Direct Analytics of Generalized Deduplication Compressed IoT Data

Aaron Hurst, Qi Zhang and Daniel E. Lucani
DIGIT, Department of Electrical and Computer Engineering
Aarhus University
{ah, qz, daniel.lucani}@ece.au.dk

Ira Assent
DIGIT, Department of Computer Science
Aarhus University
ira@cs.au.dk

GLOBECOM 2021 - 2021 IEEE Global Communications Conference | 978-1-7281-8104-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/GLOBECOM46510.2021.9685589

***Abstract*—Given the ever increasing volume of data generated by the Internet of Things, data compression plays an essential role in reducing the cost of data transmission and storage. However, it also introduces a barrier, namely decompression, between users and the data-driven insights they require. We propose methods for *direct* analytics of compressed data based on the Generalized Deduplication compression algorithm. When applied to data clustering, the accuracy of the best performing method differs by merely 1-5% when compared to analytics performed upon the uncompressed data. However, it runs four times faster, accesses only 14% as much data and requires significantly less storage since the data is always compressed. These results show that it is possible to simultaneously reap the benefits of compression and accurate, high-speed analytics in many applications.**

***Index Terms*—data mining, data compression, Internet of Things, clustering methods, explainable AI**

I. Introduction

While enabling many transformative applications, the amount of data generated by the Internet of Things (IoT) also poses significant challenges for data transmission, storage and analytics. Data compression naturally fulfils an important role here by reducing the cost of data transmission and storage [1], [2]. However, compression introduces a barrier to analytics, in that data must be decompressed before it can be analysed.

Compression algorithms capable of rapid decompression [2], [3], high-speed, stand-alone decompression techniques [4]–[6], and optimising the choice of compression scheme based on the desired analytics task [7] have all been proposed as solutions to this obstacle. However, all these approaches still fundamentally rely on decompression.

A few works have investigated performing analytics *directly* on compressed data, obviating the need for decompression. TRISTAN [8] and Plato [9], for example, use lossy compression schemes to represent data in a convenient form for approximate analytics. Plato offers the added benefit of tight error guarantees. These approaches provide significantly faster and reasonably accurate analytics compared to analytics performed on uncompressed data. However, they are limited to time series data and are unsuitable for applications such as anomaly detection due to their lossy compression. Zwift [10] and ALACRITY [11], on the other hand, use lossless compression and provide exact analytics for textual and floating-point data, respectively. However, this comes at the expense of less compression, and slow compression in the case of Zwift.

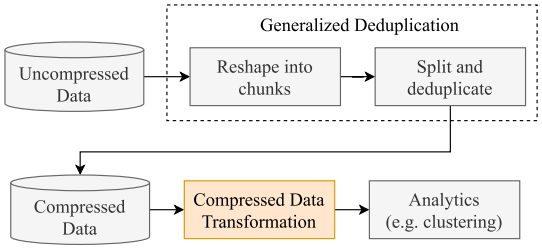

Fig. 1. System diagram for compressed data analytics.

In general, the benefits of direct analytics of compressed data are twofold. First, data can be permanently stored in compressed form. Second, it is often possible to achieve a speed-up relative to analytics on uncompressed data by leveraging the structure of compressed data [7]–[10].

In this paper, we explore direct compressed data analytics. The proposed solution promises to reduce Internet traffic load and cloud storage, accelerate analytics and improve overall energy efficiency in the IoT ecosystem. Our approach, as illustrated in Fig. 1, is based on the compression algorithm Generalized Deduplication (GD). This algorithm was chosen as it is lossless, well suited to IoT data and offers efficient random access capabilities [12]–[14]. Similar to previous works [8]–[11], we exploit the structure of compressed data for efficient analytics. However, our approach is unique in that it is applicable to any kind of single or multivariate numeric data and is based on a lossless compression scheme. In this paper, we specifically focus on clustering and test our methods with k-means and two explainable clustering algorithms. Experiments on synthetic and real-world IoT data show that our compressed data analytics matches the quality of clustering performed on uncompressed data to within 1–5% difference. Meanwhile, it accesses only 14% as many bits and runs more than four times faster. Our main contributions are:

1) Efficient compressed data analytics methods for numeric data based on a lossless compression algorithm,
2) Experimental validation of these methods in the context of clustering, including explainable clustering, and
3) Insights into key challenges of direct compressed data analytics and considerations for managing compression and analytics holistically.

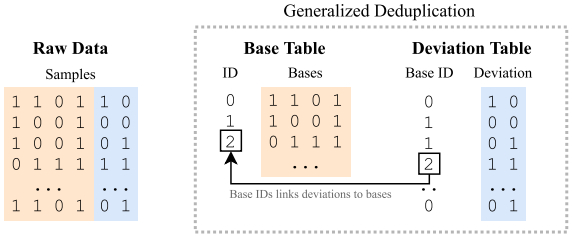

Fig. 2. GD splits data at the bit level into bases (orange), which are deduplicated, and deviations (blue), which are stored verbatim and linked, via the base IDs, to their corresponding base.

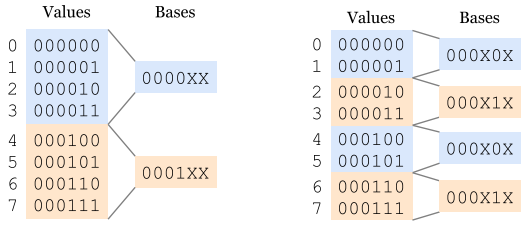

Fig. 3. Generalized Deduplication maps or 'bins' data values to bases, which may progress linearly (a) or be interleaved (b).

The remainder of this paper includes an overview of relevant background material in Section II, followed by the details of our compressed data analytics methods in Section III. Section IV provides an evaluation of their performance and the paper is concluded in Section V.

II. Background

This section provides an overview of GD and the clustering algorithms that were used for testing.

A. Generalized Deduplication

GD is a data compression algorithm which operates on data *chunks*, which consist of one or more concatenated data samples. The core idea is to consider chunks at the bit level and split them into a part that can be deduplicated, and a remainder that must be stored verbatim [12]. These parts are called the *base* and *deviation*, respectively. To ensure high quality splits, GD orders bits based on their correlations. For example, in many data the most significant bits have minimal variation (more duplicates) and high correlation among each other, making them good candidates for the base. Meanwhile, less significant bits with more variation are assigned to the deviation. Fig. 2 illustrates an example of GD applied to six bit chunks with the first four bits assigned to the base.

By mapping each data object to a base, GD performs a kind of data binning, as illustrated in Fig. 3(a). In this example, all chunks beginning with the same four bits are mapped or 'binned' to the same base. The points `000100`, `000101`, `000110` and `000111` are therefore all binned to the base `0001XX`, where `X`s indicate the location of the deviation bits. We exploit this binning effect for compressed data analytics. For a more detailed explanation of GD, see [12], [13].

B. Clustering & Explainability

Our compressed data analytics methods were tested using k-means and two explainable variations of k-means. Explainability has recently emerged as critical to analytics, especially for artificial intelligence and machine learning [15], [16]. Clustering is an important and widely used data analytics task that is highly relevant to IoT use cases. For example, finding patterns in traffic data [17], understanding electricity consumption patterns [18] and optimising logistics systems [19]. Unfortunately, both widely-used and state-of-the-art algorithms alike tend to provide poor explainability [20], [21]. Recently, a post-hoc explainable clustering algorithm known as Iterative Mistake Minimisation (IMM) has been proposed [21]. IMM approximates a given representative-based *reference clustering* by training a threshold tree that minimises the number of data points that are separated from their correct cluster centres, relative to the reference clustering. For k clusters and tree height H, the authors prove an approximation guarantee of $8Hk + 2$ relative to k-means based on the sum of squared errors (SSE). They also show that using a standard decision tree algorithm in place of their threshold tree results in arbitrarily high SSE. We refer to this algorithm as Decision Tree Clustering (DTC). Note that neither IMM nor DTC have previously been empirical evaluated.

III. Compressed Data Analytics

In this section we describe the proposed methods for direct analytics of compressed data.

A. Key Concepts & Nomenclature

Two key concepts for our methods are the *base centroid* and the *maximum deviation*. These represent a central value for a given base and the size of the region that maps to it, respectively. The base centroid is the mean of the minimum and maximum values that can be mapped to the given base. Meanwhile, the maximum deviation is their difference. For example, consider again the six bit dataset from Fig. 2(a). The second base, `0001XX`, has maximum and minimum values of `000111` and `000100`. These correspond to all ones and zeros, respectively, in the deviation bits. The base centroid is therefore $\texttt{000101.1}_b$ or 5.5_{10} while the maximum deviation is $\texttt{000011}_b$ or 3_{10}. Note that these are both vectors in the case of multi-variate data. Also, if chunks contain more than one sample, each base will have multiple base centroids.

The nomenclature used in the subsequent sections is summarised in Table I. A *sample* refers to an uncompressed d-dimensional data point. The number of samples n and the sample (bit) length l_s define the size of the uncompressed data. Prior to compression, GD determines the optimal number of samples, c, to concatenate to form *chunks*. The chunk length l_c is therefore cl_s, while $n_c = \frac{n}{c}$. During compression, GD splits chunks into a base and deviation based on the mapping $\mathcal{I}$, hence $l_c = l_b + l_d$. The number of unique bases after deduplication, n_b, determines the base ID length, l_{id}, such

TABLE I
NOMENCLATURE

Notation	Meaning	Notation	Meaning
l_s	bits per sample	n	number of samples
l_c	bits per chunk	n_c	number of chunks
l_b	bits per base	n_b	number of unique bases
l_d	bits per deviation	d	dimensionality
l_{id}	bits per base ID	$\mathcal{I}$	deviation bit indices
c	samples per chunk	η	compression ratio

that $l_{id} = \lceil \log n_b \rceil$. For good compression, it is generally true that $n_b \ll n_c$. Finally, the *compression ratio*, η, is defined as

$$\eta \triangleq \frac{compressed\ size}{uncompressed\ size} = \frac{n_b l_b + n_c(l_{id} + l_d)}{n l_s}. \quad (1)$$

As such, smaller values correspond to better compression and a value of 1 indicates no compression.

B. Compressed Data Transformation

To enable direct analytics of compressed data, we propose the following methods for transforming compressed data into suitable representations for analytics.

1) **Centroids-only:** Represent the data using base centroids.
2) **Maximum deviation weighted centroids:** Represent the data using base centroids and weights derived from the maximum deviations. Three weighting methods are proposed: (i) maximum of the maximum deviations, (ii) mean of the maximum deviations and (iii) product of the maximum deviations, which corresponds to the volume of space mapped to a base. Note that these weighting methods are equivalent for univariate data.
3) **Frequency-weighted centroids:** Represent the data using base centroids weighted by their frequency.

These methods are described collectively in Algorithm III-B. The algorithm takes the GD compressed data as input, namely the parameters file $\mathcal{P}$, base table $\mathcal{B}$ and deviations table $\mathcal{D}$. First the compression parameters are extracted, followed by calculating the base centroids and maximum deviations in lines 4–7. The minimum and maximum base values, B_{min} and B_{max}, are computed from the bases, $\mathcal{B}$, by filling `0`s or `1`s, respectively, in the deviation bits according to $\mathcal{I}$ and then converting to floating point numbers (lines 4 and 5). Note that for multi-dimensional data ($d > 1$), B_{min} and B_{max} will contain vectors. The base centroid frequencies are computed in lines 8–12. First, l_{id} is decoded from $\mathcal{D}$ using Elias Gamma coding [13]. The base IDs are then loaded from $\mathcal{D}$ and converted to integers. If $c > 1$, the base IDs are extended c times, since each base corresponds to c base centroids for a total of cn_b base centroids. Finally, the centroid frequencies are computed. Analytics algorithms, such as k-means, can then be applied to the output of these methods.

Algorithm 1 Compressed data transformation

Inputs: GD parameters $\mathcal{P}$, base table $\mathcal{B}$, deviations table $\mathcal{D}$
Outputs: base centroids, max deviations, centroid frequencies
1: **procedure** TRANSFORMCOMPRESSED($\mathcal{P}, \mathcal{B}, \mathcal{D}$)
2: $l_s, c, d, \mathcal{I} \leftarrow$ load compression parameters from $\mathcal{P}$
3: $l_c \leftarrow cl_s$
4: $B_{min} \leftarrow float\left(\kappa\left(\mathbf{0}_{n_b \times l_c}, \mathcal{B}, \mathcal{I}\right)\right)$
5: $B_{max} \leftarrow float\left(\kappa\left(\mathbf{1}_{n_b \times l_c}, \mathcal{B}, \mathcal{I}\right)\right)$
6: $B_{centroids} \leftarrow mean\left(B_{max}, B_{min}\right)$
7: $B_{max.dev.} \leftarrow B_{max} - B_{min}$
8: $l_{id} \leftarrow$ decode l_{id} from $\mathcal{D}$
9: $J \leftarrow$ load base IDs from $\mathcal{D}$
10: $J \leftarrow int\left(J\right)$ ▷ Convert to integer
11: $J' \leftarrow$ extend base IDs list if $c > 1$
12: $j_{freq.} \leftarrow hist\left(\mathcal{J}'\right)$ ▷ Base centroid frequencies
13: **return** $B_{centroids}, B_{max.dev.}, j_{freq.}$
14: **end procedure**

C. Complexity

This section provides an analysis of the number of bits accessed and the time complexity for our compressed data analytics methods. Table II summarises the results, including the time complexity of k-means clustering.

The centroids-only and maximum deviation weighted methods only require accessing the base table, which is $n_b l_b$ bits. The frequency-weighted method accesses an additional $n_c l_{id}$ bits from the deviations table to load the base IDs. This can be reduced, with a small increase in compressed data size, by storing the base counts as metadata.

The centroids-only method requires (1) determining the minimum and maximum values for each base centroid, (2) converting these from binary to decimal and (3) calculating the mean. These steps require $2cn_b d$, $2cn_b dl_s$ and $2cn_b d$ operations, respectively, giving a total time complexity of $cn_b d(4 + 2l_s)$. The maximum deviation weighted method requires also calculating the maximum deviations and computing weights ($2cn_b d$ operations) for a total total time complexity of $cn_b d(6 + 2l_s)$. The frequency-weighted method requires instead computing the base centroid frequencies by (1) converting the n_c base IDs from binary to integer, (2) extending them c times and (3) computing their frequencies. These steps require $n_c l_{id}$, 0 and cn_c operations, respectively, for a total time complexity of $cn_b d(4 + 2l_s) + n_c(l_{id} + c)$. Adding the time to run k-means, which has time complexity $O(Nkdt)$ for N data points and t iterations, gives the results in Table II.

Importantly, clustering on compressed data scales with n_b rather than n, which is typically much smaller. This can also be expressed in terms of n and the compression ratio, η, by rearranging the definition of η to obtain

$$n_b = n\left(\alpha\eta - \beta\right), \quad (2)$$

where $\alpha = l_s/l_b$ and $\beta = \left(l_{id} + l_d\right)/cl_b$, which are both constants given a compression configuration. Hence, better compression (smaller η) corresponds to better efficiency.

D. Challenges

By splitting data into bases and deviations, GD introduces two challenges related to the distribution of base centroids.

TABLE II
NUMBER OF BITS ACCESSED AND TIME COMPLEXITY INCLUDING K-MEANS.

Approach	Bits accessed	Time complexity
Centroids-only	$n_b l_b$	$O(cn_b d(l_s + kt))$
Maximum deviation	$n_b l_b$	$O(cn_b d(l_s + kt))$
Frequency-weighted*	$n_b l_b + n_c l_{id}$	$O(cn_b d(l_s + kt) + n_c l_{id})$
Uncompressed	nl_s	$O(ndkt)$

* Can be reduced by storing base counts as metadata.

TABLE III
DATASETS USED FOR PERFORMANCE EVALUATION.

Dataset	Datasets	n	d	Size	η
Synthetic	10	100,000	2	782 KB	0.819
Synthetic + noise*	30	$\geq$101,000	2	823 KB	0.836
Power [23], [24]	1	400,000	7	10.9 MB	0.536

* n = 101,000, 105,000 or 110,000 depending on the noise rate.

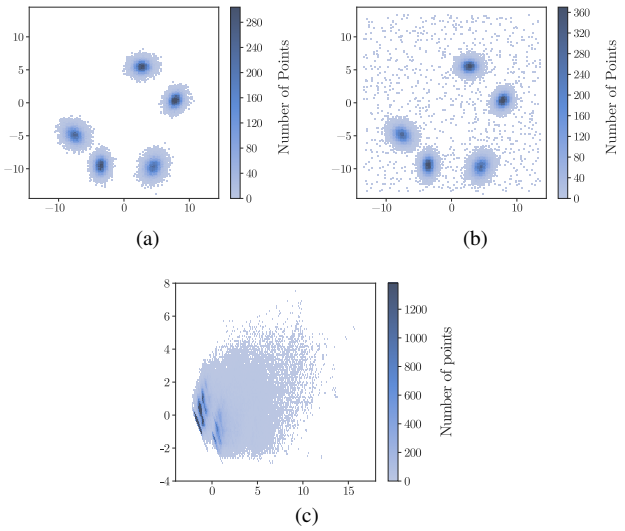

Fig. 4. Datasets used for evaluating compressed data clustering: (a) one of the synthetic datasets with well-defined Gaussian clusters, (b) the same dataset with an additional 1% noise points, and (c) power consumption data (originally seven dimensional, the plot shows the two Principal Components).

The first is that the mapping between data values and bases may not follow a linear progression. This occurs if deviation and base bits are interspersed. For example, consider a six bit dataset with bits 0, 1, 2, and 4 assigned to the base. As shown in Fig. 3(b), the values `000001` and `000100` map to base `000X0X`, whereas the intermediate values `000010` and `000011` map to a different base. In this scenario, base centroids are *outside* the domain of values mapped to them.

The second is that, when compressing IEEE 754 standard floating point numbers [22], the spacing between base centroids is multiplicative rather than additive. This occurs if the base contains any exponent bits. For example, take the case of 32-bit floating point numbers and allocate the sign and all exponent bits to the base and all mantissa bits to the deviation. Consider the bases `001111111`, `010000000`, `010000001` and `010000010`. The exponent part of the first base is `01111111` or 127. Hence, its minimum and maximum values are $1 \times 2^{127-127}$ and $2 \times 2^{127-127}$ and its centroid is $1.5 \times 2^{127-127} = \mathbf{1.5}$. Similarly, the other bases' centroids are $1.5 \times 2^{128-127} = \mathbf{3}$, $1.5 \times 2^{129-127} = \mathbf{6}$ and $1.5 \times 2^{130-127} = \mathbf{12}$, which are spaced in multiples of two.

IV. PERFORMANCE EVALUATION

This section presents the performance evaluation of clustering directly on compressed data in comparison to clustering performed on uncompressed data. Both synthetic and real-world IoT datasets are examined. Specifically, we used two-dimensional synthetic data with well-separated Gaussian clusters, with and without noise, and real household power consumption data with seven attributes [23]. These are summarised in Table III and selected datasets are visualised in Fig. 4. Dataset size reflects the uncompressed binary data.

All datasets were compressed using GD with the following two modifications to enhance data analytics performance.

1) Storing d in the parameters file to avoid building in assumptions to determine the number of attributes.
2) Restricting the sign and exponent bits to the base for floating point data.

Without the second modification, clustering results were unacceptably inaccurate. This was due to the minimum and maximum values for base centroids being orders of magnitude apart if even one exponent bit is assigned to the deviation. In the case of the power consumption dataset, this modification produced highly accurate clustering on the compressed data, at the cost of slightly worsening η by 2.3% from 0.524 to 0.536. This modification was unnecessary for the synthetic data.

For clustering, we used k-means and two explainable algorithms, IMM and DTC. This allowed us to evaluate how clustering directly on compressed data affects clustering quality and explainability independently. Clustering performance was measured using sum of squared errors (SSE) and compared using the *approximation ratio* (AR), calculated by normalising relative to the SSE k-means on uncompressed data. Hence, an AR of 1 corresponds to no loss in clustering quality relative to the uncompressed case and higher values indicate increasingly poor quality. Note that when clustering directly on compressed data, the result is a clustering of the base centroids, not the original points. We therefore used the uncompressed data points together with the cluster centres derived from compressed data clustering to calculate the SSE for compressed data clustering.

A. Synthetic Data

The synthetic data consisted of ten two-dimensional synthetic datasets each with five well-separated Gaussian clusters. For each cluster, attribute variances were drawn from a normal distribution $N \sim (0.5, 0.15^2)$, while covariances were drawn

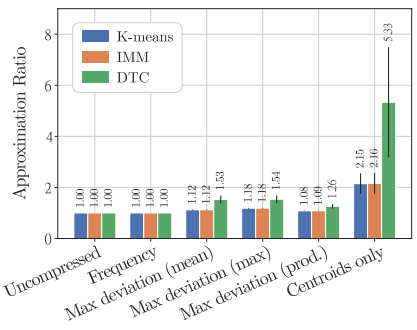

Fig. 5. Clustering results from synthetic data in terms of approximation ratio.

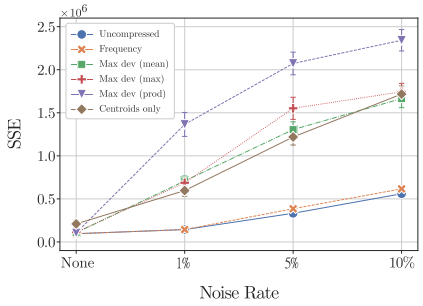

Fig. 6. Clustering results from noisy synthetic dataset in terms of SSE (k-means only).

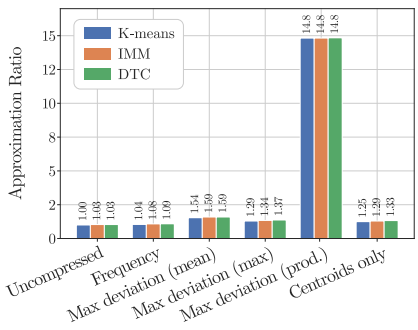

Fig. 7. Clustering results from the power consumption dataset with $k = 4$. Error bars omitted due to trivially small 98% confidence intervals.

from an exponential distribution ($\beta = 0.1$) with random sign. Cluster centres were drawn uniformly at random in the interval $[-10, 10]$, subject to being a distance of at least six apart to prevent overlap.

K-means, IMM and DTC were each run ten times for all four compressed data analytics approaches and uncompressed data on all ten datasets. This included the three variants of the maximum deviation weighted approach for a total of $3 \times 10 \times 6 \times 10 = 1800$ clustering runs. Aggregate results are shown in Fig. 5 where ARs have been calculated relative to k-means on uncompressed data, averaged over the ten runs. Error bars represent 98% confidence intervals. It can be seen that both the frequency and maximum deviation weighted approaches closely approximate the uncompressed data results. In particular, the frequency-weighted approach almost perfectly matches with an AR of 1.00 for all clustering algorithms. Notably, IMM performs far better than its theoretical worst case [21], nearly matching the performance of k-means in all cases while consistently outperforming DTC.

B. Synthetic Data With Noise

This section details our analysis of clustering on synthetic data with additional noise. Three versions of the synthetic datasets were generated by adding 1%, 5%, and 10% uniformly distributed noise points. An example with 1% noise points is shown in Fig. 4(b). We ran k-means with each of the compressed data analytics approaches and uncompressed data ten times for the ten datasets and three noise rates for a total of $6 \times 10 \times 10 \times 3 = 1800$ clustering runs. The SSE results, including data with no noise, are visualised in Fig. 6.

It can be seen that, as expected, the SSE increases with the amount of noise for all approaches. However, the frequency-weighted method scales far better than the centroids-only and maximum deviation weighted methods. For the maximum deviation weighted methods, this can be explained by observing that the weights are independent of the number of points mapped to a base. A base linked to a single noise point is therefore treated the same as a base at the centre of a cluster.

C. Power Consumption Data

This section details our analysis with household power consumption data, which contains seven attributes and was obtained from the UCI Machine Learning Repository [24]. K-means, IMM and DTC were again run using each compressed data analytics method and uncompressed data. The number of clusters was set to 4 based on the Elbow method [25]. Results are shown in Fig. 7.

As with synthetic data, the frequency-weighted method closely matches the uncompressed data results and outperforms the other compressed data analytics methods. That said, the approximation ratios are worse compared to the synthetic data experiments for all methods. This is primarily due to the data having less well-defined clusters and noise (see Fig. 4). However, the especially poor performance of the product weighted method is due to the higher dimensionality of the data, since this creates more potential for a single attribute's maximum deviation to incorrectly skew the weights. To show this, k-means clustering was run on subsets of the dataset's attributes (2–7 attributes), the results of which are shown in Fig. 8. It can be seen that the performance of the product weighted maximum deviation method worsens substantially as dimensionality increases, while the other approaches perform consistently. Note that the number of clusters used for k-means was selected independently for each dimensionality using the Elbow method [25] and set to 6, 4, 5, 5, 5 and 4 (low to high dimensionality). Finally, as with synthetic data, the marginal increase in SSE of using the explainable clustering algorithms is very low relative to k-means.

D. Running Cost

This section details our analysis of the running costs of compressed and uncompressed data clustering. This was evaluated by counting the number of bits accessed and operations performed, which is used as a proxy for runtime and includes both compressed data transformation (if applicable) and clustering. The results are shown in Table IV. Calculations are based on k-means and take the number of operations for clustering to be $Nkdt$. We used $t = 300$ as this is the scikit-learn default.

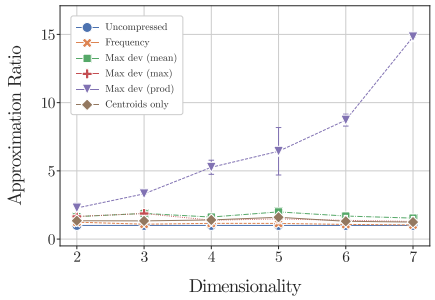

Fig. 8. Clustering results on subsets of the power consumption dataset.

TABLE IV
COMPARISON OF RUNNING COST.

	Bits Accessed ($\times 10^3$)		**Operations** ($\times 10^6$)	
Approach	**Synthetic**	**Electricity**	**Synthetic**	**Electricity**
Centroids	136	9,441	17.5	806
Max. Deviation	136	9,441	17.5	807
Freq. Weighted	1,376	12,641	18.8	809
Uncompressed	6,400	89,600	300.0	3,360

Compared to analytics on uncompressed data, our frequency-weighted method required accessing only 22% and 14% as many bits for the synthetic and power consumption datasets, respectively. Likewise, it performed only 6.3% and 24.1% as many operations. It can thus be considered 15.9 and 4.1 times faster, respectively for the two datasets, than performing the same analytics on the uncompressed data.

V. CONCLUSION

A new, efficient approach for direct analytics of compressed data has been proposed, based on the lossless compression algorithm Generalized Deduplication. When applied to clustering, our base frequency-weighted method approximated uncompressed data analytics with only a 1–5% difference, while accessing only 14% as much data and running in less than a quarter of the time. Significant storage savings were also realised with a compression ratio of 53.6%. Two important modifications to Generalized Deduplication that enable effective compressed data analytics were also implemented. For future work, we intend to stress-test these results on greater variety of datasets. We also intend to investigate more analytics tasks such as outlier detection, classification and basic queries (e.g. mean, maximum and variance).

VI. ACKNOWLEDGEMENTS

This work is supported by Analytics Straight on Compressed IoT Data (Light-IoT) project (Grant No. 0136-00376B) granted by the Danish Council for Independent Research and Aarhus University's DIGIT Centre.

REFERENCES

[1] G. Chiarot and C. Silvestri, "Time series compression: a survey," 2021, arXiv:2101.08784v1.

[2] D. Blalock, S. Madden, and J. Guttag, "Sprintz: Time series compression for the internet of things," *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1–23, 2018.

[3] S. Funasaka, K. Nakano, and Y. Ito, "Adaptive loss-less data compression method optimized for GPU decompression," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 24, p. e4283, 2017.

[4] E. Sitaridi, R. Mueller, T. Kaldewey, G. Lohman, and K. A. Ross, "Massively-parallel lossless data decompression," in *45th International Conference on Parallel Processing (ICPP)*, 2016.

[5] M. Plauth and A. Polze, "GPU-based decompression for the 842 algorithm," in *Seventh International Symposium on Computing and Networking Workshops (CANDARW)*, 2019.

[6] J. Fang, J. Chen, J. Lee, Z. Al-Ars, and H. P. Hofstee, "An efficient high-throughput LZ77-based decompressor in reconfigurable logic," *Journal of Signal Processing Systems*, vol. 92, no. 9, pp. 931–947, 2020.

[7] J. Paparrizos *et al.*, "VergeDB: A database for IoT analytics on edge devices," in *11th Annual Conference on Innovative Data Systems Research*, 2021.

[8] A. Marascu *et al.*, "TRISTAN: Real-time analytics on massive time series using sparse dictionary compression," in *IEEE International Conference on Big Data*, 2014.

[9] C. Lin, E. Boursier, and Y. Papakonstantinou, "Plato: Approximate analytics over compressed time series with tight deterministic error guarantees," *Proceedings of the VLDB Endowment*, vol. 13, no. 7, pp. 1105–1118, 2020.

[10] F. Zhang, J. Zhai, X. Shen, O. Mutlu, and W. Chen, "Zwift: A programming framework for high performance text analytics on compressed data," in *2018 International Conference on Supercomputing*, 2018.

[11] J. Jenkins *et al.*, "ALACRITY: Analytics-driven lossless data compression for rapid in-situ indexing, storing, and querying," in *Transactions on Large-Scale Data- and Knowledge-Centered Systems X*. Springer Berlin Heidelberg, 2013, pp. 95–114.

[12] R. Vestergaard, Q. Zhang, and D. E. Lucani, "Lossless compression of time series data with generalized deduplication," in *IEEE Global Communications Conference (GLOBECOM)*, 2019.

[13] R. Vestergaard, D. E. Lucani, and Q. Zhang, "A randomly accessible lossless compression scheme for time-series data," in *IEEE INFOCOM*, 2020.

[14] R. Vestergaard, Q. Zhang, M. Sipos, and D. E. Lucani, "Titchy: Online time-series compression with random access for the internet of things," *IEEE Internet of Things Journal*, 2021.

[15] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[16] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (XAI): Toward medical XAI," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020.

[17] D. Puschmann, P. Barnaghi, and R. Tafazolli, "Adaptive clustering for dynamic IoT data streams," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 64–74, 2017.

[18] J. Xiong *et al.*, "Enhancing privacy and availability for data clustering in intelligent electrical service of IoT," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1530–1540, 2019.

[19] J. Yang, Y. Han, Y. Wang, B. Jiang, Z. Lv, and H. Song, "Optimization of real-time traffic network assignment based on IoT data using DBN and clustering model in smart city," *Future Generation Computer Systems*, vol. 108, pp. 976–986, 2020.

[20] I. T. Christou, N. Kefalakis, A. Zalonis, and J. Soldatos, "Predictive and explainable machine learning for industrial internet of things applications," in *16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2020.

[21] M. Moshkovitz, S. Dasgupta, C. Rashtchian, and N. Frost, "Explainable k-means and k-medians clustering," in *37th International Conference on Machine Learning*, vol. 119, 2020, pp. 7055–7065.

[22] "IEEE standard for floating-point arithmetic," *IEEE Standard 754-2019 (Revision of IEEE 754-2008)*, 2019.

[23] G. Hebrail and A. Berard, "Individual household electric power consumption data set," 2012. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

[24] D. Dua and C. Graff, "UCI machine learning repository," 2019. [Online]. Available: http://archive.ics.uci.edu/ml

[25] R. L. Thorndike, "Who belongs in the family?" *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953.