Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Time Series Data Compression for Industrial Data

Riya Bansal

riyambansal@gmail.com

PES University

Prarthana P Bhat

PES University

Praneeth Cheepurupalli

PES University

Pratham P Bhat

PES University

Dr. Bhargavi Mokashi

PES University

Research Article

Keywords: compression, pattern-recognition, time-series data, CNN Autoencoder, Deflate Algorithm

Posted Date: November 19th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11771222/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Time Series Data Compression for Industrial Data

1st Prarthana P Bhat
PES University
Bangalore, India
prarthanapb14@gmail.com

2rd Pratham P Bhat
PES University
Bangalore, India
prathampb2003@gmail.com

3th Riya Bansal
PES University
Bangalore, India
riyambansal@gmail.com

4th Praneeth Cheepurupalli
PES University
Bangalore, India
chpraneeth2003@gmail.com

5nd Dr. Bhargavi Mokashi
Professor
PES University
Bangalore, India
bhargavim@pes.edu

***Abstract*—Industrial processes generate vast amounts of time-series data, necessitating efficient compression methods to reduce storage and transmission costs while preserving data accuracy. This project introduces a time-series compression algorithm tailored to industrial data, leveraging pattern recognition to identify and compress repetitive sequences. The proposed method not only enhances compression efficiency but also ensures the retention of critical information required for analysis and forecasting. Experimental results demonstrate that our approach achieves a 5 to 6.5 times higher compression ratio compared to traditional compression algorithms, with minimal loss of data fidelity. This makes it a scalable and effective solution for real-time industrial data management.**

***Index Terms*—compression, pattern-recognition, time-series data, CNN Autoencoder, Deflate Algorithm**

I. Introduction

In industries that rely on heavy machinery, such as power generation, manufacturing, and transportation, close real-time monitoring of equipment is critical to maintaining operational performance and preventing costly shut downs. To ensure proper operation of machines such as turbines, generators, and other heavy-duty equipment, a wide range of sensors is installed to continuously measure critical parameters like rotational speed, temperature, wind speed, and power. These sensors generate significant amounts of time series data, which are recorded and transmitted to centralized analysis centers on a regular basis—often hourly. This allows for timely analysis of equipment performance, identifying potential issues before they lead to operational downtime or expensive repairs. However, as industrial operations expand, the volume of data from those machines grows exponentially. The constant influx of time series data creates significant challenges for data storage and management. Cloud storage has become the most widely adopted solution for handling such large amounts of data, offering scalability and flexibility. However, the costs of these extremely large datasets may quickly balloon and become expensive for organizations. The size of the data, together with the intent to maintain it for significant periods, highlights the need for better storage schemes.

Compressing time series data before storage helps reduce costs. Traditional methods encode raw data without considering underlying patterns and cyclic behaviors common in machine-generated time series. Since machine activity often recurs under similar conditions, these patterns are easily identifiable. However, conventional compression techniques fail to fully leverage this periodicity, leading to suboptimal compression ratios.

This research proposes a new approach using autoencoder CNNs to enhance compression by recognizing and learning these recurring patterns. This allows the algorithm to encode similar sequences just once, reducing redundancy. The result is significant storage savings without losing essential information, as the data remains easily decompressible.

By applying advanced machine learning for compression, this method offers a more efficient solution for managing large-scale industrial time-series data. It not only reduces storage costs—potentially saving millions annually—but also improves long-term data management, making it highly competitive with traditional algorithms.

II. Literature Survey

A. Periodic Time Series Data Analysis by Deep Learning Methodology

H. Zhang, H. Lu, and A. Nayak [1] use Convolutional Networks (CNNs) in this study, and propose an algorithm called the PTSD algorithm. This uses layers of CNN's to predict the period length of the time series data.It however seems to work best only for strictly periodic segment. It is not most suitable for classifying the data solely based on the pattern that it forms.

B. An Unknown Pattern Detection Method for Time Series Data Based on Convolutional Neural Networks

The paper [2] proposes a method to classify time series, specifically that of ECGs. It uses the ecg5000 dataset. The model consists of convolutional layers followed by pooling. Then it is fed to a fully connected layer. In this paper, improvements are made to the traditional fully connected

layer and classifier. The network model outputs the probability of a pattern's category as coordinate points and employs Euclidean distance for classification. Each pattern represents a binary classification, and the network consists of N binary classifications for known patterns. In the fully connected layer, the model extracts the x and y coordinate values from the sequence data of the known patterns, maps them to the range [0,1] using the sigmoid function, and generates a one-dimensional vector of size 1N for each coordinate. The results are then concatenated using the concat function, producing a (12N) vector that represents the probability coordinates for each element across all the given patterns. The probability coordinate vector is submitted to the softmax classifier which classifies the pattern.

C. Robust Dominant Periodicity Detection for Time Series with Missing Data

The paper [3] presents a novel approach to detecting periodicity in time series data, addressing challenges like abrupt trend changes, noise, outliers, and missing data blocks. Traditional methods—such as Fourier transforms, autocorrelation functions (ACF), and joint time-frequency techniques—often struggle with these issues. The authors propose a robust solution combining a trend filter that handles missing data without imputation, a noise-resistant ACF estimator using FFT/IFFT, and a time-frequency method for accurate period estimation. Compared to methods like Fisher's test, the Lomb-Scargle periodogram, and RobustPeriod, their algorithm outperforms in real-world datasets with significant missing data and outliers. Future research could extend this method to handle multiple periodicities and broader real-world applications.

D. Periodicity Detection in Time Series Databases

The paper [4] explores mining periodic patterns in time series data without requiring pre-specified period lengths. It introduces segment periodicity, detecting repeated segments, and symbol periodicity, identifying recurring symbols. The proposed O(n log n) algorithms are efficient, requiring a single dataset scan. Segment periodicity detection uses approximate string matching and FFT, while symbol periodicity employs a modified convolution and binary mapping. Experiments on synthetic and real-world data confirm accuracy, noise resilience, and scalability, making these methods valuable for time series pattern mining.

E. Research and application of pattern recognition LSTM based bridge data anomaly detection

Recent advancements in machine learning, particularly with recurrent neural networks (RNNs) and their variants like Long Short-Term Memory (LSTM) networks, have shown strong potential for capturing temporal dependencies in sequential data. Studies that incorporate domain-specific feature engineering further improve detection by embedding contextual knowledge into model inputs. However, balancing accuracy, computational efficiency, and real-time performance remains challenging. The approach proposed in [5] tackles these issues by constructing nine optimized features for high-frequency acceleration data and employing an LSTM-based framework for anomaly detection. Integrated with the Flux framework for real-time operation, this method significantly boosts detection accuracy to 99.65

F. Dynamic Memory Induction Networks for Few-Shot Text Classification

R. Geng, B. Li, Y. Li, J. Sun, and X. Zhu [6] propose a method to improve few-shot text classification using dynamic memory routing and a query-aware induction approach. Leveraging pre-trained models like BERT for effective sentence representation, their model—DMIN—outperforms previous methods like Induction Networks and HATT by 2-4% on datasets such as miniRCV1 and ODIC. Its strengths include adaptability across diverse support sets, improved class representation, and thorough ablation studies validating each component's contribution. However, dynamic memory increases computational and memory demands, and limited dataset testing raises concerns about its broader applicability. While DMIN excels in classification tasks, its potential for other applications remains uncertain, and its complexity may reduce interpretability compared to simpler models.

G. An Exploratory Study of Periodicity Detection Algorithms for Energy Theft Prevention in Smart Grids

Ayogu et al. [7] address the challenge of detecting electricity theft by proposing a hybrid model for periodicity analysis. It reduces computational load by using autocorrelation with FFT to enhance Dynamic Time Warping (DTW) for similarity detection. The model segregates data into normal and theft, then employs machine learning algorithms—Decision Tree, Random Forest, Extremely Randomized Trees (ETC), and SVM—to detect theft using daily and weekly consumption data. The ETC model showed the best performance, achieving AUC scores of 95% and 98% on daily and weekly data, respectively.

H. Machine Learning for Univariate Time Series Imputation

The study [8] presents MLBUI, a machine learning-based method for imputing missing values in univariate time series using Random Forest and SVM. It transforms the series into a multivariate format and applies forward and backward forecasting, averaging both predictions for imputation. Tested on four real-world datasets, MLBUI outperforms existing methods across various gap sizes, leveraging multi-step-ahead prediction and machine learning for improved accuracy.

I. Detecting Time Series Periodicity Using Complex Networks

Ferreira et al. [9] present a noise-resistant method for detecting periodicity in time series using complex networks. The series is transformed into a network, with intervals as vertices and edges linking consecutive observations in different intervals. Community detection reveals periodic patterns. This parameter-free approach handles noise and missing data

through discretization and network analysis. Tested on synthetic and meteorological data, it effectively identifies periodic changes, overcoming limitations of traditional methods.

J. Developing Fast Techniques for Periodicity Analysis of Time Series

Mohammed et al. [10] focus on the development of two new algorithms to address the computational complexity in periodicity detection using Dynamic Time Warping (DTW). The proposed methodologies combine Fast Fourier Transform (FFT) and Wavelet Transform (WT) with DTW to form FFT-DTW and WT-DTW algorithms, respectively. These techniques are aimed at speeding up the DTW process by using FFT and WT to identify candidate periods, reducing the need to analyze all possible periods. The authors evaluate the performance of these algorithms on both real and synthetic datasets, showing promising results in terms of both accuracy and computational efficiency. The paper highlights the effectiveness of the combined techniques in various application domains like meteorology and econometrics .

K. Sprintz: Time Series Compression for the Internet of Things

Blalock et al. [11] propose Sprintz, a high-throughput compression algorithm optimized for IoT time-series data streams. Sprintz combines delta coding and byte-level compression, achieving compression ratios of 3.5x–8.0x. It processes data at over 250 MB/s on a single core, significantly outperforming popular algorithms like Gorilla and RLE. The paper emphasizes Sprintz's suitability for low-latency IoT devices, enabling real-time compression in edge computing environments with limited resources. Sprintz strikes an excellent balance between efficiency, speed, and memory usage, making it ideal for large-scale sensor networks.

L. TSXor: A Simple Time Series Compression Algorithm

Bruno et al. [12] introduce TSXor, an efficient algorithm that applies XOR-based delta encoding to compress time-series data streams. TSXor achieves 50% better compression ratios than standard delta encoding techniques, with compression speeds of around 120 MB/s and decompression speeds exceeding 150 MB/s. It is particularly effective for sensor networks with frequent small-value changes. Due to its low computational complexity, TSXor is suitable for real-time IoT applications, ensuring high-speed transmission with minimal energy consumption, which is essential for battery-powered devices.

M. Cocv: A compression algorithm for time-series data with continuous constant values in IoT-based monitoring systems

Lin et al. [13] propose Cocv, a compression algorithm targeting datasets with continuous constant values, which are common in IoT monitoring systems. Cocv leverages run-length encoding to efficiently compress long sequences of static data, achieving an average compression ratio of 6.3:1. It operates at a speed of over 200 MB/s, outperforming generic algorithms like Gzip and Snappy. Cocv is highly applicable in fields such as healthcare and environmental monitoring, where data streams often remain stable over time. Its lightweight design ensures minimal computational overhead, making it ideal for low-power IoT devices. nayn

N. RAKE: A simple and efficient lossless compression algorithm for the Internet of Things

Campobello et al. [14] propose RAKE, a lossless compression algorithm designed specifically for IoT environments. RAKE combines predictive coding with adaptive encoding, achieving compression ratios ranging between 2.8:1 and 5.2:1. With a throughput of 100 MB/s, it ensures efficient performance even on constrained hardware, such as microcontrollers and wireless sensor networks. RAKE's energy-efficient design makes it well-suited for real-time applications where maintaining data integrity is crucial, such as industrial monitoring systems and smart agriculture networks.

III. Dataset

The dataset utilized in this study is the Wind Turbine SCADA Dataset. For analysis, two subsets of time-series data were employed: Dataset 1, comprising wind speed measurements, and Dataset 2, containing wind direction observations. Both datasets consist of time-series data collected from a wind turbine at 10-minute intervals, capturing critical operational and environmental parameters for further investigation.

IV. Methodology

This methodology outlines a data compression approach optimized for industrial machinery wind speed data in JSON format, where each entry contains a timestamp and a floating-point wind speed value. The method encompasses data preprocessing, window-based classification with a Convolutional Neural Network (CNN), compression using the Deflate algorithm, and a comprehensive evaluation of compression efficacy.

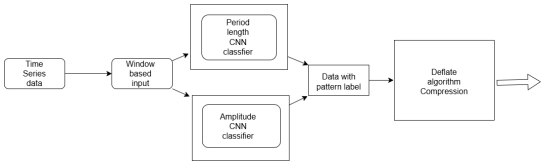


Fig. 1. Classifier Architecture.

A. Data Preprocessing

Conversion to JSON Format: Both datasets, Dataset 1 (wind speed time series) and Dataset 2 (wind direction time series), were converted into a JSON format. Each entry in this format consists of a timestamp and the corresponding value as separate fields, i.e., timestamp, value. This format mirrors the structure of time-series files typically generated by industrial systems. These JSON files are treated as the original dataset files and will serve as a baseline for evaluating the accuracy of the decompressed files.

Segmentation into Fixed-Length Windows: The datasets were further segmented into fixed-length windows, with each window containing consecutive time-series values. This segmentation is crucial for capturing distinct patterns and trends over specific intervals, making the data more suitable for subsequent analysis. Each window represents a ”wave” that will later be classified by the predictive model. The choice of window length is informed by the variability and periodic nature of the wind turbine data, striking a balance between detail and computational efficiency. The segmented data is stored in a tab-separated values (TSV) file for downstream processing.

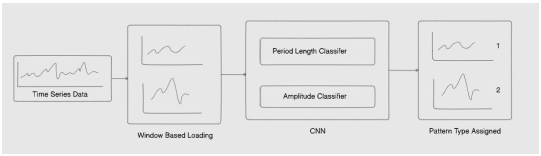


Fig. 2. Classifier Architecture.

B. Window Classification Using Convolutional Neural Network (CNN) and Encoder

The process begins by feeding the windows into a Convolutional Neural Network (CNN) integrated with an autoencoder. This combination extracts and represents the input in a structured feature space, where the parameters of this feature space are optimized during training. Each instance in the feature space is subsequently processed by a k-Nearest Neighbors (kNN) clustering model, which assigns a label based on its pattern type.

To provide additional classification, each instance is also passed through a second CNN-autoencoder pipeline, structurally similar to the first. This model extracts features specifically relevant to amplitude classification. The resulting feature representation is again clustered using kNN, which assigns a label corresponding to the instance's amplitude.

As a result, each instance is characterized by two distinct labels: one representing its pattern type and another representing its amplitude. These dual classifications enrich the descriptive power of the model, enabling more nuanced analysis.

Aggregation of Identical Windows:

Windows with similar amplitude and period length are assigned a unique identifier. For each consecutive matching window, a count value is incremented rather than storing repeated data entries. This aggregation process produces a reduced dataset, where each unique window pattern is stored only once, with the count value denoting its frequency. The classified windows and their counts are saved in a classified.csv file, including columns for timestamp, values and count .

C. Data Compression using Deflate Algorithm

After classification, the classified.csv file is converted into a binary format to facilitate compression. The data is then compressed using the Deflate algorithm provided by the zlib library:

Deflate Algorithm: Deflate combines LZ77 dictionary-based compression with Huffman coding to achieve lossless compression. The LZ77 algorithm identifies and encodes repeated patterns in the data, while Huffman coding assigns shorter codes to more frequent patterns, resulting in significant storage reduction.

Binary Conversion: To further optimize storage, the classified data is converted into binary form. Each row in the classified.csv file is encoded into a binary format, leveraging bit-level encoding to minimize the overall file size. The resulting binary file is then processed through the Deflate algorithm to maximize compression efficiency.

S.No	MSE	MAE
Pattern Classifier CNN	8.47e-07	6.81e-4
Amplitude Classifier CNN	9.84e-2	0.26

TABLE I
EVALUATION OF CNN AUTOENCODER MODEL

Metric	Classifer1	Classifier2
Silhouette Score	0.376	0.319
Davies-Bouldin Index	0.918	0.937
Calinski-Harabasz Index	43134.37	34766.31

TABLE II
EVALUATING THE CLUSTERING

D. Data Decompression and Reconstruction

Decompression involves expanding the classified and compressed data back to its original structure. This is achieved by:

Data Expansion Based on Count: Each entry in the classified dataset is repeated according to its count value, reconstructing the original frequency of each window pattern. The timestamp of each window is recalculated based on the initial timestamp stored in the classified.csv file and the period length, ensuring the temporal structure of the original dataset is preserved.

Temporal Alignment of Timestamps: The first timestamp of each unique window is recorded in the classified.csv file, and subsequent timestamps are calculated by adding the period length for each repeated entry. This approach maintains the sequential timing of wind speed readings, critical for time-series data analysis.

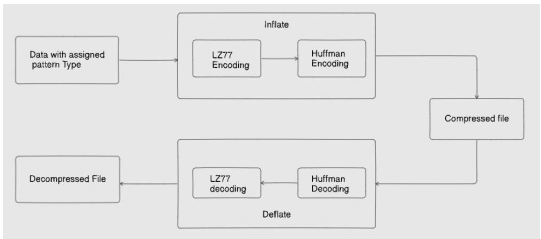


Fig. 3. Compression and Decompression Architecture

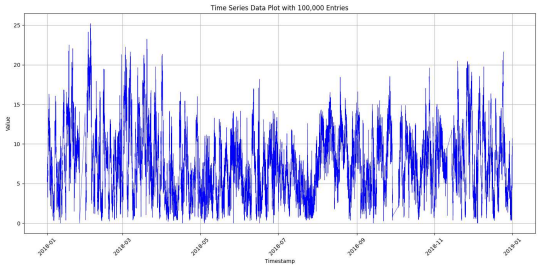


Fig. 4. Original data plot for Dataset1

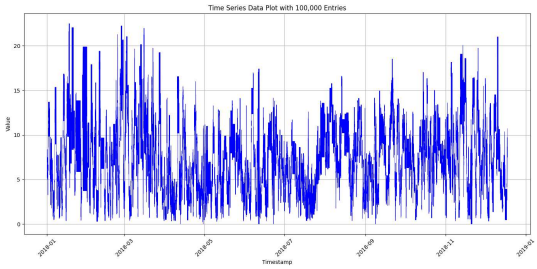


Fig. 5. Decompressed data plot for Dataset1

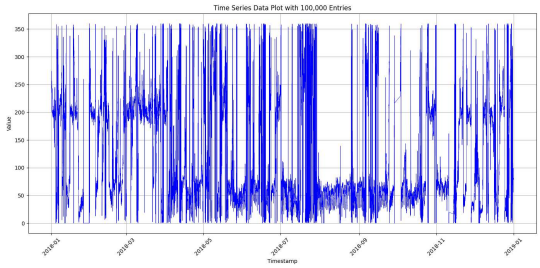


Fig. 6. Original data plot for Dataset2

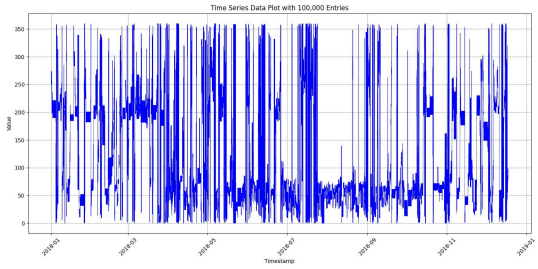


Fig. 7. Decompressed data plot for Dataset2

E. Performance Evaluation Metrics

To assess the effectiveness of this compression method, a series of performance metrics are employed:

Compression Ratio: Defined as the ratio of the original data size to the compressed data size, this metric quantifies the reduction in storage requirements achieved by the compression algorithm.

Mean Squared Error (MSE) and Root Mean Squared Error (RMSE): These metrics measure the fidelity of the decompressed data by calculating the average squared and root squared difference between the original and reconstructed wind speed values, respectively. Lower MSE and RMSE values indicate higher accuracy in data reconstruction.

Data Loss Index (DLI): The DLI evaluates the structural integrity of the decompressed data by measuring any loss in essential data points that might affect interpretability or accuracy.

Information Loss Ratio (ILR): This ratio assesses the information loss due to compression, defined as the proportion of data points altered or lost relative to the total dataset. A lower ILR signifies better retention of original data information after compression.

A. Results

Normalized Mean Squared Error (NMSE) and Normalized Root Mean Squared Error (NRMSE): These metrics measure data fidelity in terms of the average squared difference between the original and decompressed values.

Data Loss Index (DLI) and Information Loss Ratio (ILR): DLI quantifies the structural data loss, while ILR measures the informational loss relative to the data size.

Information Loss Ratio (ILR): It evaluates key information loss, helping assess pattern preservation in the compressed data.

Compression Ratio: This metric quantifies the extent of data reduction achieved by the method, calculated as the ratio of the original file size to the compressed file size.

S.No	Metric	Dataset 1	Dataset 2
1	Normalized Mean Squared Error (NMSE)	0.044	0.115
2	Normalized Root Mean Squared Error (NRMSE)	0.209	0.339
3	Data Loss Index (DLI)	0.039	0.039
4	Information Loss Ratio (ILR)	0.161	0.239

TABLE III
METRICS FOR DATASET 1 AND DATASET 2.

Dataset	Without Our Algorithm	With Our Algorithm
Dataset1	6.377	31.83
Dataset2	6.406	41.310

TABLE IV
COMPARISON OF COMPRESSION RATIOS WITH AND WITHOUT OUR ALGORITHM.

B. Conclusion

This project developed an efficient system for compressing industrial machinery wind speed data. The approach applies a window-based segmentation and feature extraction, with classification through a CNN model, such that it aggregates similar data patterns that are compressed after the categorization process is completed. Effective lossless compression has been achieved using the Deflate algorithm, which combines

Compression Algorithm	Avg Compression Ratio
TSXOR	3.36
Sprintz 16-bit	2.4
Sprintz 8-bit	6.0
RAKE	12.0
gZip	7.4
Our Algorithm	36.57

TABLE V
COMPARISON OF COMPRESSION RATIOS FOR DIFFERENT ALGORITHMS.

LZ77 and Huffman coding, balancing reduced data size while retaining the same level of data fidelity.

Our methodology has demonstrated considerable improvements in storage density as well as seamless communication of massive-scale time-series data. To put it in a nutshell, this project highlights that advanced and specific data compression techniques can significantly improve industrial data handling by optimizing storage and enabling data to be transferred efficiently, thus fostering smartly connected industrial systems.

C. Future Work

One potential avenue for future research lies in enhancing the pattern recognition capabilities of the compression algorithm to identify patterns that are amplified or deamplified versions of previously encountered waves. Currently, the algorithm does not recognize a wave that is a scaled multiple of a prior wave as the same pattern, limiting its ability to generalize across such transformations.

Developing a mechanism to account for this scaling factor could significantly improve the algorithm's ability to compress time-series data. By recognizing amplified or deamplified versions of a wave as the same underlying pattern, the algorithm could achieve a higher level of compression. This advancement would be particularly beneficial for applications where patterns exhibit consistent scaling, such as in dynamic systems or fluctuating industrial processes.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to our research guide, Prof. Bhargavi Mokshi, for her unwavering support, valuable guidance, and constant encouragement throughout the course of this research. Her insightful feedback at every stage of the project, coupled with her patience and encouragement, has made it possible for us to explore new ideas and push the boundaries of our research. We are deeply appreciative of the time and effort she dedicated to mentoring us, providing direction, and ensuring we stayed focused on the key challenges of developing a time series data compression approach. This research would not have been possible without her continued support, and we are truly grateful for the opportunity to have worked under her guidance.

REFERENCES

[1] H. Zhang, H. Lu, and A. Nayak, "Periodic Time Series Data Analysis by Deep Learning Methodology," *IEEE Access*, vol. 8, pp. 223078–223088, 2020. doi: 10.1109/ACCESS.2020.3041600.

[2] J. Bao and X. Li, "An Unknown Pattern Detection Method for Time Series Data Based on Convolutional Neural Network," in *Proc. 2019 2nd International Conference on Safety Produce Informatization (IICSPI)*, 2019, pp. 51–55. doi: 10.1109/IICSPI48186.2019.9095913.

[3] Q. Wen, L. Yang, and L. Sun, "Robust Dominant Periodicity Detection for Time Series with Missing Data," 2023, *arXiv preprint*. [Online]. Available: https://arxiv.org/abs/2303.03553.

[4] M. G. Elfeky, W. G. Aref, and A. K. Elmagarmid, "Periodicity Detection in Time Series Databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 7, pp. 875–887, 2005. doi: 10.1109/TKDE.2005.114.

[5] Z. Gao, F. Li, X. Yu, J. Yan, and Z. Chen, "Research and Application of Pattern Recognition LSTM Based Bridge Data Anomaly Detection," in *Proc. 2022 China Automation Congress (CAC)*, 2022, pp. 4630–4635. doi: 10.1109/CAC57257.2022.10054767.

[6] R. Geng, B. Li, Y. Li, J. Sun, and X. Zhu, "Dynamic Memory Induction Networks for Few-Shot Text Classification," 2020, *arXiv preprint*. [Online]. Available: https://arxiv.org/abs/2005.05727.

[7] I. I. Ayogu and R. E. Ogu, "An Exploratory Study of Periodicity Detection Algorithms for Energy Theft Prevention in Smart Grids," in *Proc. 2021 IEEE PES/IAS PowerAfrica*, 2021, pp. 1–5. doi: 10.1109/PowerAfrica52236.2021.9543267.

[8] T. T. H. Phan, "Machine Learning for Univariate Time Series Imputation," in *Proc. 2020 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*, 2020, pp. 1–6. doi: 10.1109/MAPR49794.2020.9237768.

[9] L. N. Ferreira and L. Zhao, "Detecting Time Series Periodicity Using Complex Networks," in *Proc. 2014 Brazilian Conference on Intelligent Systems*, 2014, pp. 402–407. doi: 10.1109/BRACIS.2014.78.

[10] M. Y. Y. Mohammed and M. Celik, "Developing Fast Techniques for Periodicity Analysis of Time Series," in *Proc. 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2019, pp. 1–5. doi: 10.1109/ISMSIT.2019.8932865.

[11] D. Blalock, S. Madden, and J. Guttag, "Sprintz: Time Series Compression for the Internet of Things," *Proc. ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1–23, 2018. doi: 10.1145/3264903.

[12] A. Bruno, F. M. Nardini, G. E. Pibiri, R. Trani, and R. Venturini, "TSXor: A Simple Time Series Compression Algorithm," in *Advances in Information Retrieval*, 2021, pp. 217–223. doi: 10.1007/978-3-030-86692-1_18.

[13] S. Lin, W. Lin, K. Wu, S. Wang, M. Xu, and J. Z. Wang, "Cocv: A Compression Algorithm for Time-Series Data with Continuous Constant Values in IoT-Based Monitoring Systems," *Internet of Things*, vol. 25, p. 101049, 2024. doi: https://doi.org/10.1016/j.iot.2023.101049.

[14] G. Campobello, A. Segreto, S. Zanafi, and S. Serrano, "RAKE: A Simple and Efficient Lossless Compression Algorithm for the Internet of Things," in *Proc. 2017 25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 2581–2585. doi: 10.23919/EUSIPCO.2017.8081677.