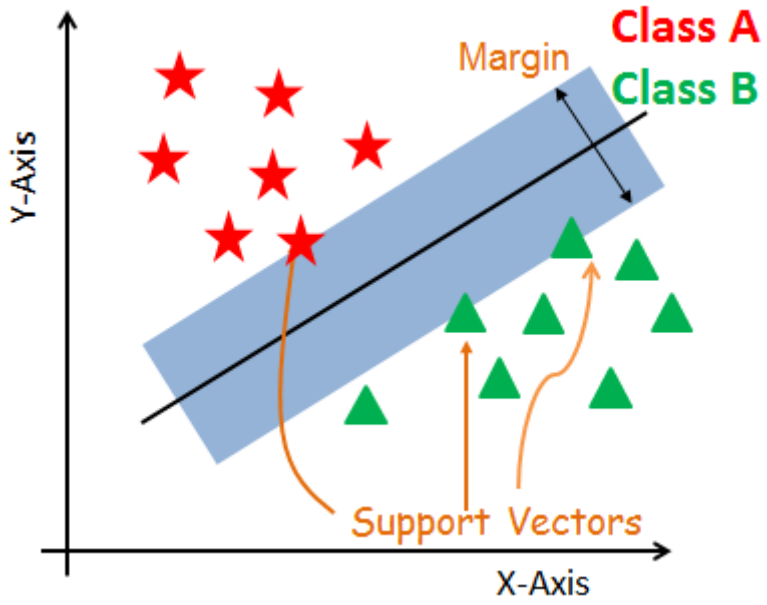

Figure 10. Two-dimensional example of SVM algorithm application.[48]

- **Neural Network:** Neural networks[11] belong to a class of machine learning algorithms that draw inspiration from the structure and operations of the human brain. These computational models are comprised of interconnected nodes, commonly referred to as artificial neurons, arranged in layers. Each connection between neurons has a weight, and by using forward propagation, the network turns input data into useful predictions. The activation functions, like ReLU in our model, introduce flexibility to our neural network processes information. This bending helps the network grasp complicated patterns in the data, making it better at understanding and learning from the information it's given. Training involves adjusting weights to minimize prediction errors through a

process called backpropagation.

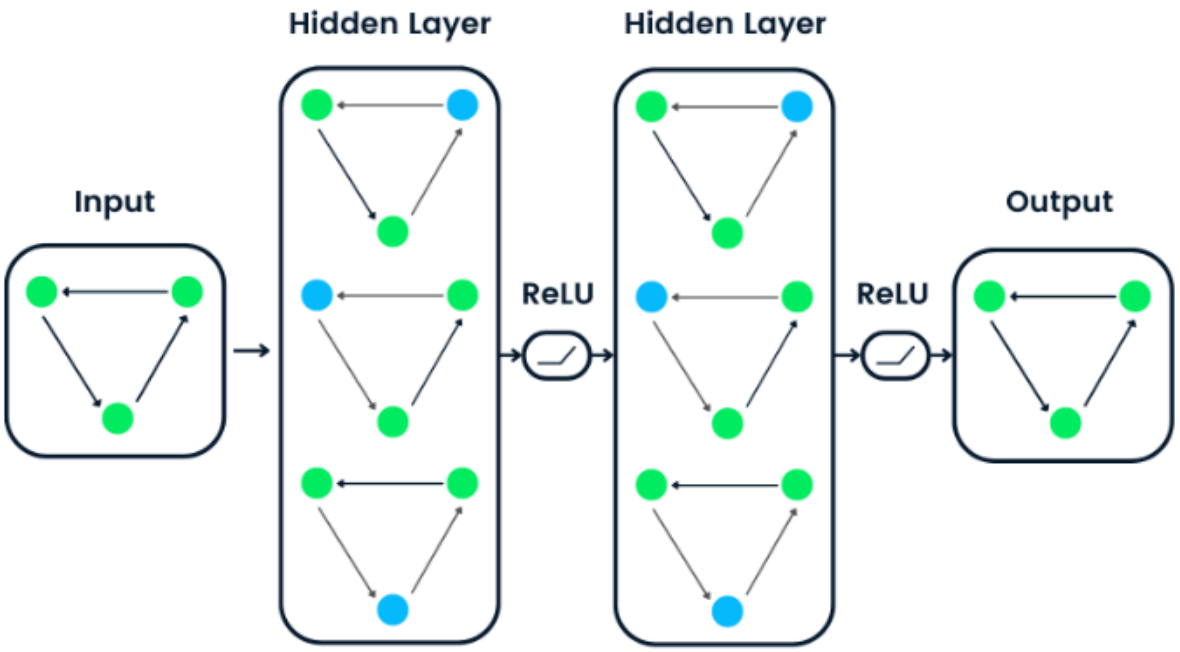

Figure 11. Neural network visual showcase.[71]

In our model, we used a Sequential neural network design where information moves step by step from the input layer, passes through hidden layers, and finally reaches the output layer. The first layer has 64 neurons, followed by 32 neurons in the subsequent layer. To prevent overfitting, where the model becomes too specialized to the training data, we've also incorporated dropout layers. These layers randomly deactivate some neurons during training, promoting a more robust and general understanding of the data.

```
# model
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(3, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train model
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10)
history = model.fit(X_train_scaled, y_train, validation_split=0.2, epochs=50, batch_size = window_size , callbacks=[es])
```

Figure 12. Our neural network model.

The model is trained using the Adam optimizer and sparse categorical crossentropy loss function, which are common in similar applications, while predictions are generated using softmax activation which is suitable for multi-class classification problems. Early stopping, with a patience of 10 epochs, is also implemented to prevent overfitting by halting training when the validation loss stops improving.

5.3 Metrics and Evaluation

To evaluate the effectiveness of the machine learning algorithms on both datasets, a diverse set of metrics was utilized. These metrics provide insight into the algorithms' performance in capturing patterns, predicting outcomes, and classifying data points, tailored to the unique features and objectives of each algorithm.

A 95% confidence interval was calculated for the aggregated metrics within each data window. The 95% confidence interval is a statistical technique that aids in approximating a plausible range for the true value of a parameter. In our data analysis, the 95% confidence interval becomes a valuable tool for assessing the uncertainty associated with each metric within a window. By doing this calculation for each window, we get insights into how much the metric varies and how reliable it is across different parts of the data. This approach not only lets us evaluate the metric's central estimate for individual data windows but also helps us understand how precise these estimates are within the broader context of the dataset, making our evaluation more robust and nuanced.

Evaluating Clustering Algorithms (K-means and DBSCAN):
For the clustering algorithms, K-means and DBSCAN, two key metrics were utilized to evaluate their performance: Silhouette Score and Adjusted Rand Index (ARI).

- **Silhouette Score:** The Silhouette Score assesses how clearly defined clusters are in a dataset, ranging from -1 to 1. A score closer to 1 indicates better-defined clusters. To compute the silhouette score the following formula is used [72]:

$$\text{Silhouette Score} = \frac{(b-a)}{max(a,b)},$$

where a is the average distance between each point within a cluster and b is the average distance between all clusters.

- **Adjusted Rand Index (ARI):** ARI evaluates the similarity between true class labels obtained by clustering the raw, uncompressed data and the clusters formed by applying the same clustering algorithm to the compressed and reconstructed data. It serves as a measure of clustering accuracy, ranging from -1 to 1, with 1 indicating perfect clustering alignment, 0 random agreement, and -1 that the two clusterings are entirely different. A simplified formula can be used to provide insight on how ARI is calculated:

$$\text{AdjustedIndex} = \frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex}$$ [73]

Evaluating Linear Regression:

The two metrics employed to evaluate the performance of our Linear Regression application are R-squared (R2) and Root Mean Squared Error (RMSE).

- **R-squared (R2):** R2, tells us how well the independent variables in a model can predict or explain the changes in the dependent variable. It's like a percentage that shows the proportion of variability in the dependent variable that the model can account for. The R2 score ranges from negative infinity to 1. If it's 0, it means the model can't explain any variability, and negative values suggest the model is worse than a simple horizontal line representing the average of the dependent variable. A value close to 1 suggests the model fits the data well. To calculate R-squared, the following formula is used[74]:

$$\text{R-Squared} = \frac{SS_{regression}}{SS_{total}}$$

Where $SS_{regression}$ represents the sum of squares due to regression and SS_{total} is the total sum of squares.

- **Root Mean Squared Error (RMSE):** RMSE is a way to see how much the predicted values in a model differ, on average, from the actual values. It is calculated by taking the square root of the average of the squared differences between predicted and actual values. This mathematical measure helps us understand how accurate the model is, and lower RMSE values mean the model is doing a better job at prediction. Assuming $Y_{p,i}$ represents the predicted output and $Y_{a,i}$ the actual output, then the RMSE is calculated as[75]:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_{p,i} - Y_{a,i})^2}$$

Evaluating Classification Algorithms (Logistic Regression, KNN, SVM, Neural Network):

Two metrics were also employed to evaluate the performance of our classification algorithms: Accuracy and F1 Score.

- **Accuracy:** Accuracy reflects how accurate a model is by showing the percentage of correctly predicted instances and is simply calculated as *Number of correct predictions/Total number of predictions.* The accuracy score ranges from 0 to 1, where 0 means that the model made no correct predictions, and 1 means that the model made 100% correct predictions.

- **F1 Score:** The F1 Score is a balanced measure of a model's performance that takes into account both precision and recall. Precision is the accuracy of positive predictions while recall is the ability to capture all actual positives, in our case, based on the raw data. F1 Score combines these two aspects, making it handy, especially when dealing with imbalances between different classes. Using precision and recall, the F1 formula is expressed as[76]:

$$F_1 = 2\frac{precision \cdot recall}{precision + recall}$$

The F1 score ranges between 0 and 1, where 1 represents the perfect F1 score, indicating a perfect balance between precision and recall.

Having established the foundation of our experiments, we can proceed with the presentation and interpretation of our experimental results.

CHAPTER 6: PERFORMANCE COMPARISON AND RESULTS

After reviewing the compression techniques, tools, datasets, machine learning algorithms, and evaluation metrics used, we're ready to share our experimental outcomes.

6.1 Clustering Results

6.1.1 K-means Results

In the following series of plots, we maintain a consistent window size of 16 while experimenting with varying compression ratios to evaluate the performance of K-means clustering across the different compression methods.

The evaluation is conducted on the Intel Lab Data (abbreviated as ILD) dataset, specifically focusing on sensor measurements related to lights and humidities. This analysis aims to assess how different compression methods impact the clustering performance in the context of the specified sensor data.

The red line in each plot represents the mean of the depicted metric Scores. Above each plot, a corresponding confidence interval for the metric is also depicted.

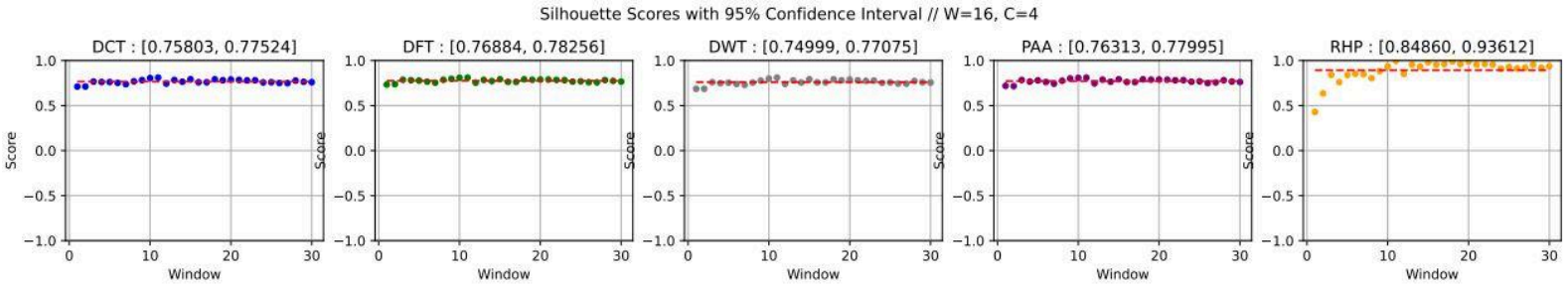

Figure 13. K-means Silhouette Scores for IntelLabData(Lights), Window = 16, Compression Ratio = 4.

Examining the K-means clustering plot for the ILD Light sensor data with a compression factor of 4, RHP stands out with a mean Silhouette score of 0.89235. This is 15% higher than the next best-performing method, DFT, which has a mean score of 0.77569. RHP's confidence interval (CI) [0.84860, 0.93612] also stands out due to its impressive proximity to the optimal value of 1. The

RHP lower limit, in particular, approaches 1 more closely than the upper limits of other method CIs.

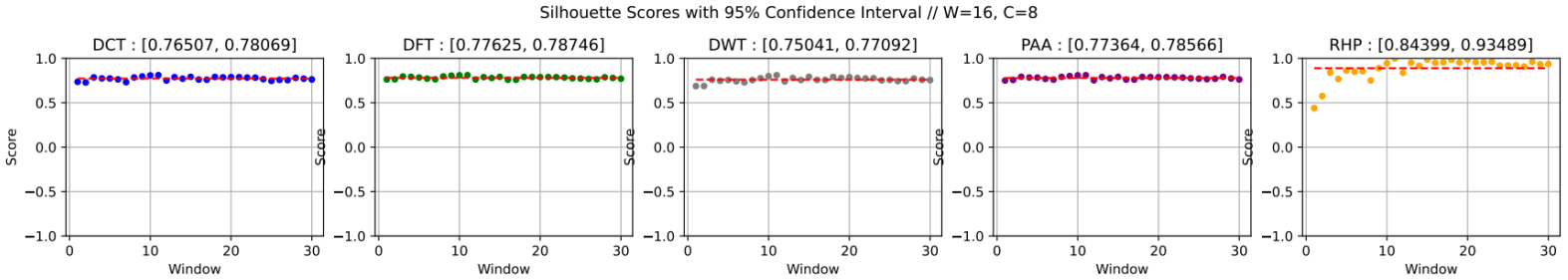

Figure 14. K-means Silhouette Scores for IntelLabData(Lights), Window=16, Compression Ratio=8.

For a compression ratio equal to 8, RHP continues to outperform the rest of the candidates, showcasing a mean Silhouette score of 0.88944, which is 13.7% higher than the next best methods' score, DFTs, which records a mean score of 0.78185. RHP's confidence interval [0.84399, 0.93489] reinforces its reliability, with both lower and upper limits close to the optimal value of 1, including once again a lower limit approaching 1 more closely than the upper limits of the confidence intervals associated with alternative compression methods.

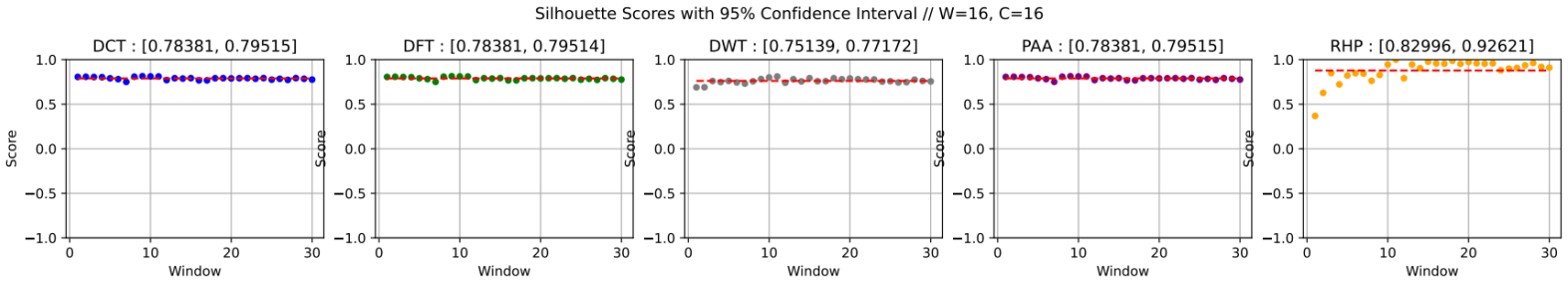

Figure 15. K-means Silhouette Scores for IntelLabData(Lights), Window=16, Compression Ratio=16.

For a compression ratio of 16, RHP remains consistent, demonstrating the best results. In this case, RHP has a mean Silhouette score of 0.87808, outperforming the next-best results by 11.2% (DCT, DFT, and PAA produce the same clustering with a mean silhouette score of 0.78947). The confidence interval for RHP, [0.82996, 0.92621], indicates a performance level similar to previous compression ratios when contrasted with other methods.

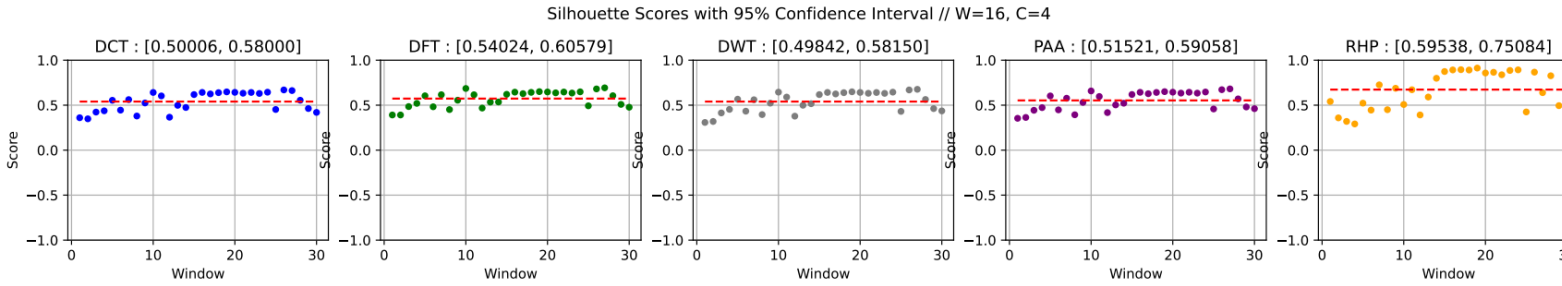

Figure 16. K-means Silhouette Scores, IntelLabData(Humidities), Window=16, Compression Ratio=4.

Switching our focus to the K-means clustering plots for the ILD Humidity sensor data, for a compression factor of 4, RHP showcases a 0.67311 mean Silhouette score that outperforms the next-best mean, belonging to DFT at 0.57301, by 17.4%. RHP's confidence interval [0.59538, 0.75084] reinforces its reliability, maintaining its edge over alternative methods due to its larger upper limit.

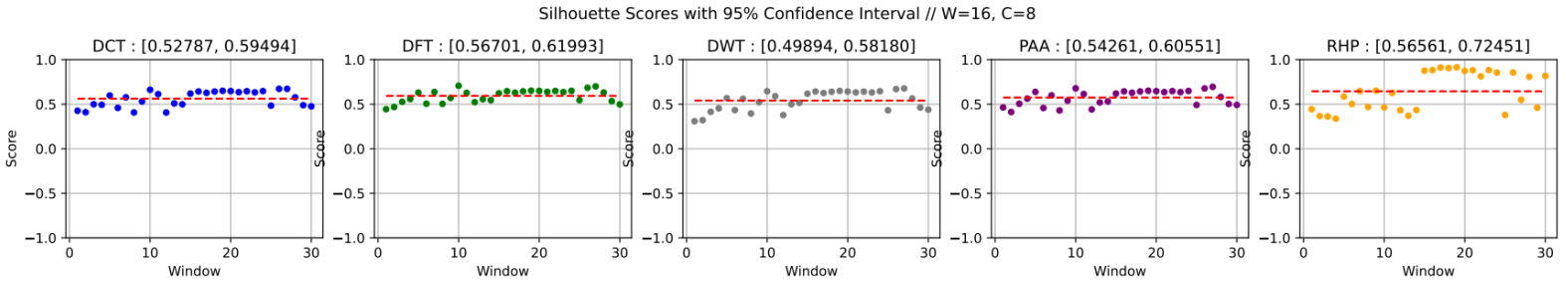

Figure 17. K-means Silhouette Scores, IntelLabData(Humidities), Window=16, Compression Ratio=8.

For a compression ratio of 8, RHP continues to lead, displaying a mean Silhouette score of 0.64505, which is 8% higher than the next best method. RHP's confidence interval [0.56561, 0.72451] remains robust, indicating a sustained level of reliability when compared to other compressions.

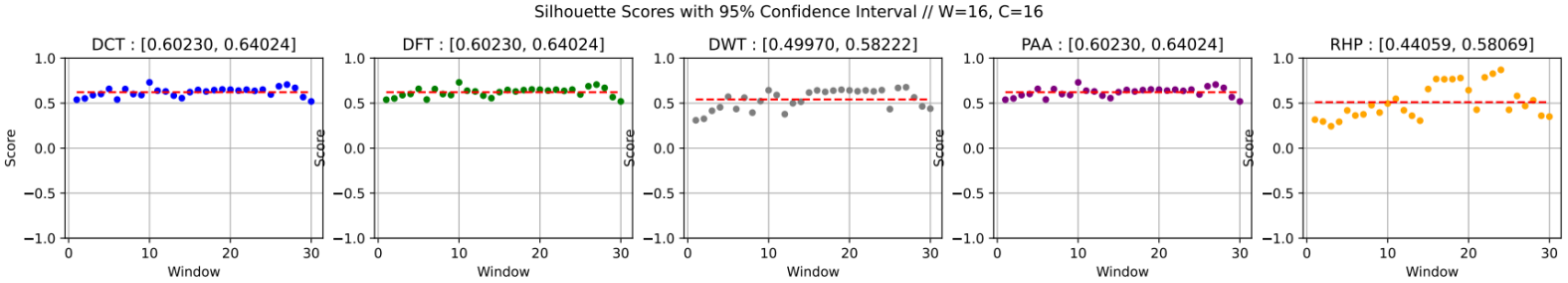

Figure 18.K-means Silhouette Scores, IntelLabData(Humidities), Window=16, Compression Ratio=16.

With a compression ratio of 16, RHP experiences a decline in mean Silhouette score. In this scenario, DCT, DFT, and PAA emerge as the best methods, each with the same mean Silhouette score of 0.62127 and a [0.60230, 0.64024] confidence interval. Even RHP though gets outperformed by 21% when comparing the means, it remains reliable with a mean score of 0.51064 and a [0.44059, 0.58069] confidence interval.

6.1.2 DBSCAN Results

In the subsequent series of plots, our focus shifts from K-means clustering to DBSCAN, this time maintaining a fixed compression ratio of 8 while exploring varying window sizes—16, 32, and 64. This transition allows us to investigate the impact of window size on performance. Similar to the K-means plots we previously presented, our DBSCAN results were calculated using the Intel Lab Data (ILD) dataset. Specifically, our attention centers on sensor measurements related to lights.

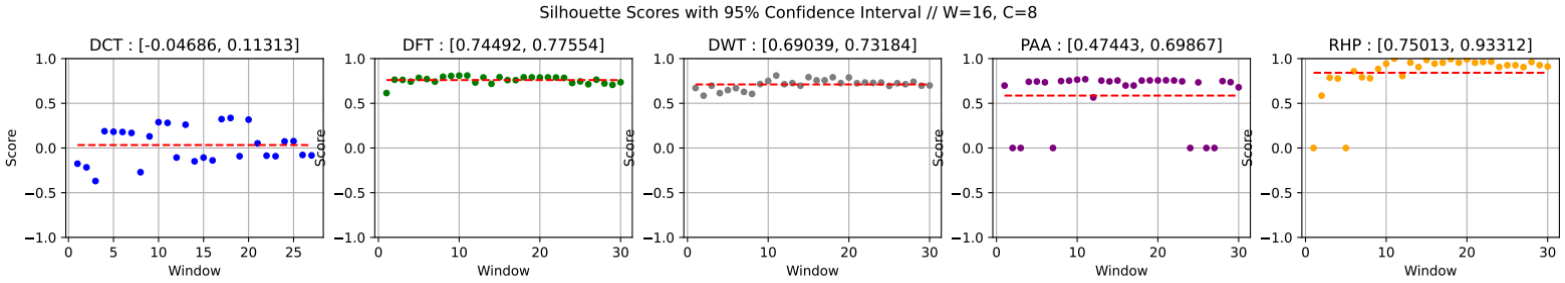

Figure 19. DBSCAN Silhouette Scores, IntelLabData(Lights), Window=16, Compression Ratio= 8.

For a compression ratio of 8 and a window size of 16, the RHP method stands out as a top performer in DBSCAN clustering analysis. It achieves the best mean Silhouette score of 0.84162, outperforming other methods by at least 10.7%. RHP's confidence interval [0.75013, 0.93312] further supports its good performance with an upper limit very close to 1 and a lower limit closer to 1 than the confidence interval upper limits of most of the other compression methods.

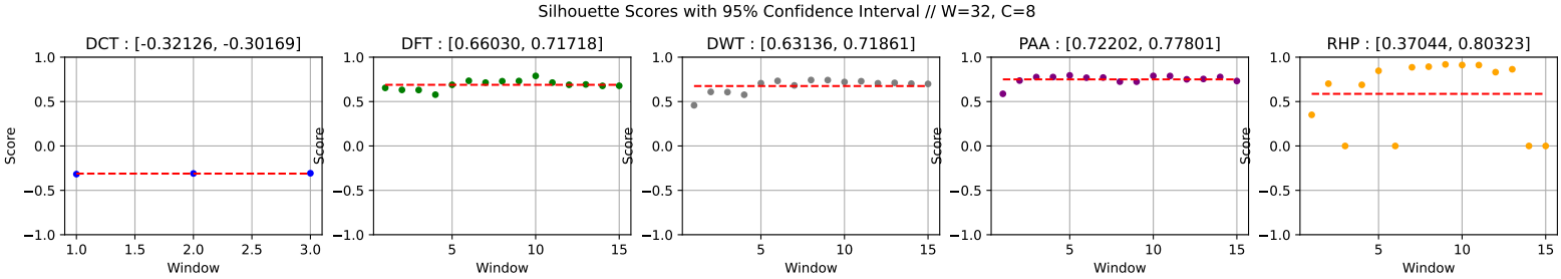

Figure 20. DBSCAN Silhouette Scores, IntelLabData(Lights), Window=32, Compression Ratio= 8.

For a window of 32 data points, RHP encounters a drop in performance, showcasing a mean Silhouette score of 0.58683, which is surpassed by 15%-27.8% from DWT, DFT, and PAA. Despite this, the upper limit of RHP's confidence interval, 0.80323, is close to the optimal value of 1, showing good potentially and outshining other methods in this aspect. However, the performance and stability issue can be seen as RHP's lower limit, 0.37044, is notably lower than the lower limits of alternative methods. This discrepancy indicates a level of instability within RHP's clustering performance for a window size equal to 32, warranting further exploration and consideration.

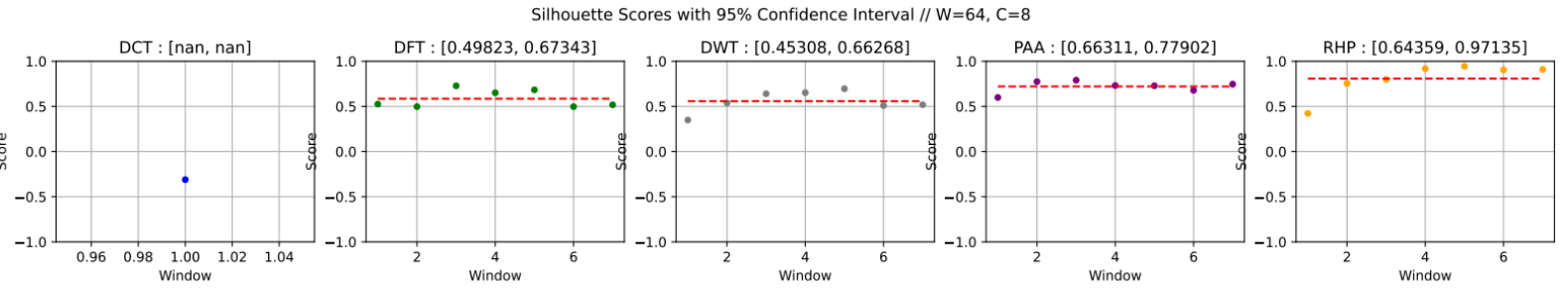

Figure 21. DBSCAN Silhouette Scores, IntelLabData(Lights), Window=64, Compression Ratio= 8.

With a window size of 64, RHP recovers in performance, once again emerging as the top-performing method among those considered. The mean Silhouette score of 0.80747 for RHP outperforms the next-best mean, belonging to PAA at 0.72106, by almost 12%. RHP's confidence interval [0.64359, 0.97135] reinforces its reliability, highlighting its potential perfect silhouette score results and effectiveness even in larger window sizes.

In summary, our clustering analysis on the ILD dataset underscores the reliable performance of the Random Hyperplane Projection method. In K-means clustering, RHP consistently achieves favorable mean Silhouette scores and confidence intervals across various compression ratios. When applied to

DBSCAN with a fixed compression ratio of 8, RHP maintains its effectiveness, excelling particularly in window sizes of 16 and 64, with a minor performance dip observed at a window size of 32. These consistent findings highlight RHP's reliability across diverse scenarios, providing valuable insights for informed decision-making in the context of data compression methods and their impact on data processing.

6.2 Classification Results

Transitioning from clustering analyses, we now shift our focus to classification results using Support Vector Machines (SVM), a Neural Network, and K-nearest neighbors.

6.2.1 SVM Results

For our SVM measurements, we maintain a fixed compression ratio of 8 and explore the impact of varying window sizes—16, 32, 64, and 128—on Accuracy and F1 scores. The evaluations are conducted on the Pump Sensor Data dataset which is suited for classification tasks.

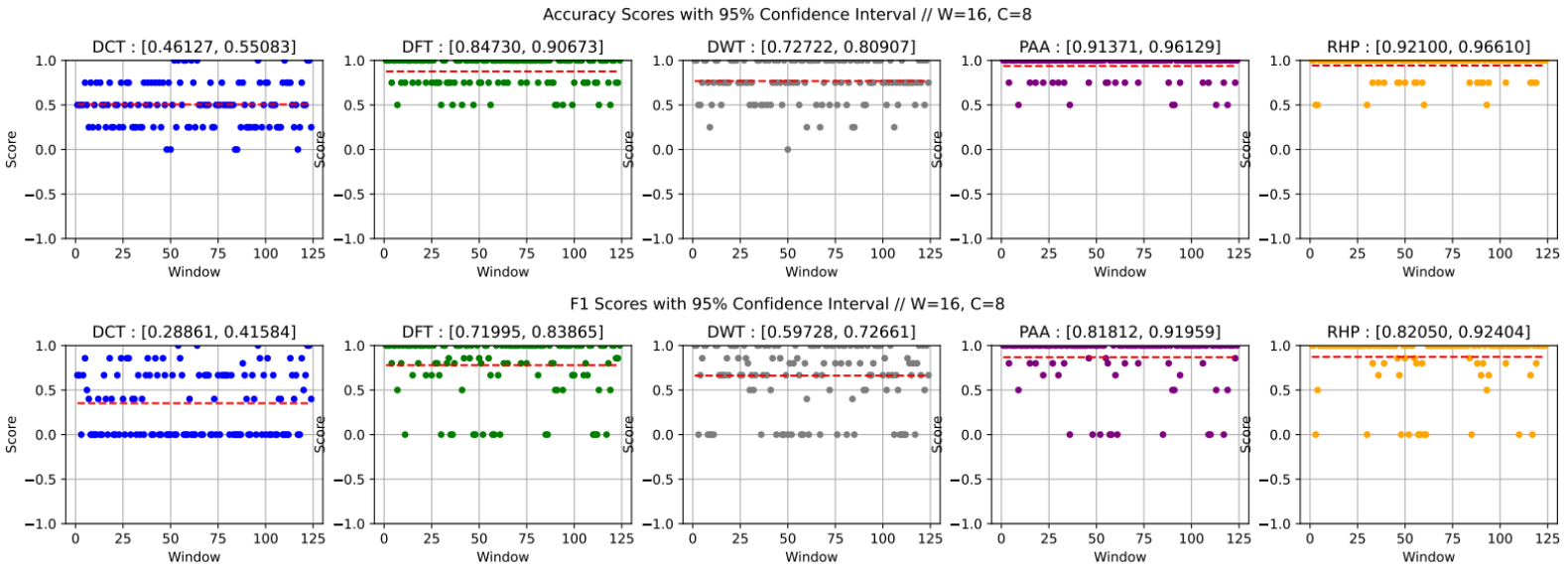

Figure 21. SVM Accuracy and F1 Scores, Pump Sensor Data, Window=16, Compression Ratio= 8.

In terms of Accuracy and F1 scores, for a window size of 16, RHP and PAA exhibit substantially better performance than the other methods, possessing the highest mean values and confidence intervals. Specifically, RHP achieves mean

Accuracy and F1 scores of 0.94354 and 0.87227, respectively, slightly surpassing PAA by 0.6% and 0.3%, where PAA records mean scores of 0.9375 and 0.86885. The confidence intervals corroborate this trend, with RHP displaying lower and upper limits closer to 1, in comparison to the confidence interval limits of PAA for both Accuracy and F1 scores.

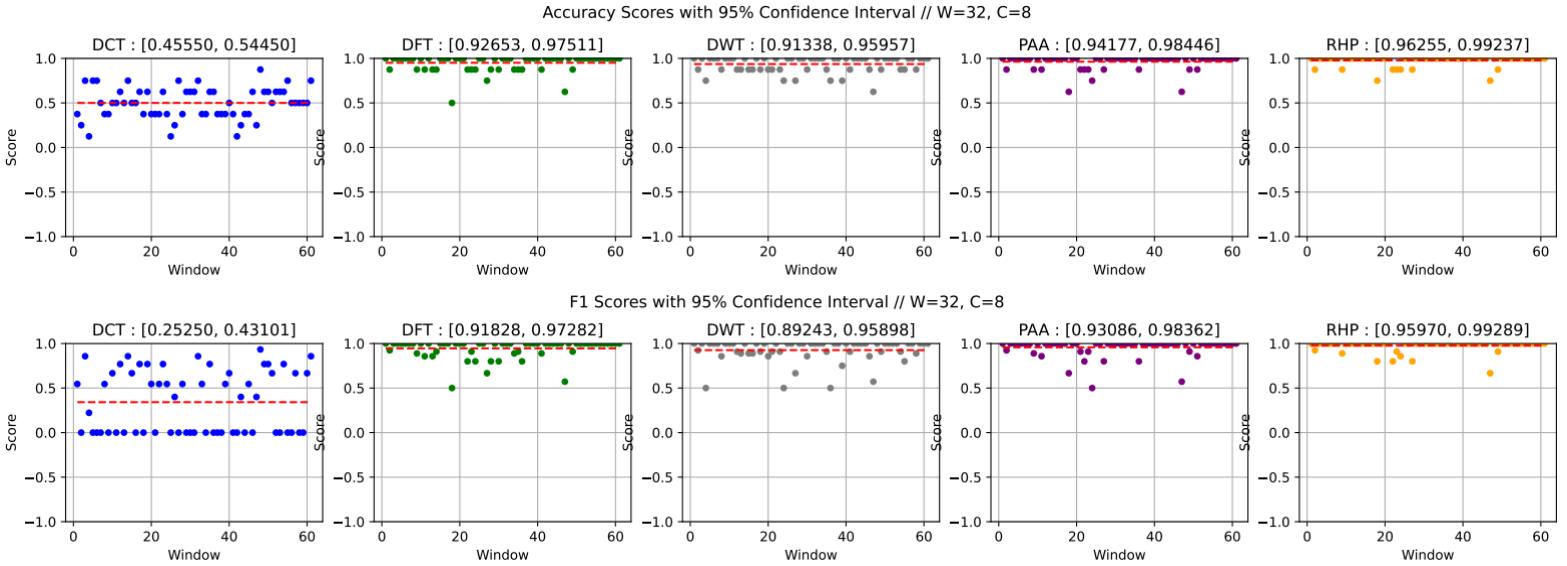

Figure 22. SVM Accuracy and F1 Scores, Pump Sensor Data, Window=32, Compression Ratio= 8.

Moving to a window size of 32, RHP once again distinguishes itself as the best method for both Accuracy and F1 scores, with better mean scores and confidence intervals. RHP achieves an Accuracy confidence interval upper limit of 0.99237 and a lower limit of 0.96255, indicating near-perfect results. The confidence interval for F1 scores also reflects outstanding performance, with a lower limit of 0.95970 and an upper limit of 0.99289, both exceptionally close to the best possible value 1. These results solidify RHP's exceptional performance and reliability for this window size, making it the optimal choice for classification tasks in this scenario. When comparing mean scores, RHP outperforms the other methods by at least 1.4% for Accuracy scores and 2% for F1 scores, with mean values of 0.97745 and 0.97629 respectively.

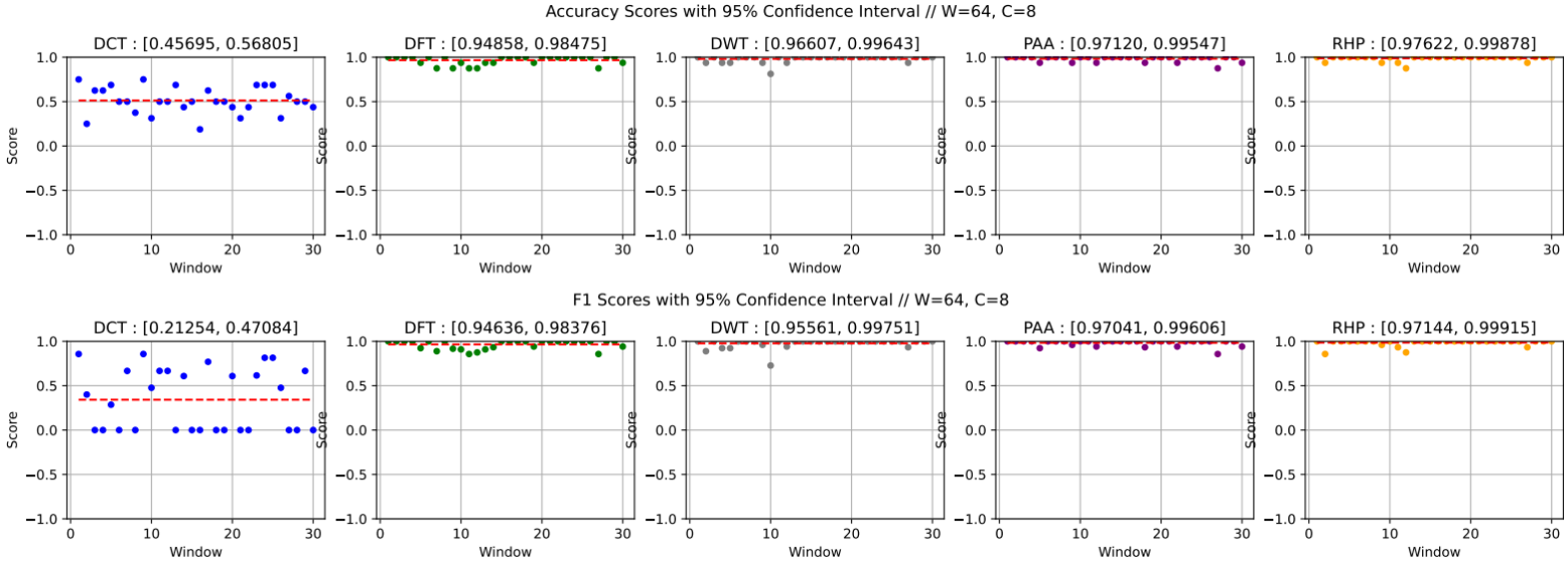

Figure 23. SVM Accuracy and F1 Scores, Pump Sensor Data, Window=64, Compression Ratio= 8.

With an increased window size of 64, RHP demonstrates nearly perfect Accuracy and F1 score results, as evidenced by the confidence intervals. RHP maintains the highest upper and lower limits, underscoring its exceptional performance. Notably, the lower limits of RHP's Accuracy and F1 score confidence intervals indicate near-perfect results even in worst-case scenarios, emphasizing its remarkable consistency and reliability. RHP's mean values of 0.9875 for Accuracy and 0.98529 for F1 scores further confirm its almost perfect performance, marginally surpassing other methods.

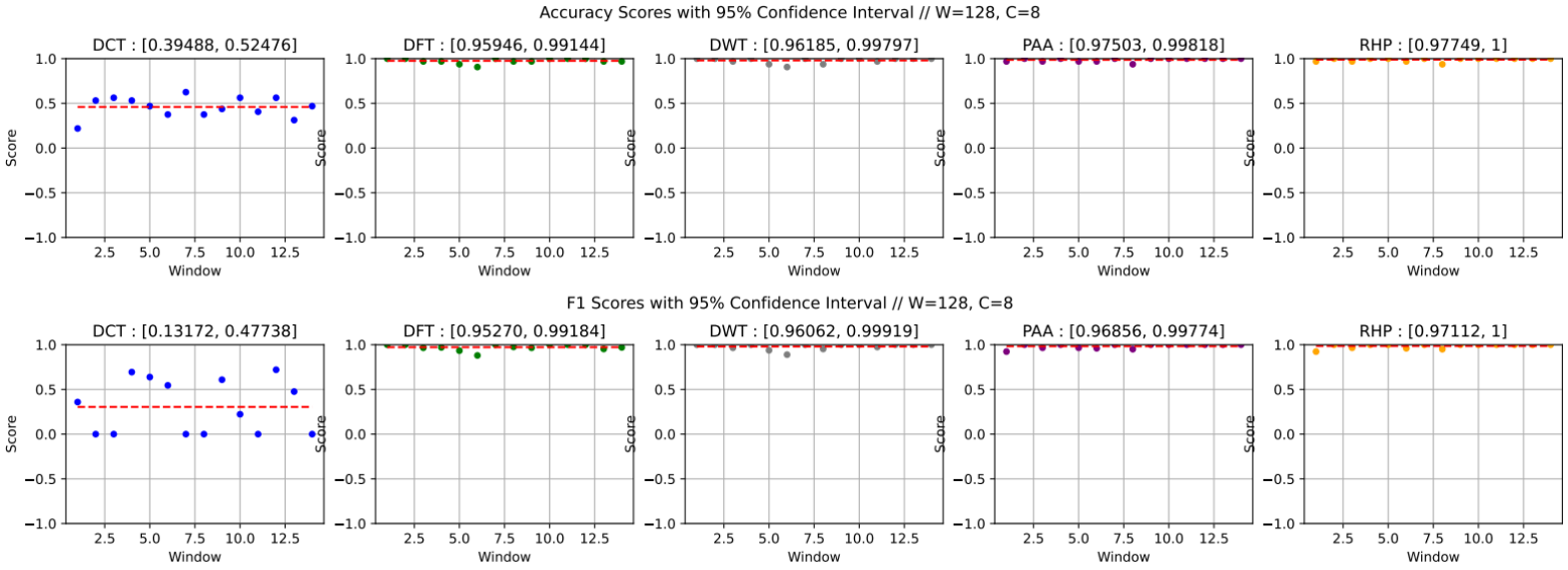

Figure 24. SVM Accuracy and F1 Scores, Pump Sensor Data, Window=128, Compression Ratio= 8.

By further increasing the window to 128 data points, RHP maintains its performance, once again delivering nearly perfect results for both Accuracy and F1 scores. In terms of Confidence Intervals, RHP stands out with lower limits that are closer to 1 than all other methods, coupled with an upper limit of 1, which is the optimal value. In addition, RHP's mean values for both Accuracy and F1 scores are also marginally better, very closely approaching the value of 1.

The SVM-plotted results underscore that RHP's performance exhibits not only consistent superiority, but also remarkable precision, particularly with increasingly larger window sizes. Examination of the confidence intervals for Accuracy and F1 scores reveals a consistently favorable and narrow range of uncertainty. Notably, the lower limits of these intervals indicate a noteworthy proximity to perfection, while the upper limits consistently approach an ideal scenario. The mean score values further emphasize RHP's good performance, consistently nearing 1 for both Accuracy and F1 scores. In this comparison, RHP not only demonstrates superiority, but also showcases a high degree of precision and stability in its performance.

6.2.2 Neural Network Results

Using the same Pump Sensor dataset, but transitioning to Neural Network classification, the next set of plots maintains a fixed window size of 32, while exploring the impact of varying compression ratios—4, 8, and 16.

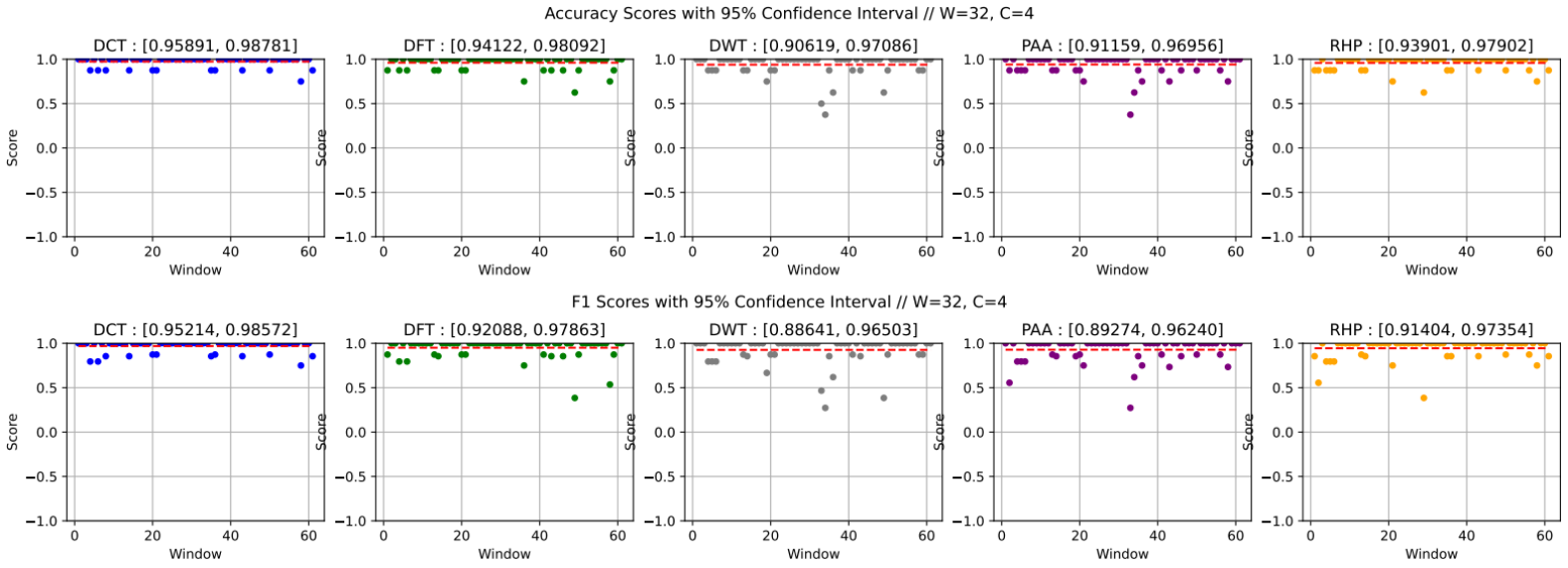

Figure 25. Neural Network Accuracy and F1 Scores, Pump Sensor Data, Window=32, Compression Ratio= 4.

With a compression ratio of 4, all methods demonstrate commendable Accuracy and F1 scores, with DCT, DFT, and RHP emerging as notable performers.

DCT exhibits an advantage, providing marginally more favorable classification results. DCT's upper and lower limits of confidence intervals for both metrics are slightly closer to 1, while its Accuracy score mean outperforms DFT and RHP by 1.2 - 1.5%, and its F1 score mean is 2 - 2.6% superior to the respective means of DFT and RHP.

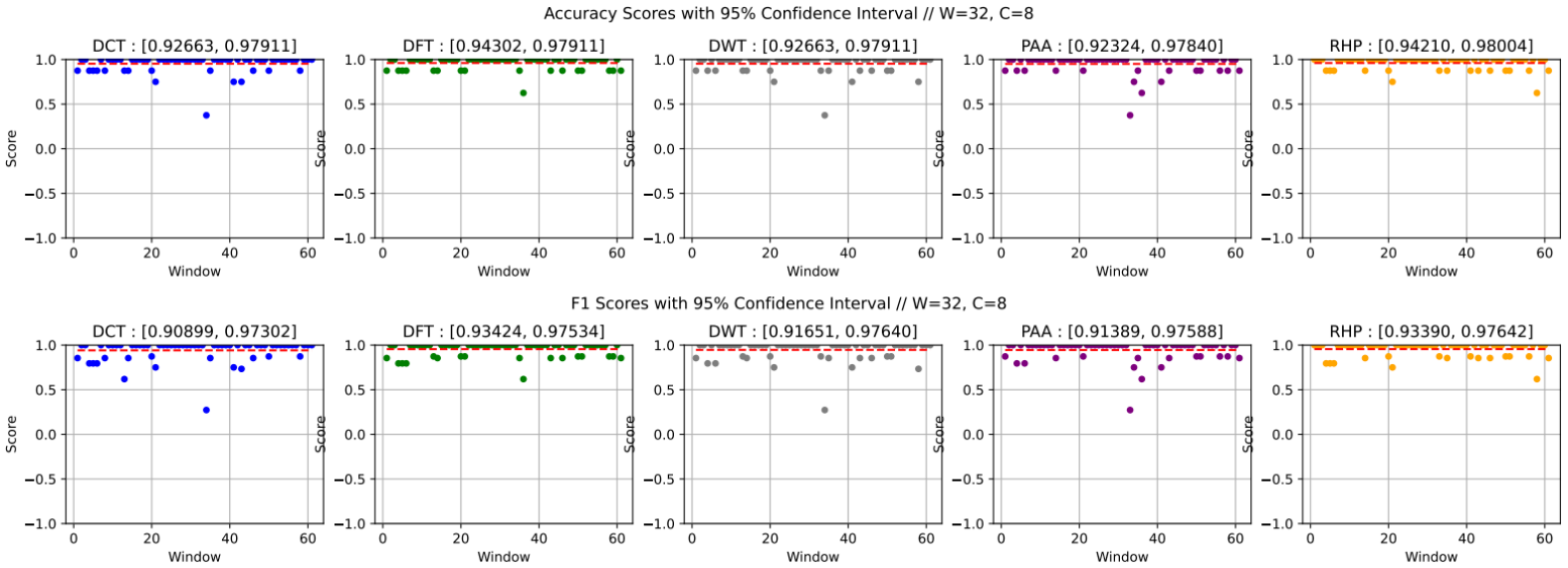

Figure 26. Neural Network Accuracy and F1 Scores, Pump Sensor Data, Window=32, Compression Ratio= 8.

With a compression ratio of 8, the classification DCT shows a slight dip in performance while DFT and RHP emerge as the frontrunners, delivering Accuracy and F1 scores that approach 1 very closely, even with a bigger compression ratio.

Looking at Accuracy and F1 score confidence intervals, RHP demonstrates lower limits slightly below the DFT equivalents, but, contrastingly, the upper limits are marginally closer to 1 compared to the respective DFT upper limits. Comparing mean scores, DFT and RHP share an identical mean Accuracy score equal to 0.96106, surpassing other methods by 1.6 - 1.9%. In terms of F1 score means, RHP exhibits a very slightly superior mean of 0.95515 compared to DFT's 0.95478, along with a 0.9 - 1.5% improvement over the means of other methods.

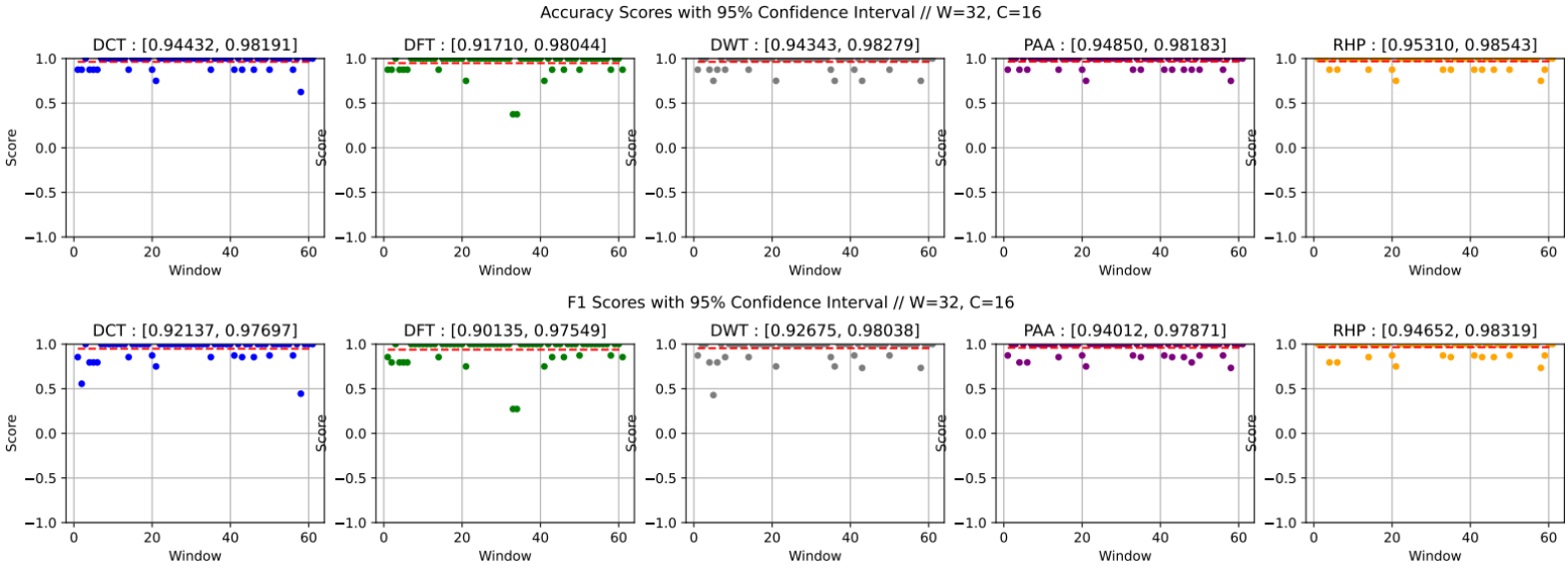

Figure 27. Neural Network Accuracy and F1 Scores, Pump Sensor Data, Window=32, Compression Ratio= 16.

Further increasing the compression ratio to 16, all methods provide competitive results. RHP continues to improve its performance as the compression ratio increases, standing out as the method with the best confidence interval and mean results, for both Accuracy and F1 scores.
Specifically, when examining confidence intervals, RHP distinguishes itself by exhibiting the narrowest range of values. Moreover, the upper limits of these intervals are closer to the optimal value of 1 compared to other methods'. In terms of mean scores, RHP outperforms other methods with a 0.4% to 2.1% higher Accuracy performance, with a mean score of 0.96926. Similarly, in F1 scores, RHP shows a 0.5% to 2.8% improvement over other methods, achieving a mean score of 0.96485.

In summary, as compression ratios vary in neural network classification on compressed data, DCT holds a slight advantage at a compression ratio of 4, displaying marginally better accuracy and F1 scores compared to DFT and RHP. However, at a compression ratio of 8, DCT drops in performance, while DFT and RHP continue to outperform other methods, showcasing near-optimal classification results. Notably, RHP continues to improve even at a compression ratio of 16, standing out for its reliability with the narrowest confidence intervals and the highest mean scores in both accuracy and F1. This highlights the effectiveness of RHP in maintaining performance in neural network-based classification tasks, particularly as compression ratios increase.

6.2.3 KNN Results

After examining SVM and Neural Network classification results and assessing the impact of different methods, window sizes, and compression ratios on classification outcomes, we can also take a preliminary glance at some indicative KNN classification plots.

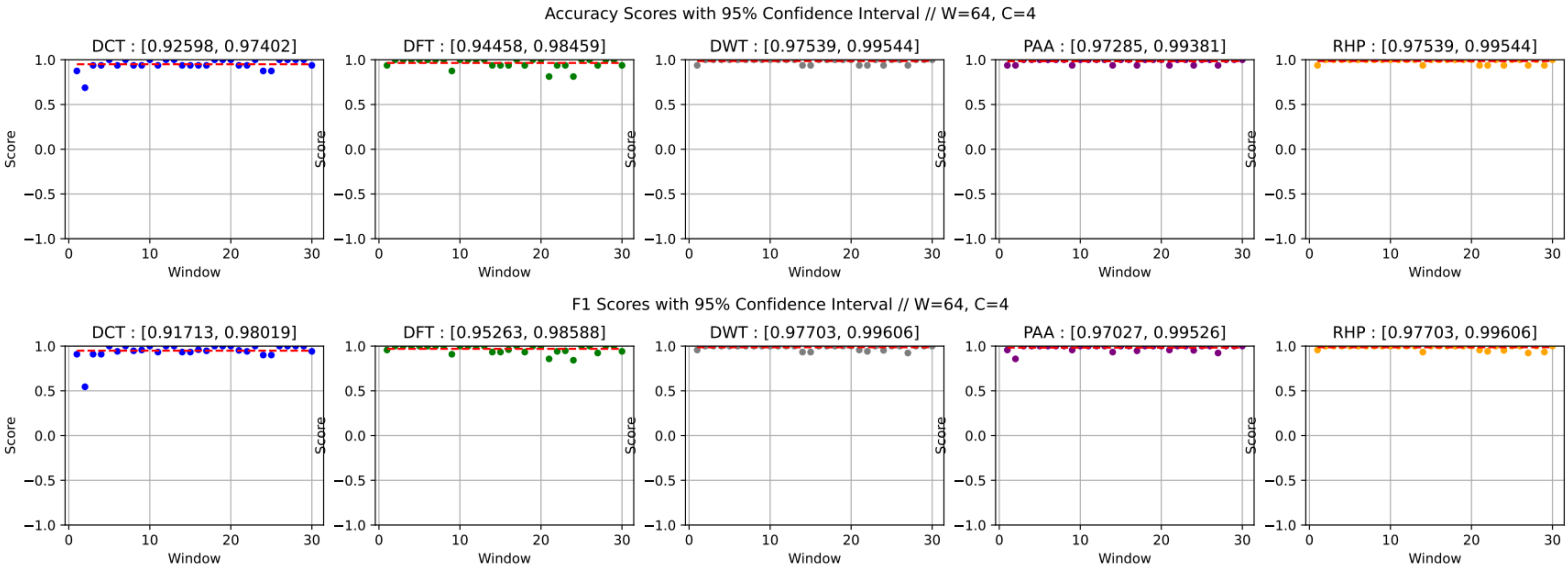

Figure 28. KNN Accuracy and F1 Scores, Pump Sensor Data, Window=64, Compression Ratio= 4.

When using as input windows of 64 points compressed by a ratio equal to 4, all methods provide competitive results. PAA, DWT, and RHP distinguish themselves by showcasing results closer to the optimal value of 1. RHP and DWT have marginally better results than PAA with more narrow confidence intervals and slightly better mean scores. While the DWT and RHP confidence intervals, for both Accuracy and F1 metrics, are the exact same, when looking at the individual score points depicted in the plots, we can see they have slight differences in how they perform the classification task. RHP and DWT also have the same mean scores, showcasing a 0.2 - 3.7% better performance in Accuracy than the other methods, with means equal to 0.98541, and a 0.3 - 4% better performance in F1 scores, with means equal to 0.98654.

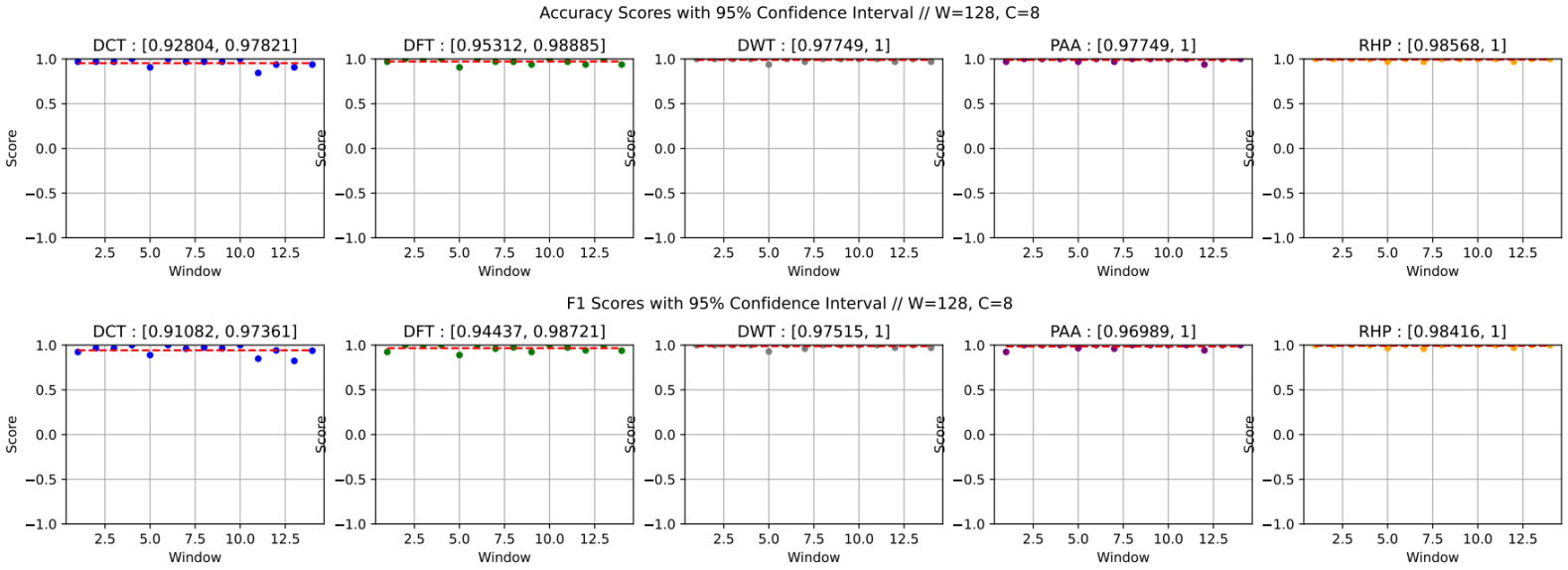

Figure 29. KNN Accuracy and F1 Scores, Pump Sensor Data, Window=128, Compression Ratio= 8.

Increasing the window size to 128 and the compression ratio to 8, all methods maintain competitiveness. Notably, RHP emerges as the top performer, delivering near-perfect results with narrower confidence intervals. RHP achieves mean scores of 0.99330 for Accuracy and 0.99263 for F1, surpassing other methods by 0.4% - 4.2% in Accuracy and 0.7% - 5.3% in F1 scores.

Consistent with the SVM classification, where we progressively increased window sizes, and the Neural Network classification, where we maintained a constant window size while progressively increasing compression ratios, our KNN plots showcase scenarios where both window size and compression ratio were increased concurrently. As with the previous classification methods, when using KNN, RHP consistently retained and enhanced its performance and stability as the data window and compression ratio increased, approaching near-perfect results.

6.3 Linear Regression Results

Transitioning from classification results, we shift our focus to evaluating the performance of the compression techniques in the context of linear regression applications.

In the initial set of three plots, we maintain a constant window size of 16 while gradually increasing the compression ratio, and focus on examining the Root Mean Square Error (RMSE) scores calculated.

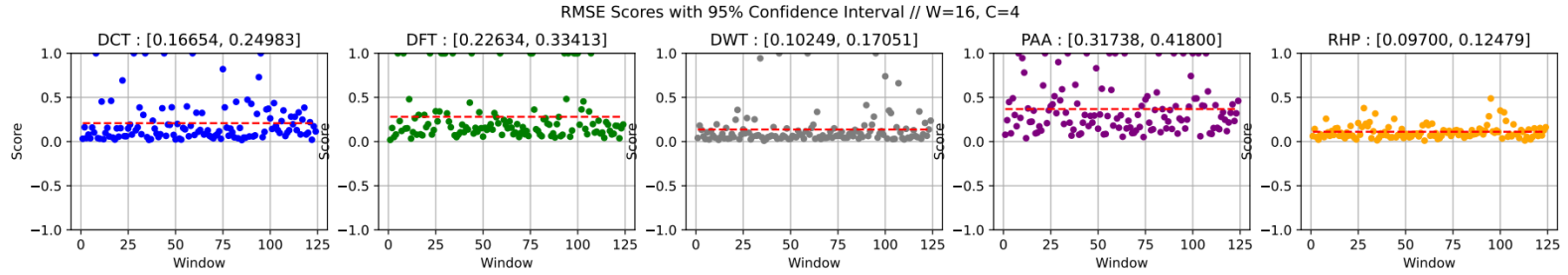

Figure 30. Linear Regression RMSE Scores, Pump Sensor Data, Window=16, Compression Ratio= 4.

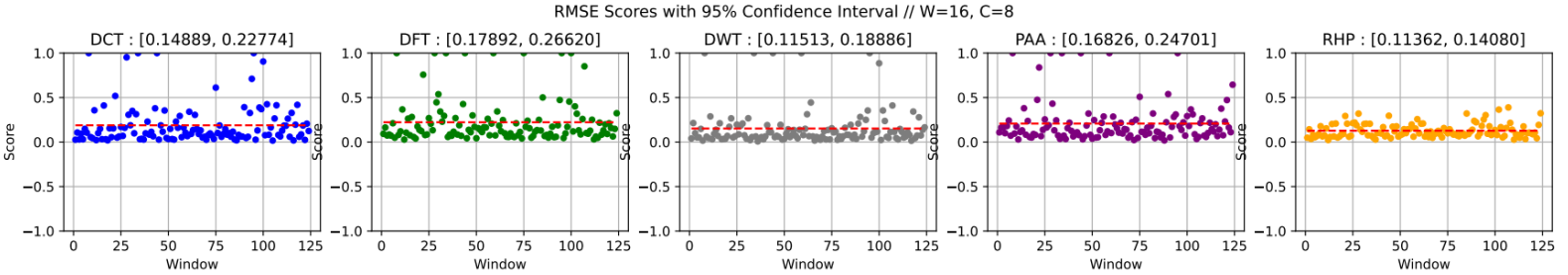

Figure 31. Linear Regression RMSE Scores, Pump Sensor Data, Window=16, Compression Ratio= 8.

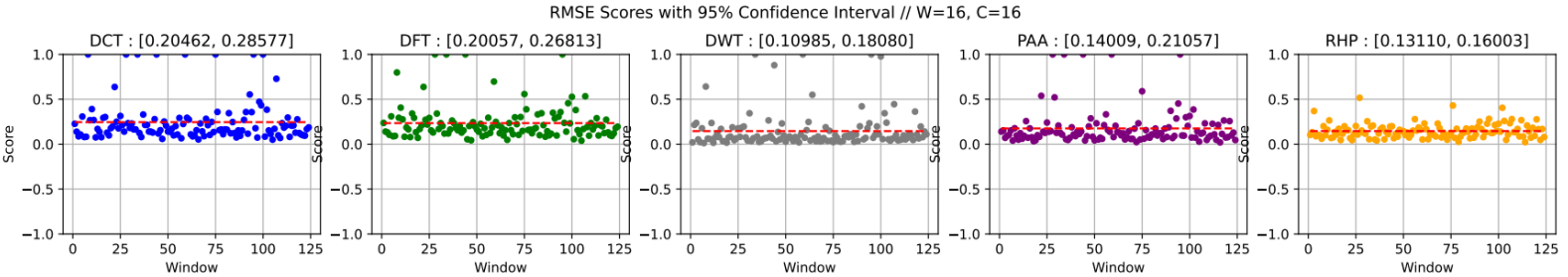

Figure 32. Linear Regression RMSE Scores, Pump Sensor Data, Window=16, Compression Ratio= 16.

Across all three compression ratios (4, 8, and 16), both DWT and RHP consistently exhibit top-tier performance in terms of RMSE scores. RHP takes the lead over DWT, showcasing superior stability and precision, particularly evident in the significantly narrower, and closer to 0, confidence intervals. In the case of a compression ratio of 16, DWT exhibits an upper limit closer to 0 than RHP, albeit with reduced stability.

For a more nuanced comparison, examining the mean of the RMSE scores provides additional insights.

- At a compression ratio of 4, RHP has a mean value of 0.11089, surpassing the other methods by 18.8% - 69.8%.

- For a compression ratio of 8, RHP leads with a mean value of 0.12721, outperforming the other methods by 16.3% - 42.8%.
- At a compression ratio of 16, RHP and DFT share similar means of 0.14557 and 0.14533 respectively, both surpassing the other methods by 17.1% - 40.7%.

In the upcoming series of plots, our focus shifts from the smallest window size tested to the largest, equal to 128 data points.

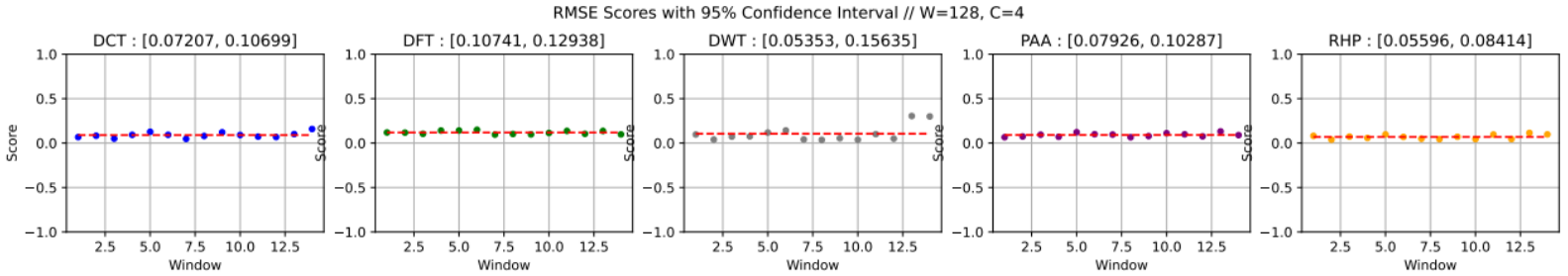

Figure 33. Linear Regression RMSE Scores, Pump Sensor Data, Window=128, Compression Ratio = 4.

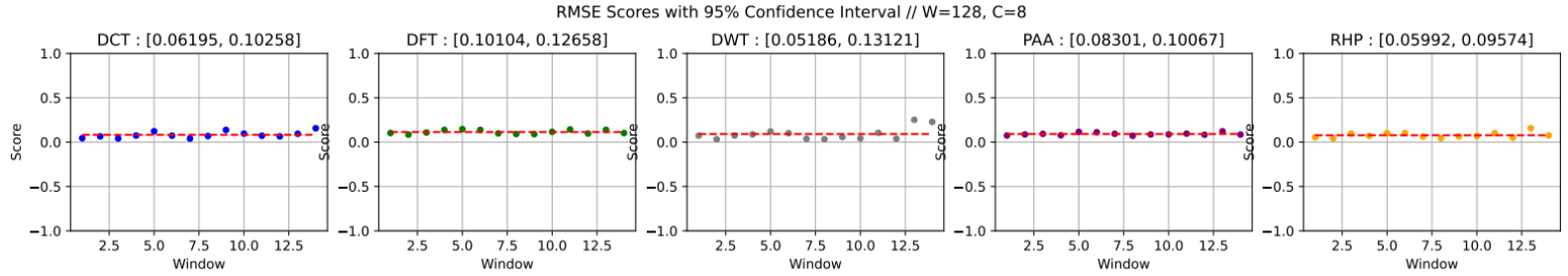

Figure 34. Linear Regression RMSE Scores, Pump Sensor Data, Window=128, Compression Ratio = 8.

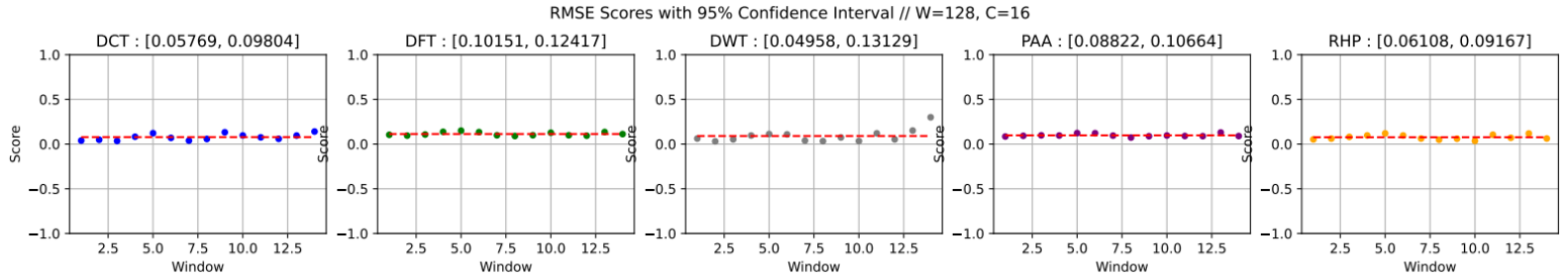

Figure 35. Linear Regression RMSE Scores, Pump Sensor Data, Window=128, Compression Ratio = 16.

In line with the findings for a window size of 16, the results for a window size of 128 also highlight RHP's superiority, showcasing the most favorable

combination of confidence interval values close to 0 and a narrow confidence range across all compression ratios.

Examining the mean of the RMSE scores:

- At a compression ratio of 4, RHP has a mean value equal to 0.07005, surpassing the other methods by 21.7% to 40.8%.
- For a compression ratio of 8, RHP leads with a mean value of 0.07783, outperforming the other methods by 5.4% to 31.6%.
- For a compression ratio of 16, while DCT shows improved performance with a mean equal to 0.07786, RHP retains stability and good performance with a mean value of 0.07637, outperforming the other methods by 2% to 32.3%.

Based on our linear regression observations on the Pump sensor dataset, RHP consistently outperforms other methods across different window sizes and compression scenarios. For both small and large window sizes, RHP shows stability and reliability. Its confidence intervals are narrow, and mean RMSE scores consistently lead and are close to 0. These findings underscore RHP's good performance in a variety of linear regression applications, showcasing its effectiveness in preserving linear relationships within the data.

6.4 Additional Experiments

In the pursuit of a thorough understanding of the effects of compression methods on machine learning tasks, we conducted several additional experiments that were not explicitly covered in the preceding subchapters. This section introduces some of these experiments in a table format, providing insights not previously featured in our systematic showcase.

In the upcoming tables, we present mean-based comparisons, incorporating a specific column that quantifies the percentage difference between the best-performing mean and the next-best-performing mean.

6.4.1 Additional Clustering Results

Dataset	Window Size (W) : Compression Ratio (C)	Algorithm	Metric	Mean					
				DFT	DCT	DWT	PAA	RHP	Min. Deviation Percentage
Intel Lab (Lights)	16 : 8	DBSCAN	Silhouette Scores	0.76022	0.03313	0.71111	0.58655	0.84162	10.7 %
Intel Lab (Lights)	16 : 16	DBSCAN	Silhouette Scores	0.78584	0.02316	0.71182	0	0.87820	11.7 %
Intel Lab (Lights)	32 : 4	K-means	Silhouette Scores	0.76374	0.75285	0.74915	0.76116	0.84502	10.6 %
Intel Lab (Lights)	32 : 4	DBSCAN	Silhouette Scores	0.68322	-0.31317	0.67507	0.71049	0.72700	2.3 %
Intel Lab (Lights)	32 : 8	K-means	Silhouette Scores	0.77035	0.76010	0.74938	0.76761	0.84449	9.6 %
Intel Lab (Lights)	32 : 16	K-means	Silhouette Scores	0.77824	0.76755	0.75038	0.77657	0.84154	8.1 %
Intel Lab (Lights)	32 : 16	DBSCAN	Silhouette Scores	0.74960	-0.30888	0.67441	0.64247	0.83678	11.6 %
Intel Lab (Lights)	64 : 4	DBSCAN	Silhouette Scores	0.54540	-0.30888	0.55787	0.66566	0.81220	22 %
Intel Lab (Lights)	64 : 8	DBSCAN	Silhouette Scores	0.58582	-0.31030	0.55787	0.72106	0.80747	10.7 %
Intel Lab (Lights)	128 : 4	DBSCAN	Silhouette Scores	0.29557	–	0.31721	0.58154	0.70697	21.5 %

Table 1. Additional Clustering Results

Examining the clustering results of the Intel Lab Data's light sensor dataset, reveals some noticeable trends:

- For Silhouette Scores, RHP generally performs well across different configurations, frequently outperforming the next-best method by a double-digit percentage.
- DWT and PAA demonstrate decent performance, but their consistency varies across different configurations.
- DFT performs reasonably well, but its performance drops significantly for larger window sizes.

- DCT shows mixed performance, with negative scores in some cases.
- RHP seems to be the most consistent performer across different configurations.

6.4.2 Additional Classification Results

Dataset	Window Size (W) : Compression Ratio (C)	Algorithm	Metric	Mean					
				DFT	DCT	DWT	PAA	RHP	Deviation Percentage
Pump Sensor Data	16 : 4	SVM	Accuracy Score	0.86088	0.5	0.76209	0.92741	0.94959	2.4%
Pump Sensor Data	16 : 4	SVM	F1 Score	0.76591	0.36543	0.66248	0.84251	0.88210	4.7%
Pump Sensor Data	16 : 4	Neural Network	Accuracy Score	0.93951	0.9375	0.93346	0.92137	0.96169	2.3%
Pump Sensor Data	16 : 4	Neural Network	F1 Score	0.92542	0.92786	0.91685	0.90382	0.95480	2.9%
Pump Sensor Data	32 : 4	SVM	F1 Score	0.93321	0.37185	0.93019	0.96027	0.96865	0.8%
Pump Sensor Data	64 : 4	SVM	Accuracy Score	0.96975	0.5	0.97708	0.9875	0.98958	0.2%
Pump Sensor Data	64 : 4	SVM	F1 Score	0.96825	0.38272	0.97133	0.98715	0.98935	0.2%
Pump Sensor Data	64 : 8	Neural Network	Accuracy Score	0.97708	0.975	0.97291	0.975	0.97916	0.2%
Pump Sensor Data	64 : 8	Neural Network	F1 Score	0.97425	0.97192	0.96960	0.97162	0.97643	0.2%
Pump Sensor Data	128 : 4	SVM	Accuracy Score	0.97098	0.44642	0.97991	0.98660	0.99330	0.6%
Pump Sensor Data	128 : 4	SVM	F1 Score	0.96767	0.37877	0.97990	0.98188	0.99293	1.1%
Pump Sensor Data	128 : 4	Neural Network	Accuracy Score	0.97991	0.97321	0.97544	0.97991	0.98660	0.6%
Pump Sensor Data	128 : 4	Neural Network	F1 Score	0.97830	0.97143	0.97313	0.97833	0.98506	0.6%

Table 2. Additional Classification Results

Further examining the Pump sensor dataset classification results:

- RHP emerges as a notably robust and consistent method, consistently surpassing other methods and achieving high Silhouette Scores across diverse configurations.

- PAA, DWT, and DFT show good Silhouette Scores across different configurations.

- DCT generally shows lower Silhouette Scores compared to other methods.

In evaluating the performance of various classification methods, it becomes evident that there aren't any Logistic Regression results since RHP faces limitations when applied to Logistic Regression. The intrinsic non-linearity introduced by the logistic function poses a challenge for the linear decision boundaries created by RHP. Logistic Regression's sigmoid-shaped curve enables it to model complex relationships inherent in the data, but the linear projections onto hyperplanes struggle to capture these intricate patterns effectively. As a result, the classification accuracy of RHP tends to be surpassed in Logistic Regression applications by the other methods.

6.4.3 Additional Linear Regression Results

Dataset	Window Size (W) : Compression Ratio (C)	Algorithm	Metric	Mean					
				DFT	DCT	DWT	PAA	RHP	Deviation Percentage
Pump Sensor Data	16 : 4	Linear Regression	R-Squared Scores	0.17887	0.30291	0.53689	-0.1863	0.5804	7.9%
Pump Sensor Data	16 : 8	Linear Regression	R-Squared Scores	0.25796	0.33948	0.48969	0.29448	0.52287	6.3%
Pump Sensor Data	32 : 4	Linear Regression	R-Squared Scores	0.33641	0.55731	0.62706	0.45277	0.71883	14.6%
Pump Sensor Data	32 : 4	Linear Regression	RMSE Scores	0.24248	0.13661	0.12753	0.16678	0.11297	11.5%

Pump Sensor Data	32 : 8	Linear Regression	R-Squared Scores	0.44597	0.57045	0.63121	0.62876	0.64279	2.2%
Pump Sensor Data	32 : 16	Linear Regression	R-Squared Scores	0.57751	0.59788	0.45927	0.65278	0.66424	1.7%
Pump Sensor Data	128 : 4	Linear Regression	R-Squared Scores	0.76086	0.85227	0.6833	0.85416	0.90816	6.3%
Pump Sensor Data	128 : 8	Linear Regression	R-Squared Scores	0.77631	0.86795	0.7802	0.85492	0.88192	1.6%
Pump Sensor Data	128 : 16	Linear Regression	R-Squared Scores	0.7816	0.87992	0.78585	0.83697	0.89069	1.2%

Table 3. Additional Linear Regression Results

Further examining the Pump sensor dataset linear regression results, besides its good RMSE score performance, RHP appears to also show the most consistent and competitive R-Squared score-based performance across different configurations, especially improving for a larger window size.

6.5 TOSSIM Simulation Results

The TOSSIM simulations were conducted with the following parameters:

- To match our dataset sensors, 48 sensors were divided into 8 groups of 6 sensors each.
- Each sensor group transmits a window of data, comprising of 16 data points, we use the IntelLabData Temperatures dataset for this application.
- Total dataset size: 624 data points.
- Each group transmits its data to a respective Cluster Head.
- Cluster Heads forward the data to a base station.

As discussed in Chapter 4, after the dataset transmission process is completed we compare the bits required to transmit the entire dataset. We take

measurements for both uncompressed and compressed data using compression ratios of 4 and 8. Then, we estimate the power consumption, considering how compression affects energy use. This analysis helps us understand the efficiency and impact our data compression has on the network. We perform 3 simulation runs for each case and do our calculations based on the average of bits measured across those runs.

Our energy consumption calculation is based on the following formulas[77]. For the energy cost of data transmission, we used $(E_{TX} + E_{RF} \times dist^2) \times b$, while for the energy cost of receiving data we used $E_{RX} \times b.$

- b represents the bits transferred.
- dist is the distance between the nodes, we set dist to be 10 meters.
- E_{TX} represents the per-bit power dissipation of the transmitter electronics and its value is set to 50nJ/bit, E_{RX}. is similarly set to a value of 50nJ/bit[49].
- E_{RF} represents the per-bit and squared distance power delivered by the power amplifier and is set to 100pJ/bit/m^2 [49].

In the following section, we present three tables containing the results of our simulations. Each table corresponds to one of the three different scenarios (uncompressed data, data compressed by a factor of 4, and data compressed by a factor of 8).

Metrics (No compression - 119.808 bytes transmitted)	Bytes (Average over 3 runs)
Cluster to Cluster Head Transmission	142.933,3
Cluster to Cluster Head NACKs bytes	23.125,3
Cluster Head to Base Station Transmission	147.626,6
Cluster Head to Base Station NACKs bytes	27.840
Average Total Bytes due to Data Transmission	290.560
Amount of Average Total Bytes due to NACKs	50.965,3

Table 4. Byte Metrics for Uncompressed Dataset Transmission.

Based on the above measurements and our formulas, the total energy required for the transmission process amounted to 0.25569 Joules. Within this total, 0.13947 Joules were expended for sending data, and 0.11622 Joules were dedicated to receiving the transmitted data.

Metrics (Compression Ratio 4 - 29.952 bytes transmitted)	Bytes (Average over 3 runs)
Cluster to Cluster Head Transmission	34.218,6
Cluster to Cluster Head NACKs bytes	4.266,6
Cluster Head to Base Station Transmission	34.960
Cluster Head to Base Station NACKs bytes	5.008
Average Total Bytes due to Data Transmission	69.178,6
Amount of Average Total Bytes due to NACKs	9.274,6

Table 5. Byte Metrics for Dataset Transmission with Compression Ratio 4.

When using a compression ratio of 4 the required energy cost to transmit the same dataset is reduced to 0.06088 Joules, of which 0.03321 Joules were expended for sending data, and 0.02767 Joules were dedicated to receiving the transmitted data.

Metrics ***(Compression Ratio 8 - 14.976 bytes transmitted)***	***Bytes*** ***(Average over 3 runs)***
Cluster to Cluster Head Transmission	16.904
Cluster to Cluster Head NACKs bytes	1.928
Cluster Head to Base Station Transmission	17.344
Cluster Head to Base Station NACKs bytes	2.368
Average Total Bytes due to Data Transmission	34.248
Amount of Average Total Bytes due to NACKs	4,296

Table 6. Byte Metrics for Dataset Transmission with Compression Ratio 8.

When using a compression ratio of 8 the required energy cost to transmit the dataset is further reduced to 0.03014 Joules, of which 0.01644 Joules were expended for sending data, and 0.01370 Joules were dedicated to receiving the transmitted data.

The transmitted byte reductions are visually represented in Figure 12, where the energy consumption for sending data and receiving transmitted data is depicted.

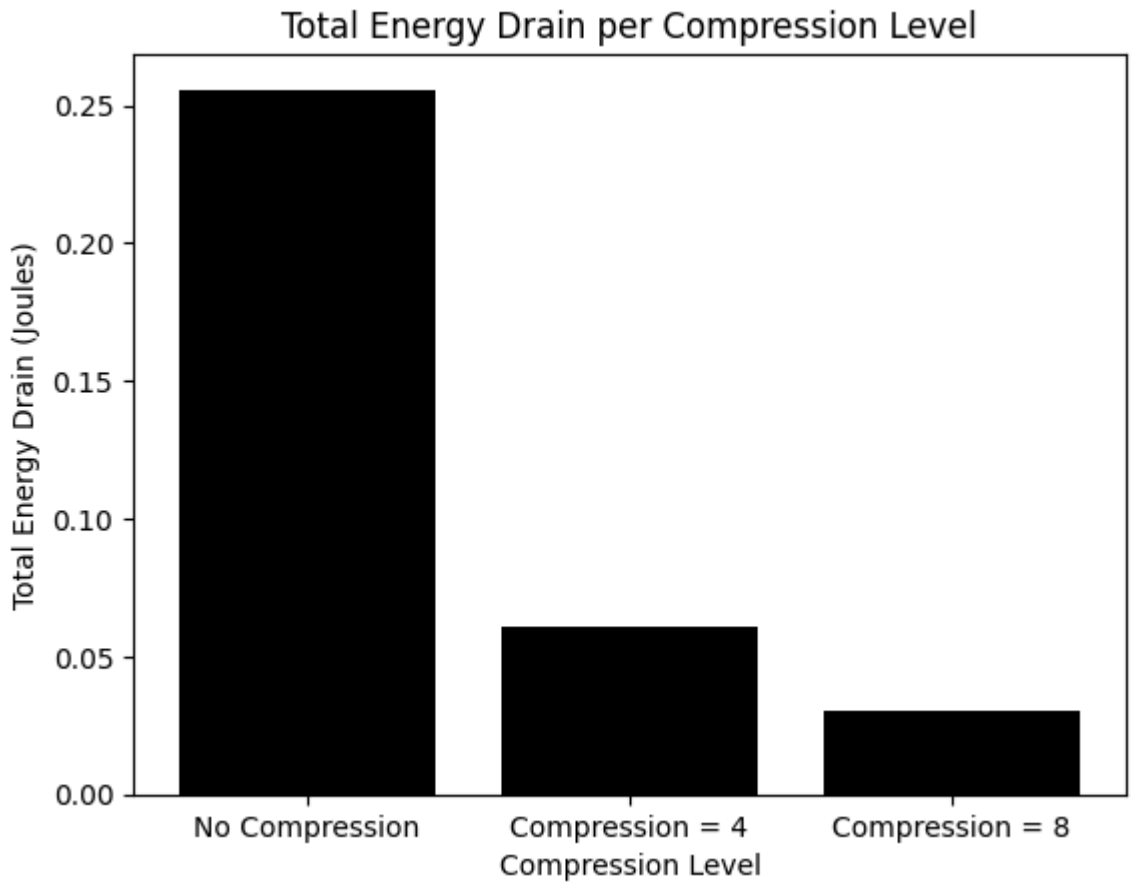

Figure 36. A graph visually showcasing the relation between the Compression Ratio and the Total Energy Drain.

Based on our measurements we can deduce that the utilization of data compression has a direct effect on the transmitted bites and the related energy consumption, causing a reduction proportional to the compression ratio. Interestingly, these reductions surpass the anticipated proportional decrease, signifying an added efficiency. The decrease in transmitted packets not only results in smaller data sizes but also reduces the occurrence of negative acknowledgments (NACKs). This highlights the ability of data compression to optimize payload size and cut down on the need for corrective transmissions, ultimately making the communication process more efficient.

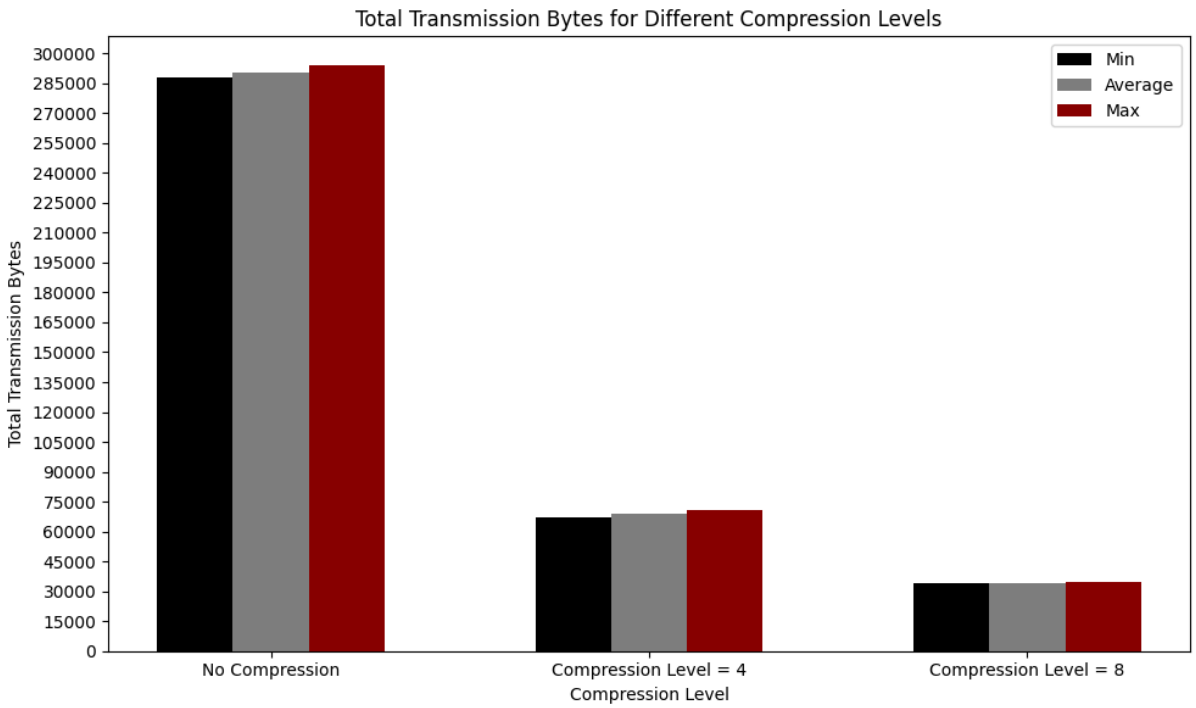

Figure 37. Visual representation of the total bytes required to transmit the dataset when using different compression levels.

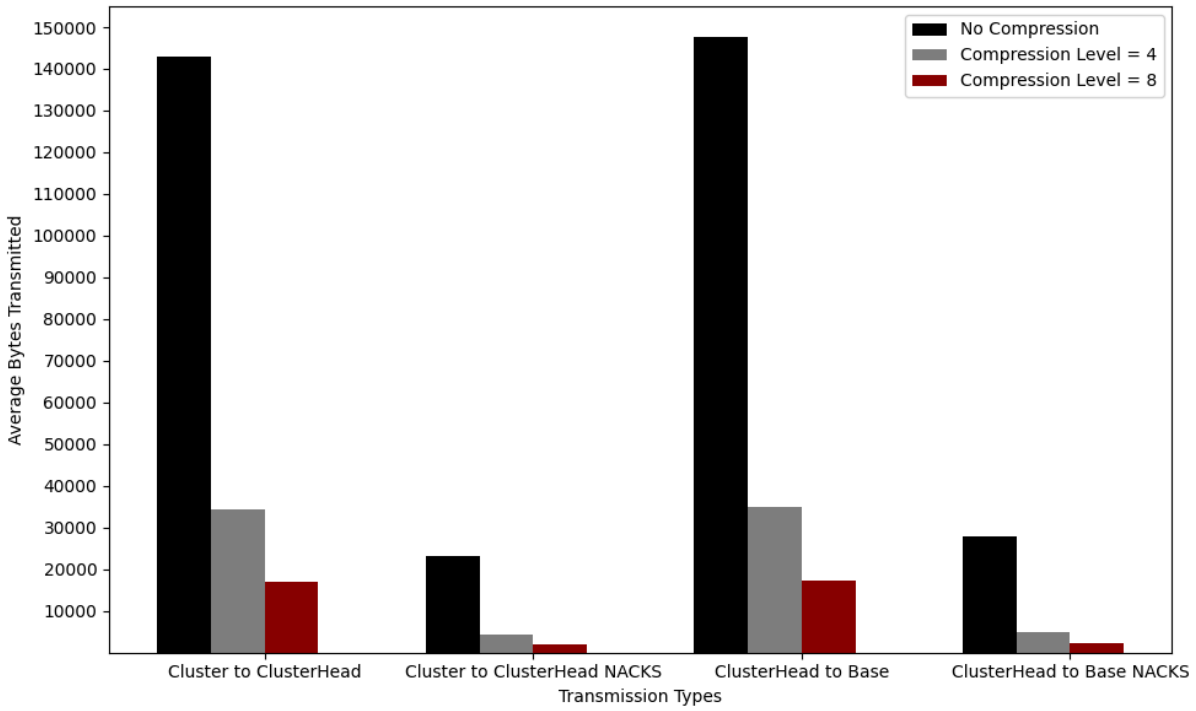

Figure 38. Visual representation of Average bytes Transmitted per Transmission Type and Compression Ratio.

6.6 Future Work and Considerations

While our present study offers valuable insights into the performance of compression methods across diverse applications, there are opportunities for future research that could enhance the robustness and applicability of our findings. The following considerations point to potential directions for further exploration and improvement:

- **Diverse Datasets and Data Varieties:** Broadening the scope of our experiments to include a wider range of datasets with diverse characteristics can offer a more comprehensive understanding of the performance of compression methods. Evaluating datasets with varying sizes, structures, and complexities would allow for a more nuanced assessment, ensuring the generalizability of our conclusions.

- **Additional Algorithms and Tuning:** Testing with varied parameters and modifications, such as using different wavelets in the case of DWT or using Symbolic Aggregate Approximation instead of PAA, and a diverse set of algorithms can unveil nuanced aspects unique to each method. This process aids in identifying scenarios where specific methods may excel, providing valuable insights for tailored applications.

- **Larger and Diverse Data Windows, Varied Compression Ratios:** Evaluating performance with larger data windows could offer insights into scalability, uncovering potential limitations or advantages not apparent with smaller datasets. At the same time, using a wider range of compression ratios can enhance our understanding of trade-offs in practical applications that demand larger levels of data compression.

- **Consideration of Computational Costs:** Analyzing the computational costs associated with each compression method could enhance our conclusions, proving crucial for understanding the practical applicability of different methods, especially in real-world scenarios.

BIBLIOGRAPHY

[1] "The Family of Fourier Transform." Accessed: Nov. 30, 2023. [Online]. Available: https://www.dspguide.com/ch8/1.htm

[2] "Discrete Wavelet Transform - an overview | ScienceDirect Topics." Accessed: Nov. 30, 2023. [Online]. Available: https://www.sciencedirect.com/topics/mathematics/discrete-wavelet-transform

[3] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," *IEEE Trans. Comput.*, vol. C–23, no. 1, pp. 90–93, Jan. 1974, doi: 10.1109/T-C.1974.223784.

[4] "Piecewise Aggregate Approximation." Accessed: Oct. 31, 2023. [Online]. Available: https://vigne.sh/posts/piecewise-aggregate-approx/

[5] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, vol. 5.1, University of California Press, 1967, pp. 281–298. Accessed: Dec. 12, 2023. [Online]. Available: https://projecteuclid.org/ebooks/berkeley-symposium-on-mathematical-statistics-and-probability/Proceedings-of-the-Fifth-Berkeley-Symposium-on-Mathematical-Statistics-and/chapter/Some-methods-for-classification-and-analysis-of-multivariate-observations/bsmsp/1200512992

[6] M. Ester, H.-P. Kriegel, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise".

[7] X. Yan, *Linear Regression Analysis: Theory and Computing*. World Scientific, 2009.

[8] J. S. Cramer, "The Origins of Logistic Regression." Rochester, NY, Dec. 01, 2002. doi: 10.2139/ssrn.360300.

[9] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967, doi: 10.1109/TIT.1967.1053964.

[10] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[11] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities.," *Proc. Natl. Acad. Sci.*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982, doi: 10.1073/pnas.79.8.2554.

[12] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, in SenSys '03. New York, NY, USA: Association for Computing Machinery, Nov. 2003, pp. 126–137. doi: 10.1145/958491.958506.

[13] S. S. Vempala, *The Random Projection Method*. American Mathematical Soc., 2005.

[14] "Intel Lab Data." Accessed: Dec. 12, 2023. [Online]. Available: https://db.csail.mit.edu/labdata/labdata.html

[15] "pump_sensor_data." Accessed: Nov. 30, 2023. [Online]. Available: https://www.kaggle.com/datasets/nphantawee/pump-sensor-data

[16] M. A. M. Y. Alsayyh, D. Mohamad, T. Saba, A. Rehman, and J. S. AlGhamdi, "A novel fused image compression technique using DFT, DWT, and DCT," *J. Inf. Hiding Multimed. Signal Process.*, vol. 8, pp. 261–271, Jan. 2017.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, fourth edition*. MIT Press, 2022.

[18] D. Rafiei and A. Mendelzon, "Similarity-Based Queries for Time Series Data," *ACM SIGMOD Rec.*, vol. 26, Apr. 1997, doi: 10.1145/253260.253264.

[19] J. O. Smith, *Mathematics of the Discrete Fourier Transform (DFT): With Audio Applications*. Julius Smith, 2008.

[20] L. P. Yaroslavsky, “Local adaptive image restoration and enhancement with the use of DFT and DCT in a running window,” in *Wavelet Applications in Signal and Image Processing IV*, SPIE, Oct. 1996, pp. 2–13. doi: 10.1117/12.255218.
[21] M. Hafez, T. Khattab, and H. Arslan, “DFT-Based Multi-Directions Directional Modulation,” *IEEE Wirel. Commun. Lett.*, vol. 8, no. 4, pp. 1232–1235, Aug. 2019, doi: 10.1109/LWC.2019.2912597.
[22] K. J. Lee and M. H. Lee, “FSK demodulation method using the short-time DFT of power line carrier channels,” in *IEEE/AFCEA EUROCOMM 2000. Information Systems for Enhanced Public Safety and Security (Cat. No.00EX405)*, May 2000, pp. 301–307. doi: 10.1109/EURCOM.2000.874821.
[23] “Channel Equalization - an overview | ScienceDirect Topics.” Accessed: Oct. 23, 2023. [Online]. Available: https://www.sciencedirect.com/topics/engineering/channel-equalization
[24] H. Cho, T. T. Zin, N. Shinkawa, and R. Nishii, “Post-Mortem Human Identification Using Chest X-Ray and CT Scan Images,” *Int. J. Biomed. Soft Comput. Hum. Sci. Off. J. Biomed. Fuzzy Syst. Assoc.*, vol. 23, no. 2, pp. 51–57, 2018, doi: 10.24466/ijbschs.23.2_51.
[25] J. W. Brault and O. R. White, “The Analysis and Restoration of Astronomical Data via the Fast Fourier Transform,” *Astron. Astrophys.*, vol. 13, p. 169, Jul. 1971, Accessed: Oct. 23, 2023. [Online]. Available: https://ui.adsabs.harvard.edu/abs/1971A&A....13..169B
[26] “The Power of the Fourier Transform for Spectroscopists,” Chemistry LibreTexts. Accessed: Oct. 23, 2023. [Online]. Available: https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Spectroscopy/Fundamentals_of_Spectroscopy/The_Power_of_the_Fourier_Transform_for_Spectroscopists
[27] E. A. Taypicahuana Loza, A. C. Nieto, and S. G. Huaman Bustamante, “Seismic Motion Detection and Classification Methodology for Buildings Using DFT and SVM,” in *2023 Argentine Conference on Electronics (CAE)*, Mar. 2023, pp. 46–51. doi: 10.1109/CAE56623.2023.10087012.
[28] T. Q. Nguyen, “A Data-Driven Approach to Structural Health Monitoring of Bridge Structures Based on the Discrete Model and FFT-Deep Learning,” *J. Vib. Eng. Technol.*, vol. 9, no. 8, pp. 1959–1981, Nov. 2021, doi: 10.1007/s42417-021-00343-5.
[29] C. S. H. Kaushik, T. Gautam, and V. Elamaran, “A tutorial review on discrete fourier transform with data compression application,” in *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, Mar. 2014, pp. 1–6. doi: 10.1109/ICGCCEE.2014.6922210.
[30] F. Mörchen, “Time series feature extraction for data mining using DWT and DFT,” Dec. 2003.
[31] S. Kumari and S. K. Mitra, “Human Action Recognition Using DFT,” in *2011 Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics*, Dec. 2011, pp. 239–242. doi: 10.1109/NCVPRIPG.2011.58.
[32] M. F. Wahab, F. Gritti, and T. C. O’Haver, “Discrete Fourier transform techniques for noise reduction and digital enhancement of analytical signals,” *TrAC Trends Anal. Chem.*, vol. 143, p. 116354, Oct. 2021, doi: 10.1016/j.trac.2021.116354.
[33] “Discrete cosine transform,” *Wikipedia*. Oct. 20, 2023. Accessed: Oct. 23, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Discrete_cosine_transform&oldid=1181096736
[34] N. Roma and L. Sousa, “A tutorial overview on the properties of the discrete cosine transform for encoded image and video processing,” *Signal Process.*, vol. 91, no. 11, pp. 2443–2464, Nov. 2011, doi: 10.1016/j.sigpro.2011.04.015.
[35] A. Shawahna, M. E. Haque, and A. Amin, “JPEG Image Compression using the Discrete Cosine Transform: An Overview, Applications, and Hardware Implementation.” arXiv, Nov. 01, 2019. doi: 10.48550/arXiv.1912.10789.

[36] H.-C. Hsu, K.-B. Lee, N. Y.-C. Chang, and T.-S. Chang, “Architecture Design of Shape-Adaptive Discrete Cosine Transform and Its Inverse for MPEG-4 Video Coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 3, pp. 375–386, Mar. 2008, doi: 10.1109/TCSVT.2008.918275.
[37] H. Ochoa-Dominguez and K. R. Rao, *Discrete Cosine Transform, Second Edition*. CRC Press, 2019.
[38] H. T. Chang and C. L. Tsan, “Image watermarking by use of digital holography embedded in the discrete-cosine-transform domain,” *Appl. Opt.*, vol. 44, no. 29, pp. 6211–6219, Oct. 2005, doi: 10.1364/AO.44.006211.
[39] A. Alahmadi, M. Hussain, H. Aboalsamh, G. Muhammad, G. Bebis, and H. Mathkour, “Passive detection of image forgery using DCT and local binary pattern,” *Signal Image Video Process.*, vol. 11, no. 1, pp. 81–88, Jan. 2017, doi: 10.1007/s11760-016-0899-0.
[40] C. O. S. Sorzano, J. Vargas, and A. P. Montano, “A survey of dimensionality reduction techniques.” arXiv, Mar. 12, 2014. doi: 10.48550/arXiv.1403.2877.
[41] I. Daubechies, “The wavelet transform, time-frequency localization and signal analysis,” *IEEE Trans. Inf. Theory*, vol. 36, no. 5, pp. 961–1005, Sep. 1990, doi: 10.1109/18.57199.
[42] R. S. Stanković and B. J. Falkowski, “The Haar wavelet transform: its status and achievements,” *Comput. Electr. Eng.*, vol. 29, no. 1, pp. 25–44, Jan. 2003, doi: 10.1016/S0045-7906(01)00011-8.
[43] A. C. H. Rowe and P. C. Abbott, “Daubechies wavelets and Mathematica,” *Comput. Phys.*, vol. 9, no. 6, pp. 635–648, Nov. 1995, doi: 10.1063/1.168556.
[44] D. Wei, A. C. Bovik, and B. L. Evans, “Generalized coiflets: a new family of orthonormal wavelets,” in *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems and Computers (Cat. No.97CB36136)*, Nov. 1997, pp. 1259–1263 vol.2. doi: 10.1109/ACSSC.1997.679106.
[45] M. Chavan, N. Mastorakis, M. Chavan, and M. Gaikwad, “Implementation of SYMLET wavelets to removal of Gaussian additive noise from speech signal,” Jan. 2011.
[46] S.-D. Lu, H.-W. Sian, M.-H. Wang, and R.-M. Liao, “Application of Extension Neural Network with Discrete Wavelet Transform and Parseval’s Theorem for Power Quality Analysis,” *Appl. Sci.*, vol. 9, no. 11, Art. no. 11, Jan. 2019, doi: 10.3390/app9112228.
[47] N. E. Miner, “An Introduction to Wavelet Theory and Analysis,” Oct. 1998, pp. SAND98-2265, 1896. doi: 10.2172/1896.
[48] J.-M. Lina, “Image Processing with Complex Daubechies Wavelets,” *J. Math. Imaging Vis.*, vol. 7, no. 3, pp. 211–223, Jun. 1997, doi: 10.1023/A:1008274210946.
[49] M. A. Ali and P. M. Shemi, “An improved method of audio denoising based on wavelet transform,” in *2015 International Conference on Power, Instrumentation, Control and Computing (PICC)*, Dec. 2015, pp. 1–6. doi: 10.1109/PICC.2015.7455802.
[50] E. Foufoula-Georgiou and P. Kumar, *Wavelets in Geophysics*. Academic Press, 1994.
[51] National Technical University of Ukraine " *et al.*, “Procedure for Processing Biometric Parameters Based on Wavelet Transformations,” *Int. J. Mod. Educ. Comput. Sci.*, vol. 13, no. 2, pp. 11–22, Apr. 2021, doi: 10.5815/ijmecs.2021.02.02.
[52] T. Y. Gan, A. K. Gobena, and Q. Wang, “Precipitation of southwestern Canada: Wavelet, scaling, multifractal analysis, and teleconnection to climate anomalies,” *J. Geophys. Res. Atmospheres*, vol. 112, no. D10, 2007, doi: 10.1029/2006JD007157.
[53] W. Lu and A. A. Ghorbani, “Network Anomaly Detection Based on Wavelet Analysis,” *EURASIP J. Adv. Signal Process.*, vol. 2009, no. 1, p. 837601, Dec. 2008, doi: 10.1155/2009/837601.
[54] C. Capilla, “Application of the Haar wavelet transform to detect microseismic signal arrivals,” *J. Appl. Geophys.*, vol. 59, no. 1, pp. 36–46, May 2006, doi: 10.1016/j.jappgeo.2005.07.005.
[55] X. Mi, H. Ren, Z. Ouyang, W. Wei, and K. Ma, “The use of the Mexican Hat and the Morlet wavelets for detection of ecological patterns,” *Plant Ecol.*, vol. 179, no. 1, pp. 1–19, Jul. 2005, doi: 10.1007/s11258-004-5089-4.
[56] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, “Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases,” *Knowl. Inf. Syst.*, vol. 3, no. 3,

pp. 263–286, Aug. 2001, doi: 10.1007/PL00011669.
[57] R. C. Brasileiro, V. L. F. Souza, and A. L. I. Oliveira, “Automatic trading method based on piecewise aggregate approximation and multi-swarm of improved self-adaptive particle swarm optimization with validation,” *Decis. Support Syst.*, vol. 104, pp. 79–91, Dec. 2017, doi: 10.1016/j.dss.2017.10.005.
[58] A. Ashouri, Y. Hu, G. R. Newsham, and W. Shen, “Energy Performance Based Anomaly Detection in Non-Residential Buildings Using Symbolic Aggregate Approximation,” in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, Aug. 2018, pp. 1400–1405. doi: 10.1109/COASE.2018.8560433.
[59] A. A. Ferreira *et al.*, “Adaptive Piecewise and Symbolic Aggregate Approximation as an Improved Representation Method for Heat Waves Detection,” in *Intelligent Computing*, K. Arai, S. Kapoor, and R. Bhatia, Eds., in Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2019, pp. 658–671. doi: 10.1007/978-3-030-01174-1_51.
[60] J. Liu, T. Li, Z. Yuan, W. Huang, P. Xie, and Q. Huang, “Symbolic aggregate approximation based data fusion model for dangerous driving behavior detection,” *Inf. Sci.*, vol. 609, pp. 626–643, Sep. 2022, doi: 10.1016/j.ins.2022.07.118.
[61] M. Gallimore, M. Riley, and C. Bingham, “Self-Organizing Piecewise Aggregate Approximation algortihm for intelligent detection and diagnosis of heart conditions,” presented at the International Conference on Medical and Health Sciences, Berlin, Germany: IRES, Jun. 2015. Accessed: Oct. 30, 2023. [Online]. Available: https://eprints.lincoln.ac.uk/id/eprint/17747/
[62] J. M. Velasco, O. Garnica, S. Contador, M. Botella, J. Lanchares, and J. I. Hidalgo, “Forecasting glucose levels in patients with diabetes mellitus using semantic grammatical evolution and symbolic aggregate approximation,” in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, in GECCO ’17. New York, NY, USA: Association for Computing Machinery, Jul. 2017, pp. 1387–1394. doi: 10.1145/3067695.3082493.
[63] A. Mezari and I. Maglogiannis, “Gesture recognition using symbolic aggregate approximation and dynamic time warping on motion data,” in *Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare*, in PervasiveHealth ’17. New York, NY, USA: Association for Computing Machinery, May 2017, pp. 342–347. doi: 10.1145/3154862.3154927.
[64] C. Guo, H. Li, and D. Pan, “An Improved Piecewise Aggregate Approximation Based on Statistical Features for Time Series Mining,” in *Knowledge Science, Engineering and Management*, Y. Bi and M.-A. Williams, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2010, pp. 234–244. doi: 10.1007/978-3-642-15280-1_23.
[65] “Random projection,” *Wikipedia*. Oct. 08, 2023. Accessed: Nov. 05, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Random_projection&oldid=1179243954
[66] N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos, and Y. Theodoridis, “TACO: tunable approximate computation of outliers in wireless sensor networks,” in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, in SIGMOD ’10. New York, NY, USA: Association for Computing Machinery, Jun. 2010, pp. 279–290. doi: 10.1145/1807167.1807199.
[67] N. Giatrakos, A. Deligiannakis, M. Garofalakis, and Y. Kotidis, “Omnibus outlier detection in sensor networks using windowed locality sensitive hashing,” *Future Gener. Comput. Syst.*, vol. 110, pp. 587–609, Sep. 2020, doi: 10.1016/j.future.2018.04.046.
[68] N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos, and Y. Theodoridis, “In-network approximate computation of outliers with quality guarantees,” *Inf. Syst.*, vol. 38, no. 8, pp. 1285–1308, Nov. 2013, doi: 10.1016/j.is.2011.08.005.
[69] P. Bholowalia, “EBK-Means: A Clustering Technique based on Elbow Method and K-Means in WSN,” *Int. J. Comput. Appl.*, vol. 105, no. 9.
[70] M. Hahsler, M. Piekenbrock, and D. Doran, “dbscan: Fast Density-Based Clustering with R,” *J. Stat. Softw.*, vol. 91, pp. 1–30, Oct. 2019, doi: 10.18637/jss.v091.i01.

[71] "A Comprehensive Introduction to Graph Neural Networks (GNNs)." Accessed: Nov. 12, 2023. [Online]. Available: https://www.datacamp.com/tutorial/comprehensive-introduction-graph-neural-networks-gnns-tutorial
[72] A. Bhardwaj, "Silhouette Coefficient : Validating clustering techniques," Medium. Accessed: Dec. 12, 2023. [Online]. Available: https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c
[73] "Adjusted Rand Index - an overview | ScienceDirect Topics." Accessed: Dec. 12, 2023. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/adjusted-rand-index
[74] "R-Squared," Corporate Finance Institute. Accessed: Dec. 12, 2023. [Online]. Available: https://corporatefinanceinstitute.com/resources/data-science/r-squared/
[75] "Root Mean Square Error - an overview | ScienceDirect Topics." Accessed: Dec. 12, 2023. [Online]. Available: https://www.sciencedirect.com/topics/engineering/root-mean-square-error
[76] "F-score," *Wikipedia*. Sep. 09, 2023. Accessed: Dec. 12, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=F-score&oldid=1174585267
[77] H. Ö. Tan and I. Körpeoğlu, "Power efficient data gathering and aggregation in wireless sensor networks," *ACM SIGMOD Rec.*, vol. 32, no. 4, pp. 66–71, Dec. 2003, doi: 10.1145/959060.959072.