Article

An Evolving TinyML Compression Algorithm for IoT Environments Based on Data Eccentricity

Gabriel Signoretti [1], Marianne Silva [1,*], Pedro Andrade [1], Ivanovitch Silva [1,*], Emiliano Sisinni [2] and Paolo Ferrari [2]

[1] UFRN-PPgEEC, Postgraduate Program in Electrical and Computer Engineering, Federal University of Rio Grande do Norte, Natal 59078-970, Brazil; gabrielsig@ufrn.edu.br (G.S.); pedro.meira.055@ufrn.edu.br (P.A.)

[2] UNIBS-DIE, Department of Information Engineering, University of Brescia, 25123 Brescia, Italy; emiliano.sisinni@unibs.it (E.S.); paolo.ferrari@unibs.it (P.F.)

* Correspondence: marianne.silva.086@ufrn.edu.br (M.S.); ivanovitch.silva@ufrn.br (I.S.)

Abstract: Currently, the applications of the Internet of Things (IoT) generate a large amount of sensor data at a very high pace, making it a challenge to collect and store the data. This scenario brings about the need for effective data compression algorithms to make the data manageable among tiny and battery-powered devices and, more importantly, shareable across the network. Additionally, considering that, very often, wireless communications (e.g., low-power wide-area networks) are adopted to connect field devices, user payload compression can also provide benefits derived from better spectrum usage, which in turn can result in advantages for high-density application scenarios. As a result of this increase in the number of connected devices, a new concept has emerged, called TinyML. It enables the use of machine learning on tiny, computationally restrained devices. This allows intelligent devices to analyze and interpret data locally and in real time. Therefore, this work presents a new data compression solution (algorithm) for the IoT that leverages the TinyML perspective. The new approach is called the Tiny Anomaly Compressor (TAC) and is based on data eccentricity. TAC does not require previously established mathematical models or any assumptions about the underlying data distribution. In order to test the effectiveness of the proposed solution and validate it, a comparative analysis was performed on two real-world datasets with two other algorithms from the literature (namely Swing Door Trending (SDT) and the Discrete Cosine Transform (DCT)). It was found that the TAC algorithm showed promising results, achieving a maximum compression rate of 98.33%. Additionally, it also surpassed the two other models regarding the compression error and peak signal-to-noise ratio in all cases.

Keywords: internet of things; online data compression; TinyML; eccentricity; evolving algorithm; LPWAN

Citation: Signoretti, G.; Silva, M.; Andrade, P.; Silva, I.; Sisinni, E.; Ferrari, P. An Evolving TinyML Compression Algorithm for IoT Environments Based on Data Eccentricity. *Sensors* **2021**, *21*, 4153. https://doi.org/10.3390/s21124153

Academic Editors: Eyhab Al-Masri, Chi-Hua Chen, Alireza Souri and Olivera Kotevska

Received: 8 May 2021
Accepted: 15 June 2021
Published: 17 June 2021

1. Introduction

The Internet of Things (IoT) allows the continuous monitoring of environments and machines using small and inexpensive sensors [1,2]. From this perspective, advances in the technologies of sensors, microcontrollers, and communication protocols have made the mass production of IoT platforms with many connectivity options possible, and at affordable prices [3–5]. Due to the low cost of IoT hardware, the sensors are being deployed on a large scale to monitor machines and, in general, public and residential spaces [6–8].

These sensors monitor the physical properties associated with their deployment environments—twenty-four hours a day, seven days a week—making them a critical facilitator to increase the volume of the data generated, due to the collection performed in a wide variety of scenarios [9,10]. Note that these data streams can form a large volume of data very rapidly, and as a consequence, they generate the need for a vast amount of disk storage space [11,12]. For example, accelerometers embedded in vehicles are constantly collecting vibration patterns for various types of roads. Likewise, several other sensors such as magnetometer, pressure, temperature, humidity, and ambient light sensors, among many others, measure the physical conditions wherever they are installed, in addition to those that have been developed for the health area [8,13].

Therefore, it can be noted that the generation of data by the sensors occurs in a continuous stream format, which has varying characteristics according to each application. The sample rate, for example, may vary from once a day to every millisecond depending on the context [14]. This variation in the collection occurs because there are several distinct application demands, different device models, and different configuration needs or limitations in the data transmission channel [15].

In this scenario, it is essential to study and use techniques that can compress the data, as well as algorithms and functionalities that can be embedded in these devices. In the last few decades, much research in the specific area of data compression in IoT-based monitoring systems has been published [9,11,16], focusing on different layers and proposing different strategies. Generally, compression aims to eliminate redundant information in order to minimize the space required for storage, consequently reducing the number of bits to be transmitted periodically to the endpoint, in addition to improving efficiency in the information analysis [9,17].

In this manner, some challenges regarding the amount of data generated per second are encountered, especially when using low-power transmission protocols as LoRaWAN [18–20]. Furthermore, it is known that most of the battery consumption of a device that communicates using a wireless medium is in the transmission of the data and not in the processing of the information, which makes it advantageous in terms of energy to compress data locally before transmission [16,21,22].

Often, IoT devices work with low-frequency processors and have limited local data storage capacity, severe memory restrictions [23,24], simple protocols, and inefficient medium access strategies (e.g., pure ALOHA). Additionally, when operating in unlicensed bands, the bandwidth is scarce. When the sub-GHz transmissions are used in order to increase the coverage, the data rate is usually very low. For all these reasons, decreasing the time-on-air of each transmission can reduce collisions, save bandwidth, and allow a higher number of coexisting nodes (for high-density applications [25])

Therefore, it is clear that the new trend is to add "intelligence" to IoT devices so that they process data and make decisions without passing all the raw data to the cloud [9,26]. In this scenario, data processing is pushed to the "edge" of the network, even to IoT devices and sensors used primarily for data acquisition [27–29]. Thus, to enable services such as those mentioned previously, some solutions are available in the literature, such as [11,21,30], which proposed different data compression algorithms.

However, in those proposals, the algorithms were not structured with Machine Learning (ML) techniques, which is the conjunction of methods for data analysis that automates the construction of analytical models [31]. Machine learning algorithms allow the discovery of interesting patterns in the data, most of which are beyond what is possible by simple manual inspection. The combination of IoT devices and ML algorithms allows for a wide range of intelligent applications and enhanced user experiences, as shown in Figure 1.

This trend that is enabling the deployment of ML models in tiny, low-power, and low-latency IoT devices is called TinyML [7,32]. This alternative paradigm of TinyML tries to bring more intelligence to IoT devices, enabling the creation of novel applications that embed ML tasks in them. Furthermore, by allowing some data analysis and interpretation to be performed locally and in real time at the collection point, services as these can translate into huge cost savings and better privacy protection [7].

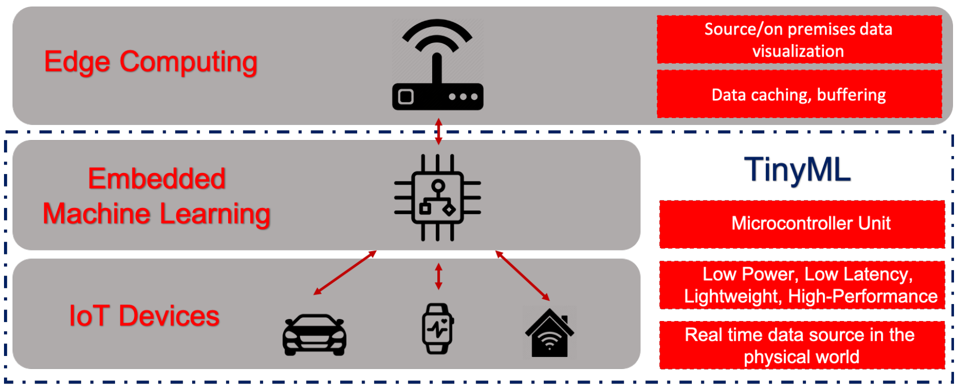

Edge Computing
Source/on premises data visualization
Data caching, buffering
Embedded Machine Learning
TinyML
Microcontroller Unit
IoT Devices
Low Power, Low Latency, Lightweight, High-Performance
Real time data source in the physical world

Figure 1. TinyML ecosystem-as-a-service and the use of machine learning at the edge.

Most of the work performed in the field of TinyML has been focused on the reduction and optimization of existing models, such as Artificial Neural Networks (ANNs), to fit into these tiny devices and commodity microcontrollers, despite their computational restrictions [7,32,33]. Additionally, for IoT scenarios, it can be argued that the algorithms should preferably work without prior knowledge of the data, i.e., unsupervised. These are a class of models that are not explicitly instructed to predict something specific; rather, the patterns in the data are identified by the algorithm itself, and it is left to the user to interpret the output and reach a conclusion [34,35].

However, many of these existing models are static and require training phases to be fit to existing data. In contrast, in many IoT applications, the monitored systems are constantly evolving in real time and may encounter unpredicted disturbances and even complete concept drifts, where the characteristics of the data stream change [36]. In these cases, models designed offline might require constant recalibration. With this in mind, the concept of evolving algorithms can be used. These are a class of intelligent methods that can be defined as autonomous and data-driven, with the ability to evolve both their structure and internal parameters dynamically according to the new data received [36].

Thus, this work aimed to propose the development of a novel evolving algorithm for online data compression in IoT devices. The model, called Tiny Anomaly Compressor (TAC), was built from the ground up based on the Typicality and Eccentricity Data Analytics (TEDA) mathematical framework. As such, it does not require the use of previously established models or any assumptions about the underlying distribution of the data. Additionally, it uses recursive equations, which enable an efficient computation with a low computational cost, using little memory and processing power [37]. With all of these characteristics, the proposed TAC algorithm seems a perfect fit for IoT applications, as it constantly evolves over time as new data become available.

Another original work contribution is the analysis of a real-world case study used as a benchmark test for evaluating the proposed solution performance. In particular, a comparison with two other well-documented strategies was carried out, Swing Door Trending (SDT) and the Discrete Cosine Transform (DCT), respectively, as they are very well documented in the literature. The comparative analysis focused on evaluating the compression performance and error rates achieved by each of the models.

Finally, the contributions of this work are detailed below:

- The development of the novel TAC evolving algorithm;
- The methodology for the auto-definition of one of the hyperparameters of the TAC model;
- The performance benchmark with other well-known algorithms for validation;
- The definition of a potential new metric for time series compression evaluation, the Compression F_β score (or CF_β).

The remainder of this paper is organized as follows: Section 2 details some concepts of data compression and provides a brief description of the SDT and DCT algorithms.

Then, in Section 3, a few related works are presented. Following that, in Section 4, the mathematical basis for the TEDA metrics' calculations is presented, whereas in Section 5, the proposed TAC algorithm is detailed. Afterwards, the methodology for the case study and benchmark evaluation are described in Section 6. Results are highlighted in Section 7. In Section 8, we detail the threats to the validity of our study. Finally, conclusions and future directions are indicated in Section 9.

2. Data Compression

Data compression can be characterized as the transformation or encoding of a set of symbols into another with a reduced size [9,11,30]. In information and code theory, data compression is one of the most relevant direct applications [17,38]. By definition, it seeks to detect redundancy in the data and tries to remove it through reduced representation. This approach is commonly accomplished by using compact representations for data that are more repeated and other longer ones for those with a lower frequency [17].

Such redundancies can be: (1) character distribution: within the domain or input alphabet, it is common for some symbols to occur more frequently than others; (2) repetition of characters: it is common for multiple repetitions of a character to appear; this sequence can be replaced by a single appearance of the character and a value indicating the number of repetitions; (3) redundancy: the frequency of data referring to the position within the input file; (4) patterns: the frequency of patterns is similar to the distribution of characters, but for a sequence of characters or symbols.

Therefore, according to [39], there are many universal compression techniques, such as (static or adaptive) Huffman and arithmetic coding, which are ubiquitous in real-world applications. In addition, it is known that they can be categorized into two types: lossy compression and lossless compression. In the first approach (lossy compression), data from the original file cannot be fully recovered after compression. As a result, the file size is permanently reduced, eliminating redundant data. In contrast, in the other approach (lossless compression), all original data from the compressed file can be recovered after decompression [40].

Alternatively, more domain-specific compression algorithms for different types of data began to emerge, such as text, image, audio, video [17], and, more recently, for IoT [40]. In this paper, the focus was on IoT and sensor time series data. For this type of scenario, lossy compression algorithms appear to generally achieve better compression efficiency, taking advantage of the existence of redundant information, as there is no need to recover it later [40].

It is important to note that some authors, as in [10], make a distinction between methods that rely on the different data encoding and transformation techniques to achieve compression and ones that adjust the sampling rate of the device dynamically, describing them as sampling optimization solutions rather than compression.

The end result of both types of algorithms tends to be the same: reducing the data storage footprint. Furthermore, the technique proposed in this paper does not fit nicely in either of those categories, as it does not transform the data, nor does it adjust the sampling rate in an online manner. Rather, the algorithm chooses which of the captured points are relevant to keep, discarding the rest. That being the case, we opted to use the terms interchangeably throughout this paper.

Finally, to test the newly proposed algorithm, called Tiny Anomaly Compressor (TAC), we used two well-documented compression algorithms found in the literature to perform a basic benchmark applied to a case study. The chosen algorithms were the DCT and SDT. These algorithms were selected due to being well documented in the literature and having a prolific use in compression tasks related to IoT scenarios, such as seen in [11,41], respectively. With all of these characteristics, they are suitable to provide a base benchmarking test for the TAC algorithm.

The DCT is a transform algorithm, similar to the discrete Fourier transform. The logic behind the use of transform techniques is to represent the data in a different domain, where they might be more easily compressible. The other fundamental aspect of the DCT is that the majority of the original signal energy is carried by only a few of the returned coefficients. This way, it is possible to discard upwards of 90% percent of the coefficients and still achieve a good representation of the original data after the decompression process (inverse DCT on the remaining coefficients) [41]. The main disadvantage of the DCT in relation to the other algorithms presented here is the fact that it needs to be executed offline with the complete dataset or in smaller batches. It does not work online and from a point-by-point analysis perspective.

In turn, the SDT algorithm is a compression method based on the concept of timed windows that uses linear trends to classify samples. The algorithm creates an area of coverage in the shape of a parallelogram between the last stored value and the most recent value [42], which it uses to classify the samples as relevant or not: samples that fall inside a window are considered similar to the original sample, thus potentially redundant, and can be discarded; on the other hand, if the sample falls outside of the window, it is classified as novel and saved [43]. Every time a new sample is stored, the algorithm creates a new window based on the sample value and the selected parameters [11].

It is important to keep in mind that, due to its popularity, several alternative implementations of the SDT exist in the literature, such as the ones proposed in [11,44,45]. Each of them suggests different improvements and changes to the original algorithm. That being the case, as the focus of the tests was to create a base benchmark, we chose to focus on the original version of the SDT for practical reasons.

Both of these algorithms, as well as the newly proposed TAC, are instances of lossy compression techniques, meaning that the original data cannot be fully reconstructed from the compressed data, as there is some information loss during the compression process. However, the goal of lossy compression algorithms is to preserve the maximum amount of essential information while getting rid of potentially useless data, thus reducing the overall storage requirements. Additionally, the compression process might also have a noise reduction effect, as the algorithms tend to filter out noisy samples, resulting in a, probably, smoother curve.

Therefore, as some portion of the information is lost due to the compression, there is always a trade-off between the amount of compression and the precision of the resulting data. For most applications, there must be a careful tuning of the model hyperparameters to achieve a good compression result while keeping the overall error within an acceptable range.

3. Related Works

A methodical search was conducted through the literature. As a result, several papers were found that impacted the direction of the research carried out. Therefore, eight of these articles were selected that contributed to the development of the proposed solution.

The SDT has been widely used for data compression in IoT scenarios. However, the main drawback is the fact that the SDT algorithm relies on a main user-defined parameter, the Compression Deviation (CD) [43]. The definition of an optimal value for this parameter is application dependent and requires testing or prior knowledge of the data behavior. Many authors proposed improvements and new auto-configuration steps to automatically set the CD parameter. In [45], the authors proposed the use of the Exponential Moving Average (EMA) as a means to define the CD in what they called Adaptive Swing Door Trending (ASDT). However, this function also requires the previous definition of the parameters.

In contrast, the improved SDT solution proposed in [11], called Self-defining SDT (SSDT), requires no prior parameters to be set. However, to do so, the algorithm needs to be trained on a subset of samples at the start of the applications. Their experiments showed good results, but, for other applications where sensor data can experience constant concept shift, it might not be as effective or else need to be periodically retrained.

On another front, the authors in [41] made use of transform-based lossy compression algorithms to compress the data from IoT weather stations. These algorithms work by transforming the data from the time domain to another domain where they can potentially be easier to compress. The three algorithms chosen were the Discrete Cosine Transform (DCT), the Fast Walsh–Hadamard Transform (FWHT), and the Discrete Wavelet Transform (DWT), from which the DCT was shown to perform best. The main drawback of this approach stems from the fact that data transformations cannot be performed online and on a point-by-point basis, as they need the full set of samples to be calculated.

In the work of [46], the authors proposed an online algorithm for streaming data compression, which took into account the generally concave trend of the compression ratio curve and optimized the key operation parameters (block size) through this metric. The algorithm successfully adapted one of the key parameters to the optimal value and yielded near-maximum compression ratios for time series data. The algorithm used was a variation of the Implementation of Dynamic Extensible Adaptive Locally Exchangeable Measures (IDEALEM), a lossy compression method based on statistical similarity. One of the disadvantages of this approach is that a number of data blocks need to be saved to be later statistically compared to newer samples.

In the approach of [10], a multi-layer efficient data transmission strategy for IoT monitoring systems was proposed. The proposed solution was based primarily on two mechanisms. First, they proposed to gather data from multiple sensor nodes into a gateway node. There, essential Key Performance Indicators (KPIs) were collected from the data to be analyzed. The KPIs were defined according to specialists and used to distinguish between important information among all the data. This way, essential features in the data can be sent in real time to the cloud server, while the rest are stored in a local buffer to be periodically transmitted. The second mechanism was the use of a standard zip compression algorithm on these stored data to reduce the amount of data to be transmitted to the cloud. This approach was designed in such a way so as to ensure both the immediate delivery of critical information and the delayed delivery of detailed information with network resource awareness for other messages. The disadvantage came from the need to define the KPIs, as they were very application-dependent, so prior knowledge was needed.

In [47], they proposed to compress industrial data using neural network regression into a representative vector with lossy compression. For the efficiency of the compression, they used the divide-and-conquer method, dividing the data by time and using machine learning technology in the conquer process. Furthermore, with additional techniques, they tried to randomly select a value around the range for conquering, but it was not affected. The drawback of the proposed techniques was that none of them could be performed in an online manner.

In the work [9], they proposed to adapt the SZ algorithm for IoT devices by considering only the floating-point data type, which makes the code smaller and easier to compile on resource-constrained devices. Additionally, the algorithm was adapted to take a 1D array of float sensor data as input and return a byte array to be transmitted to the edge node. In this method, the data are stored locally and transmitted to the edge after each period P of time. Compared to the proposed work, the main difference was that the compression process transformed the data into a byte array, which later needed to be decompressed before they became usable.

In the study [39], the author proposed a two-level compression model that selects a proper compression scheme for each individual point, making it possible for diverse patterns to be captured with fine granularity. Based on this model, they designed and implemented the Adaptive Multi-Model Middle-Out (AMMMO) framework, which provides access to a set of control parameters to categorize data patterns. However, to effectively handle diverse data patterns, they introduced a reinforcement learning-based approach to learn parameter values automatically. The main disadvantage of this approach is its complexity, making it possible to only be deployed at the database level. That being the case, it cannot be applied directly to an online IoT scenario.

Finally, in Table 1, a comparative summary of the characteristics of the works mentioned in this section is presented. There, it is possible to visualize their main similarities and differences. Each row represents one particular work, with the characteristics of the proposed work being presented on the last row. Finally, each of the four columns indicates topics (or features) and how they were covered in the given research. The four features analyzed here were: if the algorithm was executed online; if the proposal used machine learning for compression; if the solution was lightweight; and if the compressed data were encoded in a different format to achieve compression.

Table 1. Summary of related works.

Features / Works	Online Algorithm	Machine Learning	Lightweight Solution	Data Are Encoded
Bristol [43]	yes	no	yes	no
Correa et al. [11]	yes *	no	yes	no
Moon et al. [41]	no	no	no	yes
Gibson et al. [46]	yes	no	no	yes
Lounas et al. [10]	no	no	no	yes
Park et al. [47]	no	yes	no	yes
Azar et al. [9]	yes	no	yes	yes
Yu et al. [39]	no	yes	no	yes
Proposed Work	yes	yes	yes	no

* Requires a training phase.

Thus, it is clear in the literature that studies that developed compression algorithms using machine learning are still few. Furthermore, none of them were found to be lightweight and online solutions. Therefore, from the discussion presented above, it is clear that there are still gaps to be explored in this area, which favors the development of new solutions, for data compressing for the IoT, mainly for the emerging concept of TinyML.

4. Typicality and Eccentricity Data Analytics

The Typicality and Eccentricity Data Analytics (TEDA) mathematical framework, sometimes also called Empirical Data Analysis (EDA) in the literature, was initially proposed in [48] and later expanded in a series of other articles and the book Empirical Approach to Machine Learning [36].

This new set of concepts comes as an alternative to traditional machine learning approaches that are mathematically based on traditional probability theory. TEDA, as its name suggests, is based on the use of the concepts of eccentricity and typicality. These two quantities represent, respectively, the density and proximity of the samples in the data space [48].

This new approach is also statistical, but it differs from the traditional probability theory, which is, in its nature, frequentist and ideal for describing purely random processes [48]. In fact, despite its solid theoretical basis and widespread use today, the theory of probability was developed with purely random processes in mind, such as rolling dice or currency. Thus, it is perfectly suited to describe such purely random processes and variables [37].

However, the biggest motivation for the development of this new methodology is the fact that real processes are, for the most part, not purely random [48]. Thus, unlike random processes, real processes violate the main premises that the traditional probability theory requires [48].

As a result, the TEDA framework emerges as an alternative systematic methodology that does not require, among other things: previous assumptions about the data distribution; the independence of individual data samples (observations); an infinite number of observations, being able to work with only three data samples [37,48]. Therefore, because it is based entirely on the data and its mutual distribution in the data space, the approach can be categorized as empirical in nature.

In the following subsections, the main mathematical concepts behind the methodology are shown and what makes it so flexible that it can be used for the development of a variety of algorithms, such as: anomalies and failure detection [49], image processing, clustering [50,51], classification [52], regression [53], forecast, control, filtering, big data [54], and traffic analysis [55], among others.

4.1. Metrics' Calculation

TEDA is based on several quantities that are calculated from the proximity/similarity of the samples in the data space. However, these measurements are not exactly equal to the density used in statistics and other areas [37].

Among the new concepts introduced by TEDA, three of them are detailed here: accumulated proximity (π), typicality (τ), and eccentricity (ξ).

We start by defining the $x \in \mathbb{R}^n$ data space, where x represents the data stream in an *n-dimensional* space. In this case, the ordered sequence of received data can be represented as follows:

$$\{x_1, x_2, \ldots, x_k, \ldots\},\ x_k \in \mathbb{R}^n,\ k \in \mathbb{N} \tag{1}$$

Thus, we have that each sample x_k is a vector of n dimensions that represent the system in the discrete instant k. The methodology can be extended to samples of any dimensionality n. However, as an example, let us say that we want to observe a phenomenon in which samples are described in a space of one dimension, such as the temperature of a thermostat, for example.

In the context of a data stream, samples can be described as being received discreetly at times k, the first sample being received when $k = 1$, the second when $k = 2$, and so on. Therefore, for any set of two or more samples ($k \geq 2$), we can define a distance metric between them, $d(x, y)$, whether it is the Euclidean, Mahalanobis, Manhattan, cosine, or other [48].

From this, we have that the accumulated proximity (π) of a point x is calculated by adding the distances between the point x and all the k points of the set. This calculation can be seen in Equation (2) [48].

$$\pi_k(x) = \sum_{i=1}^{k} d(x, x_i),\ k \geq 2 \tag{2}$$

The values of typicality (τ) and eccentricity (ξ) can be defined as soon as three or more different samples have been obtained ($k \geq 3$). This is due to the fact that any pair of non-identical samples is equally atypical/eccentric [48]. In this manner, the eccentricity of a sample x in an instant k is defined by the ratio between its accumulated proximity and the sum of the accumulated proximities of all other samples, as seen in Equation (3) [48].

$$\xi_k(x) = \frac{2\pi_k(x)}{\sum_{i=1}^{k} \pi_k(x_i)},\ k > 2,\ \sum_{i=1}^{k} \pi_k(x_i) > 0 \tag{3}$$

The typicality (τ), in turn, is defined as the complement of the eccentricity, shown in Equation (4).

$$\tau_k(x) = 1 - \xi_k(x) \tag{4}$$

It is very important to note that the functions of eccentricity and typicality are bounded and, therefore, have normalized versions [53]. Those are presented here in Equations (6) and (8), respectively.

$$0 \leq \xi_k(x) \leq 1,\ \ \sum_{i=1}^{k} \xi_k(x_i) = 2 \tag{5}$$

$$\zeta_k(x) = \frac{\xi_k(x)}{2} \tag{6}$$

$$0 \le \tau_k(x) \le 1, \ \sum_{i=1}^{k} \tau_k(x_i) = k - 2 \tag{7}$$

$$t_k(x) = \frac{\tau_k(x)}{k-2} \tag{8}$$

These two functions are very important, as they behave similarly to the Probability Density Function (PDF) of traditional probability theory. However, unlike the PDF function, they do not require any prior knowledge about the data distribution and represent both the spatial distribution of the samples and their frequency of occurrence [48]. The typicality function also resembles histograms of probability distributions and fuzzy membership functions [37].

4.2. Recursive Case

The calculations shown so far are applicable to online data streams when samples are being received in real time and the value of k is continuously increased. They are also applicable to previously stored datasets when it is desired to use the algorithm offline.

However, all of these definitions are global, i.e., it is necessary to have access to the entire dataset prior to the application of the algorithm (if offline) or to keep a local record of all samples that have already been received in the past (online case). Thus, its use in IoT scenarios and in edge computing devices is impractical because, as discussed in the previous sections, there are major limitations of data storage and processing power on these devices.

In this case, the recursive method of calculating the values of typicality and eccentricity is particularly useful. The mathematical proof of these calculations was presented for Euclidean and Mahalanobis distance in [52]. With this method, it is not necessary to keep previous samples stored locally; instead, the calculation of the metrics can be performed using only the last sample received and aggregate values that represent the state of the system at the previous instant ($k-1$).

This can be obtained by calculating the mean (μ) and the variance (σ^2) also in a recursive manner for each new sample received. This is shown in Equations (9) and (10) [37].

$$\mu_k(x) = \frac{k-1}{k}\mu_{k-1} + \frac{1}{k}x_k, \ \ \mu_1 = x_1 \tag{9}$$

$$\sigma_k^2(x) = \frac{k-1}{k}\sigma_{k-1}^2 + \frac{1}{k-1}\|x_k - \mu_k\|^2, \ \ \sigma_1^2 = 0 \tag{10}$$

With this, it is possible to use the recursive format of the equations to calculate the eccentricity and typicality, presented in Equations (11) and (12), respectively [37].

$$\xi_k(x) = \frac{1}{k} + \frac{(\mu_k - x_k)^T(\mu_k - x_k)}{k\sigma_k^2} \tag{11}$$

$$\tau_k(x) = 1 - \xi_k(x) = \frac{k-1}{k} - \frac{(\mu_k - x_k)^T(\mu_k - x_k)}{k\sigma_k^2} \tag{12}$$

This recursive method used to update the values of eccentricity and typicality makes the execution of the algorithm very computationally efficient. In addition, even without the previous samples having to be stored locally, the calculations are performed in an exact manner and not by some approximation. As a result, there is no loss of precision in the process of updating the metrics [52]. Thus, the TEDA metrics' calculations are ideal for applications with edge computing systems and real-time processes that have major limitations in computational resources and data storage.

4.3. Anomaly Detection

The Chebyshev inequality is a statistical principle widely used to detect anomalies. Assuming a sufficiently large set of samples from any data distribution, Chebyshev's inequality ensures that no more than $1/m^2$ samples will be more than an $m\sigma$ distance away from the mean [56].

In [37], the authors demonstrated that it is possible to derive a condition that provides exactly the same result (but without making assumptions about the amount of data, their independence, and so on) as Chebyshev's inequality, but represented in terms of TEDA. This new inequality can be seen in Equation (13).

$$\zeta_k(x) > \frac{m^2+1}{2k} \tag{13}$$

This formula, called the eccentricity inequality [53], allows local and individual analyses to be carried out for each new sample obtained in an online data stream. For this, it is only a matter of calculating the normalized eccentricity of every new sample, and if it exceeds the limit, the sample is classified as an anomaly.

5. The Proposed Algorithm

With the theory presented in Section 4 in mind, this paper aimed to propose the development of a novel evolving algorithm to perform local and online data compression on smart, IoT-enabled devices. This new algorithm is called the Tiny Anomaly Compressor (TAC).

Being built on top of the concepts of typicality and eccentricity presented earlier, the algorithm does not require previously established mathematical models or any prior assumptions about the data distribution. In addition, it is based on recursive calculations, which makes it an efficient algorithm, with low computational cost and low memory usage and processing power [37]. The only requirement for the algorithm to perform well is that the time series samples present some degree of spatial or time correlation among each other, which is exactly the case for the data stream generated by a sensor measuring some variable over time.

In this section, the basis for the established TEDA anomaly detection process is initially presented. As the algorithm was developed using this process, it is important to show how it works before presenting the full compression algorithm. Then, the logic behind the TAC algorithm and its functionality are shown, along with its pseudocode and basic execution flow diagram.

5.1. Anomaly Detection with TEDA

The general sequence of steps necessary to perform the anomaly detection with TEDA are as follows: for each new sample received, the internal mean and variance metrics are updated, then the sample eccentricity is calculated, and making use of the eccentricity inequality presented in Equation (13), it is classified as an anomaly or not.

The only hyperparameter needed is the value of m to be used in the eccentricity inequality evaluation. This hyperparameter can be interpreted as the sensitivity of the model to the presence of anomalies. From this base structure, it is possible to derive a series of more specialized algorithms, such as clustering [50] and, now, a lossy compression one defined in this work.

5.2. TAC Algorithm

The TAC algorithm, as the name suggests, makes use of the anomaly detection process to perform the time series compression. However, there are some peculiarities in relation to the basic structure presented above.

The first of these differences is the introduction of the concept of a rolling and dynamic "anomaly window". To do this, we introduced a counter variable that counts how many anomalies have been found since the last saved sample. To control the behavior of these windows, a new hyperparameter needs to be introduced: the *window_limit*. With it, the user can control the number of anomalies that should be detected before the window is considered "full".

The big difference of the algorithm execution arises as a result of this concept of the "window of anomalies". This happens because, whenever the window is not yet full, no new samples should be saved. However, once the window is filled and a new anomaly is detected, this sample should be saved. At this step, the model is said to be detecting a possible concept drift in the series. At this point, the window should be reset and the model's internal parameters (k, mean, and variance) returned to their initial values to start a new window.

The window is said to be dynamic because it has no fixed size in regard to the total number of data samples included in them. The window is only controlled by the number of anomalies detected. Therefore, the total number of samples in each of them is determined by the intrinsic variability of the dataset.

The *window_limit* hyperparameter can be interpreted as the response delay of the model to a concept drift in the signal. For higher values of *window_limit*, for example, the model has to detect a larger amount of anomalous data points to trigger an internal parameter reset, thus detecting a concept drift and saving a new sample.

Finally, the pseudocode of the TAC algorithm is given in Algorithm 1.

5.3. Auto-Definition of the m Hyperparameter

Another difference introduced to the TAC algorithm is the ability to dynamically auto-define the m hyperparameter if the user chooses to do so. This possibility leaves the model with only one controllable hyperparameter (the *window_limit*) to be set. To implement this feature, another concept of traditional probability theory was borrowed: the calculation of the kurtosis of the distribution.

In traditional probability theory, the kurtosis is a measure of the tendency of the distribution to contain anomalies (long tail). It is a scaled version of the distribution's forth moment [57]. A similar metric could be calculated from the recursive TEDA variables, as shown in Equation (15).

$$fourthMoment_k(x) = \frac{k-1}{k}\, fourthMoment_{k-1} + \frac{1}{k-1}\|x_k - \mu_k\|^4 \tag{14}$$

$$kurtosis_k(x) = \frac{fourthMoment_k}{\sigma_k^4} \tag{15}$$

In early experimental testing, it was found that the performance of the algorithm would degrade quickly for values of m greater than 1.2. As a result, it was chosen to determine its value following the condition shown in Equation (16).

$$m = \begin{cases} |kurtosis_k| & \text{if } |kurtosis_k| \leq 1 \\ \frac{1}{|kurtosis_k|} & \text{otherwise} \end{cases} \tag{16}$$

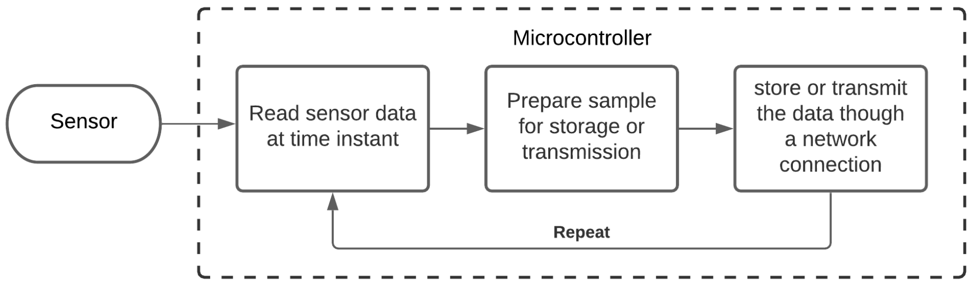
Finally, the value of m can be recursively calculated for each new sample before the evaluation with the eccentricity inequality.

Algorithm 1: Tiny anomaly compressor.

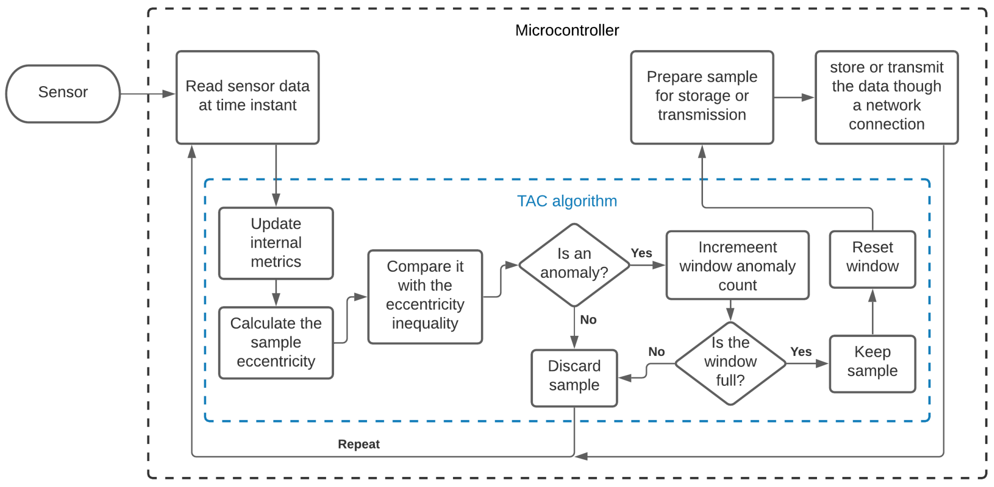
```
input  : Data stream $X$, Float $m$, Integer $window\_limit$
output: keep_point
procedure TAC(Data stream X, Integer window_limit)
    while $X$ is active do
        read sample $x_k \in X$
        if $k == 1$ then
            $\mu_k = x_k$
            $[\sigma^2]_k^x = 0$
            if $time == 1$ then
                keep_point = true
            else
                keep_point = false
            end
        else
            $\mu_k = \frac{k-1}{k}\mu_{k-1}^x + \frac{x_k}{k}$
            $[\sigma^2]_k^x = \frac{k-1}{k}[\sigma^2]_{k-1}^x + \frac{\|x_k-\mu_k\|^2}{k-1}$
            if $x == last\_value$ and $\sigma^2 == 0$ then
                keep_point = false
            else
                $\xi_k(x) = \frac{1}{k} + \frac{(\mu_k^x - x_k)^T(\mu_k^x - x_k)}{k[\sigma^2]_k^x}$
                $\zeta_k(x) = \frac{\xi_k(x)}{2}$
                if $\zeta_k(x) > \frac{m^2+1}{2k}$ then
                    is_anomaly = true
                    anomaly_count = anomaly_count + 1
                else
                    is_anomaly = false
                end
                if $anomaly\_count >= window\_limit$ then
                    resetWindow()
                    keep_point = true
                else
                    keep_point = false
                end
            end
        end
        time = time + 1
        k = k + 1
        last_point = x
    end
```

5.4. Execution Flowchart

As a means of clearly demonstrating the execution steps of the TAC model when inserted into a real scenario, two flowcharts can be observed in Figure 2. In Figure 2a, a general representation of the data acquisition and transmission process carried out by a microcontroller without the use of a compression algorithm can be seen. In contrast, in Figure 2b, it is possible to see the additional steps performed when the TAC model was deployed.


Microcontroller
Sensor
Read sensor data at time instant
Prepare sample for storage or transmission
store or transmit the data though a network connection
Repeat

(**a**)


Microcontroller
Sensor
Read sensor data at time instant
Prepare sample for storage or transmission
store or transmit the data though a network connection
TAC algorithm
Update internal metrics
Calculate the sample eccentricity
Compare it with the eccentricity inequality
Is an anomaly?
Yes
No
Increment window anomaly count
Reset window
Is the window full?
No
Yes
Discard sample
Keep sample
Repeat

(**b**)

Figure 2. Execution flow. (**a**) Data flow without the TAC model; (**b**) Data flow with the TAC model.

6. Case Study Definition

The case study aimed to evaluate the impact of the three different lossy compression algorithms and the fidelity of the recovered data after restoration/decompression. With this, we aimed to validate the newly proposed TAC algorithm. This section presents the benchmark test that was developed, with a focus on the dataset selection, objective definition, evaluation metrics, and execution process.

6.1. Dataset Selection

The datasets used in this work were obtained from a huge experiment currently ongoing at the eLUX laboratory at the University of Brescia, Italy. eLUX is a living laboratory for smart energy, smart grids, and smart environments for Industry 4.0 [58]. The laboratory concentrates and analyzes data coming from many distributed plants and from a huge quantity of sensors, meters, and actuators of different types, implementing an excellent example of the smart grid based on industrial-grade IoT systems. The permanent data storage and the specific web services made all the information available to researchers and other authorized users.

In this work, one of the use cases of the eLUX laboratory was considered. Since 2017, eLUX has run the energy supervision and management system of the entire Engineering Campus [59]. In particular, the data coming from one of the (four) Photovoltaic (PV) plants of the campus were used. This PV plant is powered by a string of 15 PV panels in series, whose current and voltage are monitored by innovative IoT meters with a sampling interval of 5s. Please note that the usual sampling time of standard commercial meters is 15 min, while in the eLUX experiments, a 90× oversampling is used. Such a high sampling rate enables the use of innovative performance analysis algorithms and predictive maintenance (based on neural network processing). When this approach is scaled to the entire campus (or an entire smart city), such a high sampling rate may generate huge quantities of data, requiring efficient compressing algorithms, as the one proposed in this paper.

The first dataset was obtained by extracting 50,879 values from the database of the PV string voltage, while the second dataset included the corresponding 50,879 values of the PV string current. Further details about the data statistical properties can be seen in Table 2.

Table 2. Basic statistical properties of the dataset considered in the case study.

Dataset	Samples	Min	Max	Mean	Standard Deviation	Skewness
Voltage	50,879	0.0	231.0	169.739	39.198	−2.633
Current	50,879	0.0	7.2	2.627	2.119	0.374

6.2. *Objective Definition*

The main goal was to conduct a base benchmark test to analyze the performance of the newly developed TAC data compression algorithm for IoT sensor data streams.

For further validation, two other algorithms found in the literature were used for comparison: SDT and the DCT. The test targeted assessing the performance differences of the three algorithms to compress the continuous data streams presented before in a simulated scenario. In this context, we aimed to find the maximum compression rate that could be achieved by the algorithms and which of them had a better overall performance in the analyzed scenario.

6.3. *Evaluation Metrics*

To evaluate the compression results of the tests, several metrics were chosen. This was necessary as the lossy compression process needs to be evaluated in regard to both the resulting Compression Error (CE), as well as the achieved Compression Rate (CR). As the performance on these two fronts tends to be inversely proportional, both of them had to be considered in the evaluation.

Furthermore, in order to highlight the different characteristics of each of the algorithms, the extra metrics of the Peak-Signal-to-Noise Ratio (PSNR) and the Normalized Cross-Correlation (NCC) were selected. Finally, to perform the hyperparameter optimization, a novel metric was introduced that took into account both the CR and CE of the models.

The definitions of all these metrics are as follows.

6.3.1. Compression Error

The compression error measures the difference observed between the original time series and the reconstructed data after the compression and decompression processes. To generate the decompressed series for the TAC and SDT compressor, simple linear interpolation was performed to fill the gaps between the saved samples. For the DCT compressor, on the other hand, the inverse DCT transformation was applied to the saved coefficients to obtain an approximation of the original signal.

To measure the error between the original signal and the decompressed signal, the primary metrics chosen were the Mean Absolute Error (MAE) and the Mean-Squared Error (MSE), which are shown in Equations (17) and (18).

$$MAE = \frac{1}{n}\sum_{i=1}^{N}|x_i - \hat{x}_i| \tag{17}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{18}$$

where x_i are the samples of the original signal, $\hat{x}_i$ are the samples of the decompressed signal, and n is the total number of samples.

6.3.2. Compression Rate

As is the case for measuring the error, there are several approaches to quantify the effectiveness of the compression on the size reduction. For consistency across the algorithms, the CR was measured as the percentage of size reduction between the original file and the file with the compressed signal representation. This value is calculated as shown in Equation (19).

$$CR = \left(1 - \frac{size(compressedFile)}{size(originalFile)}\right) \times 100 \tag{19}$$

where $size()$ is the function that returns the file size in KB.

To keep everything consistent, after every test, both the original and the generated compressed time series were saved in a standardized form as different CSV files. Each file only contained two columns: one for the measured sensor value on the given timestamps and another for the respective sample indices (used for the data reconstruction with linear interpolation). For the special DCT case, the selected coefficients returned by the model were saved instead of raw sample values.

6.3.3. Peak-Signal-to-Noise Ratio

The PSNR was also calculated. The PSRN is a metric used to evaluate the effect of noise in the signal representation. Its a commonly applied technique in fields such as digital image processing, where it can be used to measure the quality of images after passing through noise reduction filters [60].

The PSNR is presented in decibels (dB) using a logarithmic scale, and usually, higher values mean less presence of noise and, thus, a better representation. It can be used to highlight the denoising capability of the models and is calculated as shown in Equation (20).

$$PSNR = 10 \times \log_{10}\left(\frac{(max(x) - min(x))^2}{MSE(x, \hat{x})}\right) \tag{20}$$

where $MSE()$ is the mean-squared error and $max()$ and $min()$ are functions that return the maximum and minimum values of the original signal, respectively.

6.3.4. NCC

The Normalized Cross-Correlation (NCC) coefficient is a measure of the similarity between two signals. It is another metric common in the fields of signal and image processing [61,62]. Its value is limited to between negative one and one. As a result, in a perfect scenario where the decompressed signal would be exactly the same as the original, for example, their NCC would be exactly one. It can be calculated as shown in Equation (21).

$$NCC = \frac{1}{n}\sum_{i=1}^{N}\frac{(x_i - \mu_x)(\hat{x}_i - \mu_{\hat{x}})}{\sigma_x * \sigma_{\hat{x}}} \tag{21}$$

6.4. Proposed CF Score

For a balanced operation, both the CR and CE should be considered when selecting optimal compressor hyperparameters. With that in mind, we proposed a novel metric inspired by the F_β score for statistical analysis.

In traditional probability theory, the F_β score is a weighted average metric between the precision and recall rates of a classifier [63]. The β in the F_β score is a user-defined parameter that controls the importance of the rates in the calculation. Additionally, the metric computation is based on the harmonic mean formula. Therefore, when both rates have an equal weight (β = 1), the F_1 score reverts to being a simple harmonic mean between the two values used [63].

In turn, the harmonic mean is an average measure generally considered as a fusion technique of numerical data information [64]. It differs from the traditional arithmetic mean by giving more weight to smaller values being averaged and is preferred when averaging rates or ratios [65].

In the compression case, the CR is already a ratio between zero and one. However, the error magnitudes are highly dependent on the data scale; thus, it is not a normalized value. As a replacement, we proposed to use the NCC, and instead of trying to minimize the error, we tried to maximize the similarity between the signals.

With all that in mind, the so-called Compression F_β score (CF_β score) can be calculated as shown in Equation (22).

$$CF_\beta = (1+\beta^2)\frac{CR * NCC}{(\beta^2 * CR) + NCC} \tag{22}$$

where β is a positive real parameter used to adjust the importance of each metric in the CF_β calculation. The parameter was chosen such that the NCC was β times more important than the CR. Finally, when $\beta = 1$, the equation puts equal weight on both metrics and turns into a simple harmonic mean.

This proposed CF_β score is somewhat similar to the Compression Criteria proposed in [11], where they proposed the use of a harmonic mean between the compression rate and error. However, CF_β takes into account the preferred use of rates when calculating the harmonic mean by replacing the error with the NCC score. It also is inspired by the F_β score and adds the β parameter to adjust the weight of the two metrics.

6.5. Design and Operation of the Benchmark Test

To perform the benchmark test, the newly developed TAC algorithm (described in detail in Section 5) was used alongside two other well-established time series compression algorithms found in the literature, SDT and the DCT, both of which were briefly described in Section 2. The test was carried out in an offline manner, as all the data were already previously acquired.

The first barrier encountered in the design of the test was the different hyperparameters available to tune in each of the models. Since they were not the same across all three models, it was not possible to fix a single value for all of them to standardize the tests. This being the case, the first step was to perform a grid-search on each of the models to find good hyperparameter values. This consisted of executing the model multiple times on the dataset with different combinations of hyperparameters. This was performed to find the optimal configuration for each of them on the respective datasets. With the ideal hyperparameter combination, the algorithms had the best "all around" performance on the data and should be more easily comparable to each other in a benchmark scenario.

The second barrier arose from the execution of the grid-search itself. When trying to optimize hyperparameters, a metric needs to be chosen to be minimized or maximized to assess the performance. However, in the case of lossy compression, there are two fundamental metrics that have to be taken into account, the CR and CE, and they are essentially inversely proportional. For example, if the parameters are chosen so that the CR is maximized, the resulting error will be high, and the result might lose too much information; inversely, if the CE is chosen to be minimized, the model will achieve poor compression results.

For this reason, it was chosen to develop and use the CF_β score when searching for optimized hyperparameters. When trying to maximize this metric, both the compression rate and the similarity of the resulting signal were taken into account. In this manner, it was possible to find a more balanced result for each model performance. Additionally, the use of the CF_β score on the grid-search enabled the use of different values of β to change the test configuration for all models at once.

After the grid-search was performed, the models were executed on the dataset with their best hyperparameters. Then, the evaluation metrics presented in the beginning of this section were applied to each of the results, generating a performance report for each model that was then compiled to be analyzed.

This entire process was performed for the two datasets described at the beginning of this section. Furthermore, to achieve more variability in the results, the entire process was repeated twice for each dataset, once using $\beta = 1$ and once using $\beta = 4$. With this, it was possible to observe two different behaviors for the models in each dataset. When $\beta = 1$, both the compression rate and similarity were considered to have the same weight when finding optimal parameters. However, when $\beta = 4$, the similarity was considered four-times more important than the compression rate, leading to results with less error.

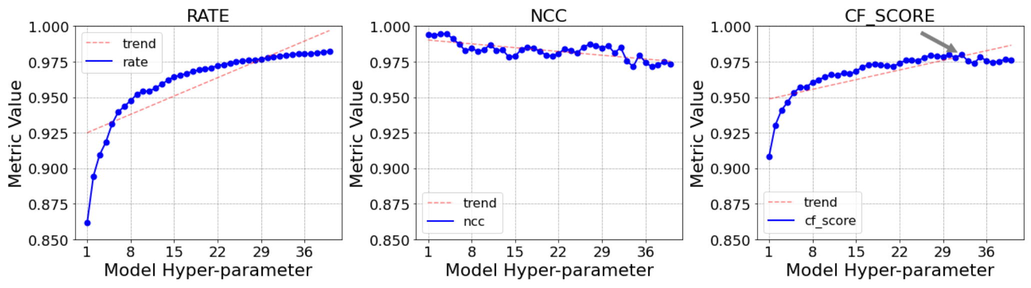
In all of these rounds, the DCT and SDT models were executed once and the TAC model was executed twice: one with both hyperparameters being defined and one where the m hyperparameter was defined dynamically by the model itself. At the end of each round, the performance metrics of the algorithms were collected for each dataset and the different values of β that were chosen.

7. Results and Discussion

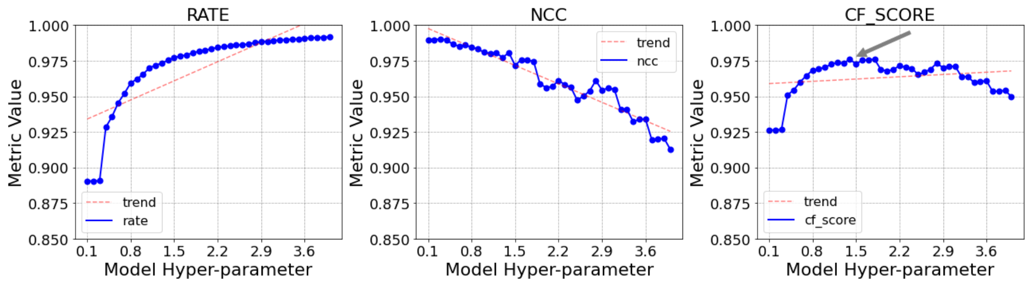
To present the results in a clear manner, it was decided to separate the discussion for each dataset used in the case study. First, the results regarding the voltage dataset are shown and, finally, the results for the current dataset. The first dataset consisted of 50,879 samples of voltage measurements, whereas the second dataset had the same amount of samples of electrical current measurements.

7.1. Dataset 1: Electrical Voltage

As stated earlier, the first step was to perform a grid-search to find the optimal hyperparameters for each algorithm. The process was performed by repeatedly running the models on the data with different hyperparameters, then choosing the configuration that yielded the best CF_{β}. As an example, in Figure 3, it is possible to see the result of the process graphically for the SDT and TAC (with dynamic m) models when $\beta = 1$. The figure illustrates the variation of the compression rate, NCC, and CF score obtained when varying the hyperparameter for each of the models. It is also possible to see the point where the maximum value of the CF score was achieved, marked with a grey arrow.



(**a**)



(**b**)

Figure 3. Variation of the compression rate, NCC, and CF score for the grid-search evaluated by varying the hyperparameter for the TAC and SDT models. $\beta = 1$. (**a**) TAC (Auto-m); (**b**) SDT.

As is possible to see, the CR values tended to increase following a logarithmic trend, while the NCC, on the other hand, decreased following a somewhat linear (albeit noisy) tendency. For this reason, it was also decided to test the models with a greater value of β, so that the NCC would have a larger effect on the CF_β calculation and, thus, the hyperparameter definition.

In Table 3, it is possible to see the optimal hyperparameters found for every model for both $\beta = 1$ and $\beta = 4$.

Table 3. Optimal hyperparameters: Dataset 1.

Algorithm	Hyperparameters for $\beta = 1$	Hyperparameters for $\beta = 4$
TAC	$window\_limit = 35, m = 0.2$	$window\_limit = 7, m = 0.1$
TAC (Auto-m)	$window\_limit = 32$	$window\_limit = 4$
SDT	$comp\_deviation = 1.4$	$comp\_deviation = 0.4$
DCT	$delta = 0.998$	$delta = 0.999$

It is possible to observe a good difference in the parameters for both the TAC and SDT models, but not so much in the DCT. This could hint at greater flexibility of the SDT and TAC models in relation to the DCT. Finally, with the optimal hyperparameters defined, the models were executed again with their respective configurations. The compression results for $\beta = 1$ and $\beta = 4$ can be seen in Table 4.

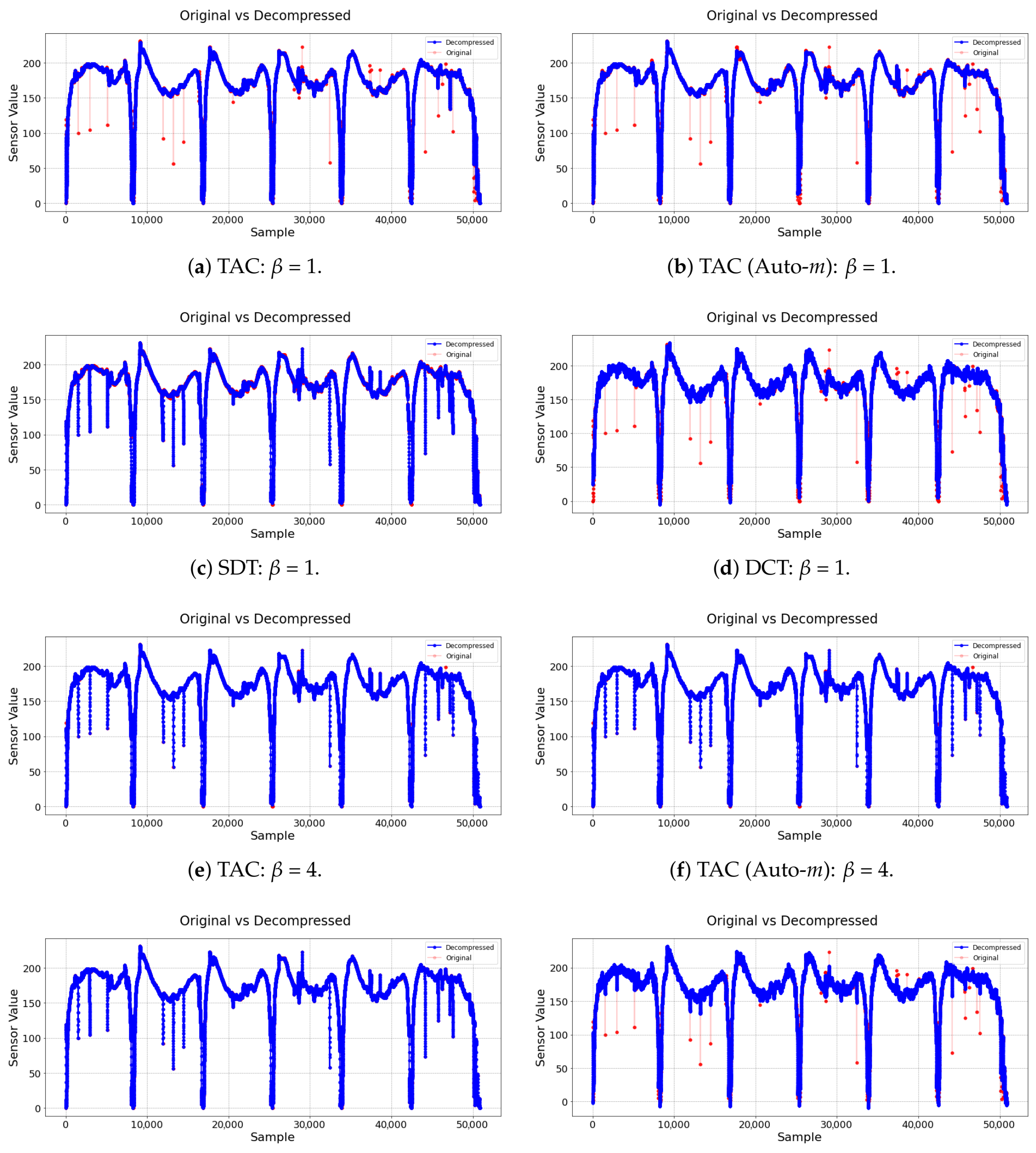
Table 4. Compression results: Dataset 1.

β	Algorithm	CR (%)	NCC	CF_1 Score	MSE	MAE	PSNR
1	TAC	97.4	0.9865	0.9802	41.3406	1.4593	31.1085
	TAC (Auto-m)	97.49	0.9849	0.9799	46.2367	1.6009	30.6224
	SDT	97.15	0.9807	0.9761	60.2682	2.3412	29.4714
	DCT	98.8	0.9801	0.984	60.6456	4.2092	29.4442
4	TAC	91.5	0.9948	0.9898	16.0263	0.738	35.2239
	TAC (Auto-m)	90.31	0.9887	0.9887	16.7838	0.6913	35.0233
	SDT	91.56	0.9896	0.9849	32.0155	1.2412	32.2186
	DCT	96.8	0.9901	0.9888	30.3387	2.9515	32.4523

The TAC model (both with dynamic m and not) outperformed the two other models in regard to the total error and NCC in all cases. They also achieved greater values of the PSNR, indicating less noise in the decompressed signal. Additionally, for $\beta = 1$, the TAC also achieved a greater compression rate than SDT, only losing to the DCT by a small margin. In the scenario where $\beta = 4$, the TAC and SDT resulted in similar compression rates, again losing by a little to the DCT.

Additionally, in Figure 4, the original and decompressed signals are displayed. The original signal is plotted in red, and the resulting decompressed signal is on top of it in blue. In Figure 4a–d, it can be noted that the TAC model (both with dynamic m and not) was able to ignore most of the samples that were big outliers, reducing the signal noise. SDT, which uses linear trends to classify the samples, was not able to do the same. The DCT, in turn, was also able to ignore most of the clear outliers; however, it also seemed to introduce a little noise in sections where the signal should be smoother. This can be attributed to a byproduct of the signal reconstruction process by cosines when applying the inverse DCT.

Finally, in Figure 4e–h, it is possible to see that, when $\beta = 1$ and the NCC were taken into account better, the TAC no longer removed the outliers, as it was more focused on maximizing the signals' similarity. However, it was still able to achieve much less compression error than the other two, and without sacrificing too much of the compression rate.



(**a**) TAC: β = 1. (**b**) TAC (Auto-m): β = 1.

(**c**) SDT: β = 1. (**d**) DCT: β = 1.

(**e**) TAC: β = 4. (**f**) TAC (Auto-m): β = 4.

(**g**) SDT: β = 4. (**h**) DCT: β = 4.

Figure 4. Original and decompressedsignals: Dataset 1.

7.2. Dataset 2: Electrical Current

The same sequence of steps was repeated for the second dataset. The optimal hyperparameters found for each model can be seen in Table 5.

Table 5. Optimal hyperparameters: Dataset 2.

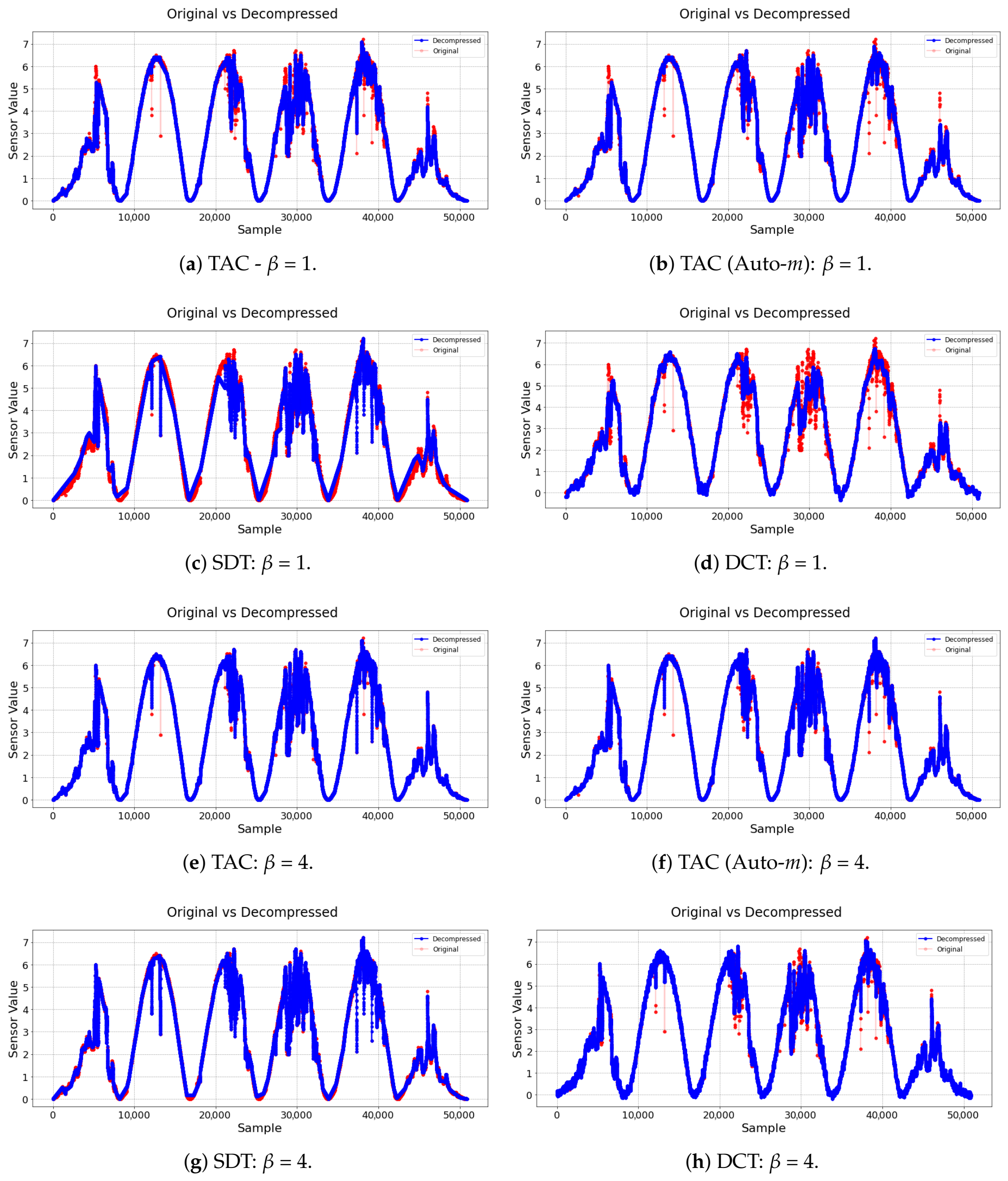
Algorithm	**Hyperparameters for $\beta = 1$**	**Hyperparameters for $\beta = 4$**
TAC	$window\_limit = 40$, $m = 0.6$	$window\_limit = 15$, $m = 0.1$
TAC (Auto-m)	$window\_limit = 40$	$window\_limit = 12$
SDT	$comp\_deviation = 0.3$	$comp\_deviation = 0.1$
DCT	$delta = 0.992$	$delta = 0.997$

The compression results for $\beta = 1$ and $\beta = 4$ can be seen in Table 6. It can be observed that very similar results to the first experiment were obtained, where the TAC model (both with dynamic m and not) consistently achieved lower values of the error than the two competitors.

Table 6. Compression result: Dataset 2.

β	Algorithm	CR (%)	NCC	CF$_1$ Score	MSE	MAE	PSNR
1	TAC	98.33	0.9919	0.9876	0.0723	0.1038	28.5547
	TAC (Auto-*m*)	98.24	0.9916	0.987	0.0753	0.101	28.377
	SDT	98.98	0.9883	0.989	0.1263	0.2671	26.1318
	DCT	99.48	0.9899	0.9923	0.0908	0.1845	27.5638
4	TAC	95.26	0.9972	0.9944	0.0256	0.0542	33.0617
	TAC (Auto-*m*)	95.37	0.9963	0.9937	0.0337	0.0602	31.8695
	SDT	97.2	0.9958	0.9943	0.0392	0.1066	31.2167
	DCT	97.31	0.9962	0.9948	0.0342	0.1117	31.8127

Similar to the other experiment, the original and decompressed signals can be seen in Figure 5 for both $\beta = 1$ and $\beta = 4$, respectively.



(**a**) TAC - β = 1. (**b**) TAC (Auto-*m*): β = 1.

(**c**) SDT: β = 1. (**d**) DCT: β = 1.

(**e**) TAC: β = 4. (**f**) TAC (Auto-*m*): β = 4.

(**g**) SDT: β = 4. (**h**) DCT: β = 4.

Figure 5. Original and decompressed signals: Dataset 2.

8. Threats to Validity

The threats to validity for the present study were the following:

- Construct validity: The construction validity verifies the relationship between the theory behind the experiments carried out and the observations found. This threat is mainly related to the algorithms used in the experiment. Therefore, we were confident that the descriptions and pseudocodes found for the studied algorithms were correct since the results were satisfactory.
- Internal validity: Internal threats are relatively linked to the experiment since it is necessary to define different parameters for the algorithms. Thus, to mitigate this threat, a grid-search was performed on each of the models analyzed.
- External validity: External threats refer to the ability to generalize our findings and conclusions appropriately to other contexts. In this study, the experiments were carried out with large sets of data (voltage and current), which allowed us to analyze the compression. Although the tests were executed only on two datasets, we were confident that the results for other sets were similar.

9. Conclusions

This paper proposed an evolving, dynamic, and lightweight data compression algorithm for IoT environments. This is a fast-growing field, as applications with IoT features are growing in number exponentially, bringing with them new challenges due to the increase of both the connected sensors and the data generated by them. In order to achieve this objective, a benchmark test for the validation of the novel TAC algorithm was introduced. Moreover, two other algorithms from the literature (SDT and DCT) were used for comparative analysis, from which the results were shown in the last section.

The TAC model is based on data eccentricity and is “empirical” in nature, as it does not require any previous assumptions for the underlying data distribution or statistical properties and learns strictly from the data’s spatial and temporal relations. Furthermore, unlike the SDT model, which uses linear trends, the TAC model has no predefined shape and is adjusted as the continuous data stream progresses. On top of that, the TAC model is based on simple and recursive calculations, making it lightweight both in regard to the storage space and processing power needed for execution.

Regarding the maximum compression rate that can be achieved by the algorithms, the TAC model was able to reach a maximum CR of 97.49% with a 1.6 MAE in the first dataset and a maximum CR of 98.33% with a 0.1 MAE for the second dataset. On the other hand, SDT achieved a CR of 97.15% with a 2.33 MAE on the first dataset and a CR of 98.98% with a 0.26 MAE on the second. Finally, the DCT obtained a CR of 98.8% with a 4.2 MAE for the first dataset and a CR of 99.48% with a 0.18 MAE on the second.

For this reason, it was concluded that the best performing model in the context analyzed was the newly proposed TAC algorithm. It obtained comparable compression rates to the other two models, even beating them in some cases, while consistently managing to deliver less error on the output signal. Sometimes, the DCT attained slightly larger compression rates, but it cannot be executed online and in a point-by-point analysis scenario, as the TAC does.

In conclusion, the TAC algorithm is a novel and interesting contribution that deserves major attention.

Future work includes, but is not limited to: expanding the compression algorithm to work for multiple variables simultaneously; and evaluating the impact on the performance of the TAC algorithm on an embedded IoT device.

Author Contributions: Formal analysis, G.S. and M.S.; project administration, I.S.; software, G.S. and M.S.; writing, original draft, G.S., M.S. and P.A.; research supervisors, I.S., E.S. and P.F. All authors have read and agreed to the published version of the manuscript.

Funding: This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil (CAPES), Finance Code 001, and we thank the Brazilian fostering agency CNPq (National Council for Scientific and Technological Development), Process No. 435683/2018-7.

Data Availability Statement: Not applicable.

Acknowledgments: We thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and the National Council for Scientific and Technological Development (CNPq).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AMMMO	Adaptive Multi-Model Middle-Out
CD	Compression Deviation
CE	Compression Error
CR	Compression Rate
DCT	Discrete Cosine Transform
IDEALEM	Dynamic Extensible Adaptive Locally Exchangeable Measures
DWT	Discrete Wavelet Transform
EDA	Empirical Data Analysis
EMA	Exponential Moving Average
FWHT	Fast Walsh–Hadamard Transform
IoT	Internet of Things
KPI	Key Performance Indicators
MAE	Mean Absolute Error
ML	Machine Learning
MSE	Mean-Squared Error
NCC	Normalized Cross-Correlation
PSNR	Peak-Signal-to-Noise Ratio
RSME	Root Mean-Squared Error
SDT	Swing Door Trending
TAC	Tiny Anomaly Compressor
TEDA	Typicality and Eccentricity Data Analytics

References

1. Marafie, Z.; Lin, K.; Zhai, Y.; Li, J. ProActive Fintech: Using Intelligent IoT to Deliver Positive InsurTech Feedback. In Proceedings of the 2018 IEEE 20th Conference on Business Informatics (CBI), Vienna, Austria, 11–14 July 2018; Volume 2, pp. 72–81.
2. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of things for smart cities. *IEEE Internet Things J.* **2014**, *1*, 22–32. [CrossRef]
3. Diniz da Silva, M.B.; Vieira, E.; Silva, I.; Silva, D.; Ferrari, P.; Rinaldi, S.; Carvalho, D.F. A customer feedback platform for vehicle manufacturing in Industry 4.0. In Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, 25–28 June 2018; pp. 1249–1254.
4. Al Homssi, B.; Al-Hourani, A.; Magowe, K.; Delaney, J.; Tom, N.; Ying, J.; Wolf, H.; Maselli, S.; Kandeepan, S.; Wang, K.; et al. A Framework for the Design and Deployment of Large-Scale LPWAN Network for Smart Cities Applications. *IEEE Internet Things Mag.* **2020**. [CrossRef]
5. Silva, M.; Vieira, E.; Signoretti, G.; Silva, I.; Silva, D.; Ferrari, P. A Customer Feedback Platform for Vehicle Manufacturing Compliant with Industry 4.0 Vision. *Sensors* **2018**, *18*, 3298. [CrossRef]
6. Zhang, Z.; Li, C.; Peng, S.; Pei, X. A new task offloading algorithm in edge computing. *EURASIP J. Wirel. Commun. Netw.* **2021**, *2021*, 17. [CrossRef]
7. Banbury, C.; Zhou, C.; Fedorov, I.; Matas, R.; Thakker, U.; Gope, D.; Janapa Reddi, V.; Mattina, M.; Whatmough, P. Micronets: Neural Network Architectures for Deploying Tinyml Applications on Commodity Microcontrollers. Available online: https://proceedings.mlsys.org/paper/2021/file/a3c65c2974270fd093ee8a9bf8ae7d0b-Paper.pdf (accessed on 16 June 2021).
8. Magsi, H.; Sodhro, A.H.; Zahid, N.; Pirbhulal, S.; Wang, L.; Al-Rakhami, M.S. A Novel Adaptive Battery-Aware Algorithm for Data Transmission in IoT-Based Healthcare Applications. *Electronics* **2021**, *10*, 367. [CrossRef]
9. Azar, J.; Makhoul, A.; Barhamgi, M.; Couturier, R. An energy efficient IoT data compression approach for edge machine learning. *Future Gener. Comput. Syst.* **2019**, *96*, 168–175. [CrossRef]

10. Lounas, R.; Salhi, D.E.; Mokrani, H.; Djerbi, R.; Bennai, M.T. Towards a Smart Data Transmission Strategy for IoT Monitoring Systems: Application to Air Quality Monitoring. In Proceedings of the 2019 International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS), Skikda, Algeria, 15–16 December 2019; Volume 1, pp. 1–7.
11. Correa, J.D.A.; Pinto, A.S.R.; Montez, C.; Leão, E. Swinging Door Trending Compression Algorithm for IoT Environments. In *Anais do IX Simpósio Brasileiro de Engenharia de Sistemas Computacionais*; SBC: Porto Alegre, Brasil, 2019; pp. 143–148.
12. Signoretti, G.; Silva, M.; Araujo, J.; Silva, I.; Silva, D.; Ferrari, P.; Sisinni, E. A Dependability Evaluation for OBD-II Edge Devices: An Internet of Intelligent Vehicles Perspective. In Proceedings of the 2019 9th Latin-American Symposium on Dependable Computing (LADC), Natal, Brazil, 19–21 November 2019; pp. 1–9.
13. Sodhro, A.H.; Sangaiah, A.K.; Sodhro, G.H.; Lohano, S.; Pirbhulal, S. An energy-efficient algorithm for wearable electrocardiogram signal processing in ubiquitous healthcare applications. *Sensors* **2018**, *18*, 923. [CrossRef]
14. Yamaoka, H.; Itakura, K.; Takahashi, E.; Nakagawa, G.; Michaelis, J.; Kanemasa, Y.; Ueki, M.; Matsumoto, T.; Take, R.; Tanie, S.; Inoue, D. Dracena: A Real-Time IoT Service Platform Based on Flexible Composition of Data Streams. In Proceedings of the 2019 IEEE/SICE International Symposium on System Integration (SII), Paris, France, 14–16 January 2019; pp. 596–601.
15. Shukla, A.; Simmhan, Y. Benchmarking distributed stream processing platforms for iot applications. In Proceedings of the Technology Conference on Performance Evaluation and Benchmarking 2016, New Delhi, India, 5–9 September 2016; pp. 90–106.
16. Lu, T.; Xia, W.; Zou, X.; Xia, Q. Adaptively Compressing IoT Data on the Resource-Constrained Edge. Available online: https://www.usenix.org/conference/hotedge20/presentation/lu (accessed on 16 June 2021).
17. Uthayakumar, J.; Vengattaraman, T.; Dhavachelvan, P. A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. *J. King Saud Univ. Comput. Inf. Sci.* **2018**. [CrossRef]
18. Ferrari, P.; Sisinni, E.; Carvalho, D.F.; Signoretti, G.; da Silva, M.D.; Silva, I.; Silva, D. Using LoRaWAN smart city infrastructure as backup network for Industry 4.0 enabled vehicles. In *Anais do IX Simpósio Brasileiro de Engenharia de Sistemas Computacionais*; SBC: Porto Alegre, Brasil, 2019; pp. 119–124.
19. Sisinni, E.; Ferrari, P.; Fernandes Carvalho, D.; Rinaldi, S.; Pasetti, M.; Flammini, A.; Depari, A. A LoRaWAN range extender for Industrial IoT. *IEEE Trans. Ind. Inform.* **2019**, *16*, 1. [CrossRef]
20. Georgiou, O.; Raza, U. Low power wide area network analysis: Can LoRa scale? *IEEE Wirel. Commun. Lett.* **2017**, *6*, 162–165. [CrossRef]
21. Hu, Z.; Wang, D.; Li, Z.; Sun, M.; Wang, W. Differential Compression for Mobile Edge Computing in Internet of Vehicles. In Proceedings of the 2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Barcelona, Spain, 21–23 October 2019; pp. 336–341.
22. Nobre, M.; Silva, I.; Guedes, L.A.; Portugal, P. Towards a WirelessHART module for the ns-3 simulator. In Proceedings of the 2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010), Bilbao, Spain, 13–16 September 2010; pp. 1–4.
23. Fedorov, I.; Adams, R.P.; Mattina, M.; Whatmough, P. Sparse: Sparse architecture search for CNNs on resource-constrained microcontrollers. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 4978–4990.
24. Porambage, P.; Okwuibe, J.; Liyanage, M.; Ylianttila, M.; Taleb, T. Survey on multi-access edge computing for internet of things realization. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2961–2991. [CrossRef]
25. Mikhaylov, K.; Petrov, V.; Gupta, R.; Lema, M.A.; Galinina, O.; Andreev, S.; Koucheryavy, Y.; Valkama, M.; Pouttu, A.; Dohler, M. Energy Efficiency of Multi-Radio Massive Machine-Type Communication (MR-MMTC): Applications, Challenges, and Solutions. *IEEE Commun. Mag.* **2019**, *57*, 100–106. [CrossRef]
26. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358. [CrossRef]
27. Maitra, S.; Richards, D.; Abdelgawad, A.; Yelamarthi, K. Performance Evaluation of IoT Encryption Algorithms: Memory, Timing, and Energy. In Proceedings of the 2019 IEEE Sensors Applications Symposium (SAS), Sophia Antipolis, France, 11–13 March 2019; pp. 1–6.
28. Premsankar, G.; Di Francesco, M.; Taleb, T. Edge Computing for the Internet of Things: A Case Study. *IEEE Internet Things J.* **2018**, *5*, 1275–1284. [CrossRef]
29. Wang, X.; Magno, M.; Cavigelli, L.; Benini, L. FANN-on-MCU: An open-source toolkit for energy-efficient neural network inference at the edge of the Internet of Things. *IEEE Internet Things J.* **2020**, *7*, 4403–4417. [CrossRef]
30. Guberovic, E.; Krišto, F.; Krivic, P.; Cavrak, I. Assessing compression algorithms on IoT sensor nodes. In Proceedings of the 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 20–24 May 2019; pp. 913–918.
31. Mahdavinejad, M.; Rezvan, M.; Barekatain, M.; Adibi, P.; Barnaghi, P.; Sheth, A. Machine learning for internet of things data analysis: A survey. *Digit. Commun. Netw.* **2018**, *4*, 161–175. [CrossRef]
32. Gorospe, J.; Mulero, R.; Arbelaitz, O.; Muguerza, J.; Antón, M.Á. A Generalization Performance Study Using Deep Learning Networks in Embedded Systems. *Sensors* **2021**, *21*, 1031. [CrossRef]
33. Gopinath, S.; Ghanathe, N.; Seshadri, V.; Sharma, R. Compiling KB-sized machine learning models to tiny IoT devices. In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, Phoenix, AZ, USA, 22–26 June 2019; pp. 79–95.

34. Bengio, Y.; Courville, A.; Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [CrossRef]
35. Zoppi, T.; Ceccarelli, A.; Bondavalli, A. Into the Unknown: Unsupervised Machine Learning Algorithms for Anomaly-Based Intrusion Detection. In Proceedings of the 2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S), Valencia, Spain, 29 June–2 July 2020.
36. Angelov, P.P.; Gu, X. *Empirical Approach to Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2019.
37. Angelov, P. Anomaly detection based on eccentricity analysis. In Proceedings of the 2014 IEEE Symposium on Evolving and Autonomous Learning Systems (EALS), Orlando, FL, USA, 9–12 December 2014; pp. 1–8.
38. Signoretti, G.; Silva, M.; Araujo, J.; Guedes, L.A.; Silva, I.; Sisinni, E.; Ferrari, P. Performance Evaluation of an evolving data compression algorithm embedded into an OBD-II edge device. In Proceedings of the 2020 IEEE International Workshop on Metrology for Industry 4.0 IoT, Roma, Italy, 3–5 June 2020; pp. 696–701.
39. Yu, X.; Peng, Y.; Li, F.; Wang, S.; Shen, X.; Mai, H.; Xie, Y. Two-Level Data Compression using Machine Learning in Time Series Database. In Proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDE), Dallas, TX, USA, 20–24 April 2020; pp. 1333–1344.
40. Hamdan, S.; Awaian, A.; Almajali, S. Compression Techniques Used in Iot: A Comparitive Study. In Proceedings of the 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), Amman, Jordan, 9–11 October 2019; pp. 1–5.
41. Moon, A.; Kim, J.; Zhang, J.; Son, S.W. Evaluating fidelity of lossy compression on spatiotemporal data from an IoT enabled smart farm. *Comput. Electron. Agric.* **2018**, *154*, 304–313. [CrossRef]
42. Silva, I.M.D.; Guedes, L.A.; Vasques, F. Performance evaluation of a compression algorithm for wireless sensor networks in monitoring applications. In Proceedings of the 2008 IEEE International Conference on Emerging Technologies and Factory Automation, Hamburg, Germany, 15–18 September 2008; pp. 672–678.
43. Bristol, E.H. Swinging door trending: Adaptive trend recording. In Proceedings of the ISA National Conference, 1990; pp. 749–753. Available online: https://ci.nii.ac.jp/naid/10030606241/ (accessed on 16 June 2021).
44. Xu, X.-D.; Fu, Y.-P. The Research and Improvement of SDT Algorithm for Historical Data in SCADA. *ITM Web Conf.* **2017**, *11*, 01009. [CrossRef]
45. Neto, E.J.M.; Augusto, L.;Souza, D.C.; Guedes, L.A. Adaptive Swinging Door Trending: Um Algoritmo Adaptativo para Compressão de Dados em Tempo Real. In Proceedings of the Anais do XX Congresso Brasileiro de Automática, Belo Horizonte, Brazil, 20–24 September 2014.
46. Gibson, J.K.; Lee, D.; Choi, J.; Sim, A. Dynamic Online Performance Optimization in Streaming Data Compression. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 534–541.
47. Park, J.; Park, H.; Choi, Y. Data compression and prediction using machine learning for industrial IoT. In Proceedings of the 2018 International Conference on Information Networking (ICOIN), Chiang Mai, Thailand, 10–12 January 2018; pp. 818–820.
48. Angelov, P. Outside the Box: An Alternative Data Analytics Framework. *J. Autom. Mob. Robot. Intell. Syst.* **2014**, *8*, 29–35. [CrossRef]
49. Sielly Jales Costa, B.; Bezerra, C.G.; Guedes, L.A.; Angelov, P.P. Online fault detection based on Typicality and Eccentricity Data Analytics. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015; pp. 1–6.
50. Bezerra, C.G.; Sielly Jales Costa, B.; Guedes, L.A.; Angelov, P.P. A new evolving clustering algorithm for online data streams. In Proceedings of the 2016 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), Natal, Brazil, 23–25 May 2016; pp. 162–168.
51. Angelov, P.P.; Gu, X.; Gutierrez, G.; Iglesias, J.A.; Sanchis, A. Autonomous Data Density based clustering method. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 2405–2413.
52. Kangin, D.; Angelov, P.; Iglesias, J.A. Autonomously evolving classifier TEDAClass. *Inf. Sci.* **2016**, *366*, 1–11. [CrossRef]
53. Angelov, P. Typicality distribution function—A new density-based data analytics tool. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015; pp. 1–8.
54. Kangin, D.; Angelov, P.; Iglesias, J.; Sanchis de Miguel, A. Evolving classifier TEDAClass for big data. *Procedia Comput. Sci.* **2015**, *53*, 9–18. [CrossRef]
55. Martins, R.S.; Angelov, P.; Sielly Jales Costa, B. Automatic Detection of Computer Network Traffic Anomalies based on Eccentricity Analysis. In Proceedings of the 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
56. Saw, J.G.; Yang, M.C.K.; Mo, T.C. Chebyshev Inequality with Estimated Mean and Variance. *Am. Stat.* **1984**, *38*, 130–132.
57. Balanda, K.P.; MacGillivray, H. Kurtosis: A critical review. *Am. Stat.* **1988**, *42*, 111–119.
58. Flammini, A.; Pasetti, M.; Rinaldi, S.; Bellagente, P.; Ciribini, A.C.; Tagliabue, L.C.; Zavanella, L.E.; Zanoni, S.; Oggioni, G.; Pedrazzi, G. A Living Lab and Testing Infrastructure for the Development of Innovative Smart Energy Solutions: The eLUX Laboratory of the University of Brescia. In Proceedings of the 2018 AEIT International Annual Conference, Bari, Italy, 3–5 October 2018; pp. 1–6.
59. Rinaldi, S.; Pasetti, M.; Flammini, A.; Ferrari, P.; Sisinni, E.; Simoncini, F. A Testing Framework for the Monitoring and Performance Analysis of Distributed Energy Systems. *IEEE Trans. Instrum. Meas.* **2019**, *68*, 3831–3840. [CrossRef]

60. Jin, F.; Fieguth, P.; Winger, L.; Jernigan, E. Adaptive Wiener filtering of noisy images and image sequences. In Proceedings of the 2003 International Conference on Image Processing (Cat. No.03CH37429), Barcelona, Spain, 14–17 September 2003; Volume 3.
61. Yoo, J.C.; Han, T.H. Fast normalized cross-correlation. *Circuits Syst. Signal Process.* **2009**, *28*, 819–843. [CrossRef]
62. Tsai, D.M.; Lin, C.T.; Chen, J.F. The evaluation of normalized cross correlations for defect detection. *Pattern Recognit. Lett.* **2003**, *24*, 2525–2535. [CrossRef]
63. Hand, D.; Christen, P. A note on using the F-measure for evaluating record linkage algorithms. *Stat. Comput.* **2018**, *28*, 539–547. [CrossRef]
64. Xu, Z. Fuzzy harmonic mean operators. *Int. J. Intell. Syst.* **2009**, *24*, 152–172. [CrossRef]
65. Ferger, W.F. The nature and use of the harmonic mean. *J. Am. Stat. Assoc.* **1931**, *26*, 36–40. [CrossRef]