electronics

Article

Dynamic Edge-Based High-Dimensional Data Aggregation with Differential Privacy

Qian Chen [1,2,3,*], Zhiwei Ni [1,2], Xuhui Zhu [1,2], Moli Lyu [1,2], Wentao Liu [1,2] and Pingfan Xia [3,4]

[1] School of Management, Hefei University of Technology, Hefei 230009, China; nzwgd@hfut.edu.cn (Z.N.); zhuxuhui@hfut.edu.cn (X.Z.); lmorry@mail.hfut.edu.cn (M.L.); wtliu@mail.hfut.edu.cn (W.L.)
[2] Key Laboratory of Process Optimization and Intelligent Decision-Making, Ministry of Education, Hefei 230009, China
[3] Intelligent Interconnected Systems Laboratory of Anhui Province, Hefei University of Technology, Hefei 230009, China; xiapingfan@mail.hfut.edu.cn
[4] School of Big Data and Statistics, Anhui University, Hefei 230601, China
* Correspondence: chenqian123@mail.hfut.edu.cn

Abstract: Edge computing enables efficient data aggregation for services like data sharing and analysis in distributed IoT applications. However, uploading dynamic high-dimensional data to an edge server for efficient aggregation is challenging. Additionally, there is the significant risk of privacy leakage associated with direct such data uploading. Therefore, we propose an edge-based differential privacy data aggregation method leveraging progressive UMAP with a dynamic time window based on LSTM (EDP-PUDL). Firstly, a model of the dynamic time window based on a long short-term memory (LSTM) network was developed to divide dynamic data. Then, progressive uniform manifold approximation and projection (UMAP) with differential privacy was performed to reduce the dimension of the window data while preserving privacy. The privacy budget was determined by the data volume and the attribute's Shapley value, adding DP noise. Finally, the privacy analysis and experimental comparisons demonstrated that EDP-PUDL ensures user privacy while achieving superior aggregation efficiency and availability compared to other algorithms used for dynamic high-dimensional data aggregation.

Keywords: data aggregation; differential privacy; edge computing; progressive UMAP; Shapley value

Citation: Chen, Q.; Ni, Z.; Zhu, X.; Lyu, M.; Liu, W.; Xia, P. Dynamic Edge-Based High-Dimensional Data Aggregation with Differential Privacy. *Electronics* **2024**, *13*, 3346. https://doi.org/10.3390/electronics13163346

Academic Editor: Javid Taheri

Received: 14 July 2024
Revised: 9 August 2024
Accepted: 20 August 2024
Published: 22 August 2024

1. Introduction

The Internet of Things (IoT), as a distributed and internet-based system, facilitates the transformation of everyday objects into intelligent entities [1]. IoT applications necessitate the aggregation of data from various sources, such as smart wearable devices, vehicle networking data, and drone data, which facilitates the subsequent analysis, prediction, and decision-making processes [2]. Edge computing allows for the storage and processing of data on edge servers located closer to end users [3]. Each edge server at the network's periphery collects and aggregates user data, effectively addressing the inefficiency associated with the direct transmission of user data over long distances to cloud servers [4,5]. IoT data aggregation based on edge computing is shown in Figure 1. Each user equipment (UE) can upload data to a nearby edge server, which can then be aggregated in the cloud. However, this method introduces new challenges related to privacy and security. From the user's perspective, an edge server cannot be fully trusted due to potential malicious nodes that exploit simulated base station signals to collect user data [6]. Additionally, the security mechanisms implemented on edge nodes may not provide sufficient preservation against theft or attacks targeting the data stored on these servers [7]. Consequently, privacy preservation becomes particularly critical in edge-based data aggregation.

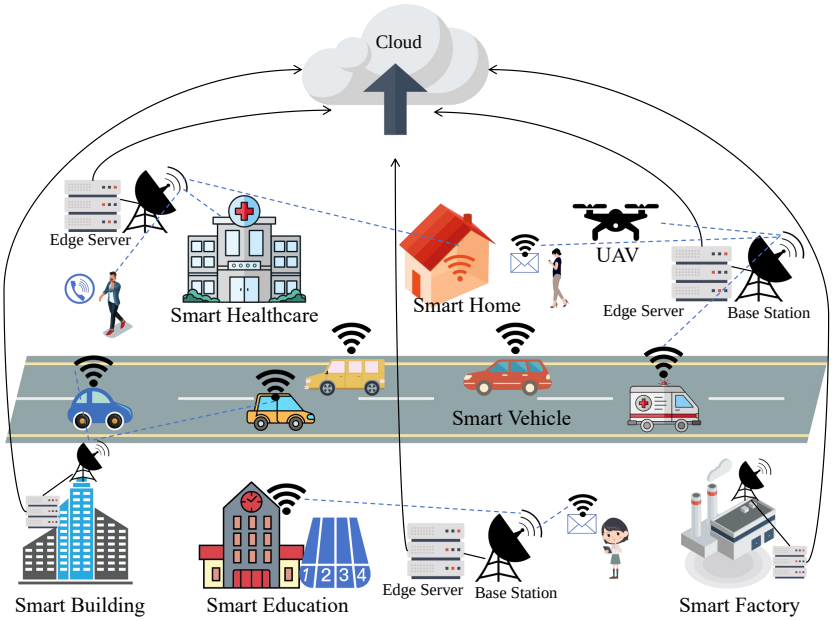

Figure 1. IoT data aggregation based on edge computing.

The implementation of differential privacy (DP) mechanisms [8], which introduce perturbation through noise to user data prior to uploading it to an edge server, can effectively prevent privacy breaches [9,10]. In distributed applications, data often exhibit dynamic and high-dimensional characteristics and contain a large amount of privacy-sensitive information [11], such as real-time healthcare information with multiple attributes like heart rate, blood pressure, blood oxygen saturation, etc. High-dimensional data contain massive amounts of private user data, which may be leaked during the direct upload of data, and entail a high computation cost for privacy processing [12,13]. Moreover, the dynamics of such data pose challenges in allocating privacy budgets in DP to ensure both their security and availability [14]. Therefore, utilizing DP for dynamic high-dimensional data prior to uploading is challenging, as it becomes imperative to enhance aggregation efficiency and ensure data availability while protecting user privacy.

Dimensionality reduction can significantly enhance the efficiency of sharing and analyzing high-dimensional data [15]. The noise of the DP mechanism perturbs the data after dimensionality reduction, preventing attackers from inferring private information in the transition from low-dimensional to high-dimensional data [16]. Thus, employing dimensionality reduction techniques based on DP is an effective approach for securely uploading high-dimensional data to an edge server while preserving privacy. Many methods for high-dimensional data processing using DP are based on dimensionality reduction, such as principal component analysis [17–19], random projection [16,20], etc. However, these are linear methods that only extract the linear features of input data and do not capture any nonlinear features. Therefore, the linear methods do not provide satisfactory results in many real scenarios [21]. Uniform manifold approximation and projection (UMAP) [22], a popular manifold learning method, effectively extracts nonlinear features from high-dimensional datasets while preserving both global and local characteristics. In comparison to existing nonlinear approaches for the dimensionality reduction of large-scale, high-dimensional data, UMAP not only exhibits superior performance but also has low latency. Based on UMAP, progressive UMAP [23] enables the embedding of out-of-sample data and efficiently handles dynamic (streaming) data, which can enhance the aggregation efficiency of high-dimensional dynamic datasets.

As mentioned above, progressive UMAP with DP is a robust methodology employed for the aggregation of dynamic high-dimensional data while ensuring privacy. However, the dynamic changes in high-dimensional data in edge computing present a significant challenge for progressive UMAP with DP. Distinct edge servers aggregate data with varying

volumes from user equipment (UE) in each time slot [24]. The window mechanism is widely used to divide the stream in order to aggregate time-series data [25]. A fixed window size results in a small amount of data in certain time windows, which are prone to privacy leakage by attacker inference [26]. Long short-term memory (LSTM) is a neural network widely employed for the prediction of data streams, effectively capturing the dynamics of data. Enabling adaptive adjustment of the window size through LSTM can effectively mitigate the risk of privacy leakage [26]. However, the UEs in edge computing exhibit diverse data sizes and dynamics. Therefore, the challenge lies in effectively dividing time windows using LSTM in edge computing and allocating privacy budgets when employing progressive UMAP with DP, in order to minimize IoT data availability loss while safeguarding user privacy.

To address the above challenges, we proposed a novel data aggregation method for dynamic high-dimensional data on edge computing, referred to as EDP-PUDL (Edge-based Differential Privacy data aggregation leveraging Progressive UMAP with Dynamic time window based on LSTM). This method utilizes the Long Short-Term Memory (LSTM) to predict data dynamics, enabling adaptive adjustment of the time window size for each UE's data uploading. Then it performs progressive UMAP with DP to reduce data dimension while ensuring data privacy preservation. Additionally, the privacy budget allocation is optimized by taking into account variations in data volume from UE and the various utilities of attribute data, aiming to minimize the negative impact on data availability. EDP-PUDL ensures privacy preservation for user data aggregation, effectively addresses the security aggregation problem of dynamic high-dimensional data in distributed applications, and guarantees data availability while maintaining efficiency.

The main contributions of this paper are as follows:

1. A novel method for dynamic high-dimensional data aggregation with edge computing was developed. It performs DP-based progressive UMAP with a dynamic time window for the dimensionality reduction and privacy preservation of user data prior to their upload. The data streams are divided into multiple windows of varying sizes based on the LSTM-predicted data volume. It improves the efficiency of dynamic high-dimensional data aggregation while guaranteeing data privacy and availability.
2. The privacy budget of DP noise for UE is allocated based on the data volume of the UE in time windows. Then, the Shapley value of each attribute is calculated after dimensionality reduction, and the privacy budget is allocated by the Shapley value to perturb each attribute's data using DP noise, which can further reduce the loss of available data.
3. Comparative experiments were conducted on multiple public high-dimensional datasets in real time to verify the utility of the proposed EDP-PUDL method. The experimental results demonstrate that EDP-PUDL outperforms the mentioned algorithms used for comparison on evaluation metrics including the total average variation distance, total L_2 error, misclassification rate, and runtime.

The rest of this paper is organized as follows: We review related work in Section 2 and provide preliminaries and problem definitions in Section 3. Section 4 introduces the details of EDP-PUDL and provides an analysis of data privacy. Section 5 provides our experimental results and an empirical analysis of the proposed algorithm. Section 6 concludes this paper.

2. Related Work

Research on privacy preservation during data aggregation aims to ensure the confidentiality of multiple data owners. Numerous methods have been proposed to ensure privacy during the aggregation of data based on one aggregator. For instance, Yan et al. [27] employed encrypted sharing techniques to safeguard the privacy of sensor data and aggregation outcomes. However, their method is only applicable to single-dimensional data aggregation. Yankson et al. [28] proposed a novel secure data aggregation algorithm in the Industrial Internet of Things (IIoT) that utilizes GNN with federated learning to enhance privacy protection for data aggregation. However, the algorithm relies on trust

assumptions among participating parties, and any compromise in trust could undermine the security and privacy guarantees it offers. Liu et al. [29] proposed a payment protection level (PPL) game model based on DP, wherein each participant submits sensing data to the platform for aggregation. Nash equilibrium points are derived in the game through deep reinforcement learning techniques to enhance utility for both platforms and participants during data aggregation. However, these methods only employ a single aggregator for data aggregation. When there are a large number of UEs and a substantial volume of data, a single aggregator may encounter insufficient bandwidth resources, resulting in significant time delays during the process of data aggregation.

Several DP-based data aggregation methods based on multiple aggregators have been employed. For instance, Tian et al. [30] introduced a DP-based data aggregation scheme that utilizes certification-based aggregated short signatures to ensure data authentication and integrity. In particular, Laplacian noise is added to aggregated data to mitigate differential attacks. However, the method lacks a data utility guarantee. Tang et al. [31] introduced a secure health data aggregation scheme that incorporates DP by securely collecting health data from diverse sources, employing signature technology to ensure equitable incentives for patients and introducing noise into health data to enhance privacy preservation. However, this method also does not provide a guarantee of data utility. Wang et al. [32] proposed an edge-based differential privacy data collection model, where raw data from wireless sensor networks (WSNs) undergo processing by DP algorithms on edge servers, aiming to reduce communication and storage costs. However, the model has a large computational overhead when processing high-dimensional data considering privacy. Shang et al. [33] proposed a robust privacy-preserving data aggregation scheme for the edge-supported IIoT. The scheme adopts the Paillier cryptosystem to protect the privacy of users and utilizes ECDSA signatures to improve efficiency. However, its has large computation and communication overhead. Lyu et al. [34] introduced a data aggregation model called PPFA, which is based on an edge-computing architecture that prevents the leakage of private user data in aggregate statistics by incorporating a mechanism for adding DP noise. The model's edge nodes can periodically collect data from connected smart meters and aggregate them in the cloud, ensuring user privacy while saving communication costs. However, these methods cannot be effectively applied to secure data aggregation for dynamic high-dimensional datasets.

The privacy preservation of DP for dynamic high-dimensional data has become a significant area of research in recent years, as real-world applications often involve high-dimensionality and dynamic data. For example, the framework proposed by Ren et al. [11] achieves real-time decentralized statistical estimation while preserving privacy for dynamic high-dimensional data using the Laplace mechanism and Kalman consistency information filtering (KCIF), but it is not suitable for IoT data aggregation due to computing complexity. Imtiaz et al. [35] developed and implemented an online health prediction system that utilizes DP and federated learning techniques to predict dietary habits and the health data of users from fitness tracking apps and wearable devices based on high-dimensional health data streams. However, the method encounters high latency when trying to achieve good performance due to the generation of a large number of communication rounds. Therefore, we developed an DP-based data aggregation method on edge computing, which aims to address the challenges of ensuring data privacy, availability, and efficiency when aggregating dynamic high-dimensional data.

3. Preliminaries and Problem Definition

This section introduces our preliminaries and notions. We also define the problem precisely. The mathematical symbols frequently used in this paper are summarized in Table 1.

Table 1. Summary of symbols.

Symbol	Description
ϵ	Privacy budget for a dataset
UE_j	j-th user equipment (UE)
D_j	Data in UE_j
$\|D_j\|$	Data volume of D_j
ϵ_j	Privacy budget for dataset D_j
$\epsilon_j(k)$	Privacy budget for k-th attribute of D_j
d	Original dimensionality
l	Reduced dimensionality
n	Total items in the data record
$X(n_j, d)$	The original high-dimensional data in UE_j
$Y(n_j, l)$	The data after dimensionality reduction in UE_j
$Y^{'}(n_j, l)$	The noise perturbation dataset of $Y(n_j, l)$
Z_i	The data aggregated in i-th edge server
Z	The data aggregated in cloud server
S_i^t	The set of UE selected by the i-th base station at time t for uploading data
$UE(i, t)$	All sets of UE covered by the i-th base station at time t that can be selected
w_t	Window at time t
$\|w_t\|$	Size of w_t
$SV_j^t(k)$	Shapley value of k-th attribute in D_j within window w_t

3.1. Differential Privacy

Differential privacy (DP) technology can ensure that inserting or deleting a record from a database does not affect the query result, thereby guaranteeing its privacy. The relevant definitions and theorems are presented below.

Definition 1 (Neighboring datasets)**.** *Two datasets D and D' are neighboring datasets (neighbors) if $D \subset D'$ and $|D'| = |D| + 1$ or vice versa.*

Definition 2 (ϵ-DP [36])**.** *Two datasets D and D' are neighbors. Privacy algorithm F gives ϵ-DP if any output O of algorithm F of D and D' satisfies the following inequality:*

$$Pr[F(D) = O] \leq exp(\epsilon) \times Pr[F(D') = O]. \tag{1}$$

The smaller the parameter ϵ, the closer the probabilities of $F(D) = O$ and $F(D') = O$, indicating a higher level of privacy preservation provided by algorithm D.

Noise perturbation is the primary mechanism employed to achieve the preservation of DP. Global sensitivity, as a fundamental parameter, significantly affects the noise mechanism.

Definition 3 (Sensitivity [8])**.** *For f: $D \to \mathbb{R}^d$, the global sensitivity of function f is*

$$\Delta f = \max_{D,D'} \parallel f(D) - f(D') \parallel_1, \tag{2}$$

where $\mathbb{R}$ represents the mapped real number space, and d denotes the query dimension of function f. Specifically, when $d = 1$, the sensitivity of f is the maximum difference between the values that the function f may take on a pair of neighbor datasets.

There are three noise mechanisms commonly used in DP, namely, the Laplace mechanism, the Gaussian mechanism, and the exponential mechanism [37]. The exponential mechanism is typically employed for non-numerical data, which was beyond the scope of this study. It should be noted that the Laplace mechanism provides ϵ-DP, while the Gaussian mechanism offers (ϵ, δ)-DP. The notion of the privacy provided by Gaussian noise has a more relaxed definition compared to that of Laplacian noise. However, considering

its proximity to the original data, we opted for using the Laplace mechanism to perturb private data.

The Laplace mechanism achieves ϵ-DP through the actual output value of the noise perturbation generated by the Laplace distributions, as shown in Theorem 1.

Theorem 1 (Laplace mechanism [8]). *For $f: D \to \mathbb{R}^d$, algorithm F satisfies ϵ-DP if the output of algorithm F satisfies the following equation:*

$$F(D) = f(D) + (Y_1, Y_2, \ldots, Y_d), \tag{3}$$

where $Y_i \sim Lap_i(\Delta f/\epsilon)$ $(1 \leq i \leq d)$ is an independent Laplacian variable, and the amount of noise is proportional to Δf and inversely proportional to ϵ.

Theorem 2 (Parallel composition [38]). *Suppose f satisfies $\epsilon - DP$ and f_i satisfies $\epsilon_i - DP$, where f_i is a DP function of D_i, which is a disjoint subset of dataset D, $D_i \subset D$, $i = 1, 2, \cdots, n$, and $D_i \neq D_j$ if $i \neq j$. Then, mechanism f defined by $f(D) = (f_1(D_1), f_2(D_2), \cdots, f_n(D_n))$ is $\max_{1 \leq i \leq n}\{\epsilon_i\} - DP$.*

Theorem 3 (Sequential composition [39]). *Suppose f satisfies $\epsilon - DP$, and f_i satisfies $\epsilon_i - DP$, where f_i is a DP function for D_i, which is a subset of dataset D. For any dataset D with n subsets of data $D_i \subset D$, $i = 1, 2, \cdots, n$, a set $f(D) = (f_1(D_1), f_2(D_2), \cdots, f_n(D_n))$ is sequentially performed on D. Then, $f(D)$ is $\sum_{i=1}^{n} \epsilon_i - DP$.*

3.2. UMAP

The uniform manifold approximation and projection (UMAP) method is a dimensionality reduction algorithm that leverages manifold learning techniques and topological data analysis [22]. UMAP constructs fuzzy topological representations for both high-dimensional data and low-dimensional embeddings of data. Subsequently, it updates the low-dimensional embedding to ensure its fuzzy topological representation aligns with that of the high-dimensional data. In the subsequent sections, we provide a brief introduction to this methodology.

3.2.1. High-Dimensional Input Space

Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$ in a high-dimensional space $\mathbb{R}^{n \times d}$, the j-th neighbor of x_i is denoted as $x_i^{(j)}$, where n represents the sample size, and d denotes the dimensionality. To construct a k-nearest neighbors (kNN) graph, we identify set $S_i = \left\{x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(k)}\right\}$ consisting of k neighboring points for each data point x_i. UMAP calculates the probability distribution between pairs of data points based on their distances in dataset X. Probabilities are initially computed as local, one-directional probabilities between a point and its neighbors in data space, then symmetrized to yield a final probability representing the relationship between pairs of points. The similarity between points in the input space is measured using either a Gaussian or a radial basis function (RBF) kernel. The probability that point x_i has its neighbor point at position x_j can be computed based on their similarity:

$$p_{j|i} = \begin{cases} \exp(-\frac{\|x_i - x_j\|_2 - \rho_i}{\varphi_i}) & \text{if} \quad x_j \in S_i \\ 0 & \text{Otherwise,} \end{cases} \tag{4}$$

where $\| \cdot \|_2$ is the ℓ_2 norm. φ_i is a local connectivity parameter set to match the local distance around its k-nearest neighbors. Parameter ρ_i denotes the distance from each point x_i to its nearest neighbor:

$$\rho_i = \min(\| x_i - x_i^{(j)} \|_2 \mid x_i^{(j)} \in S_i). \tag{5}$$

Parameter φ_i is calculated such that the total similarity of point x_i to its k-nearest neighbors is normalized. According to a binary search, φ_i satisfies

$$\sum_{j=1}^{k} \exp(-\frac{\parallel x_i - x_j \parallel_2 - \rho_i}{\varphi_i}) = \log_2 k, \tag{6}$$

where k represents the number of neighbors of x_i as determined by the user, with a default setting of $k = 15$. Noted that UMAP has a similar search for its scale using entropy as perplexity. These searches ensure that the neighborhoods of different points exhibit consistent behavior, with smaller scales assigned to points in dense regions and larger scales assigned to points in sparse regions of the dataset. Thus, UMAP assumes a uniform distribution of data points on an underlying low-dimensional manifold.

Equation (4) represents a one-directional probability measure. To achieve symmetry in the measure with respect to i and j,

$$p_{ij} = p_{j|i} + p_{i|j} - p_{j|i}p_{i|j}. \tag{7}$$

where $p_{i|j}$ is the probability that point x_i has a neighbor point x_j. p_{ij} is a symmetric probability between points x_i and x_j in the input space.

3.2.2. Low-Dimensional Embedding Space

Let the points in an $n \times d$ dataset X be embedded into another $n \times l$ set $Y = \{y_1, y_2, \ldots, y_n\}$ in a low-dimensional space $\mathbb{R}^{n \times l}$, where l ($< d$) is the dimensionality of the embedding space, and y_i is the embedding of the corresponding point $x_i \in X$.

The probability of y_i being adjacent to y_j in the embedding space with $i \neq j$ is calculated with their similarity, i.e.,

$$q_{ij} = (1 + a \parallel y_i - y_j \parallel_2^{2b})^{-1}, \tag{8}$$

which exhibits symmetry with respect to both i and j. The parameters $a > 0$ and $b > 0$ are set to fit a function:

$$\Phi(\mathrm{x}, \mathrm{y}) = \begin{cases} 1 & \text{if} \quad \parallel \mathrm{x} - \mathrm{y} \parallel_2 \leq \text{mindist} \\ \exp(- \parallel \mathrm{x} - \mathrm{y} \parallel_2 - \text{mindist}) & \text{Otherwise.} \end{cases} \tag{9}$$

where *mindist* represents the desired separation between closely located points within the embedding space. Data points that are close in high-dimensional space are mapped to a range smaller than *mindist* in low-dimensional space. This ensures a higher value of $p_{i,j}$ and a higher value of $q_{i,j}$, and vice versa. For our study, the value of *mindist* was set to 0.1.

3.2.3. Optimization

The objective of UMAP is to ensure similarity between a low-dimensional embedding space and its corresponding high-dimensional input space. Therefore, Equations (7) and (8) are treated as probability distributions, with the aim of minimizing the discrepancy between their distributions. This discrepancy represents the difference in graph similarity, which is quantified using fuzzy cross-entropy:

$$c_1 = \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} (p_{ij} \log(\frac{p_{ij}}{q_{ij}}) + (1 - p_{ij}) \log(\frac{1 - p_{ij}}{1 - q_{ij}})), \tag{10}$$

where the first term on the RHS in Equation (10) is the *attractive force* that attracts the embedding of neighboring points toward each other. This term only appears when $p_{ij} \neq 0$, implying that x_i and x_j are neighbors. The second term in Equation (10) is the *repulsive force* that repulses the embedding of non-neighbor points away from each other. Thus, the optimization objective is ensuring that the points that are close in high-dimensional

space are closer in low-dimensional space; otherwise, they are far away from each other. Stochastic gradient descent (SGD) is used for iterative optimizations, where the embedding y is updated by gradients g, with a learning rate of η [22].

3.3. Progressive UMAP

The original UMAP algorithm lacks support for embedding out-of-sample data and only allows fixed sets of data points, making it unsuitable for the gradual addition of new data points. Based on UMAP, progressive UMAP [23] enables the embedding of out-of-sample data and can be effectively utilized for dynamic (stream) data analysis.

Progressive UMAP employs a two-stage approach for initializing the positions of newly inserted points. In the first stage, it executes the algorithm with a large value of ops (e.g., ops = 15,000), utilizing the same spectral embedding technique as UMAP. Since it starts with a relatively small number of points, this step requires considerably less time compared to the original UMAP's spectral embedding process. Subsequently, in the second stage, it reduces the value of ops (e.g., ops = 1000) not to prioritize appending new points but rather to obtain an optimized projection output swiftly. From the second batch onward, each newly inserted point's initial projected position is set equal to its closest neighbor's position perturbed by slight Gaussian random noise to prevent collisions.

Analogously, the layout optimization process consists of two stages. Given its impact on convergence time and final output stability, it is crucial to accurately position the initial batch of points in order to achieve unambiguous cluster separation. To accomplish this, (1) it executes a higher number of iterations (e.g., r = 50) during the first stage to allow each cluster to settle into its optimal position. Subsequently, (2) it reduces the number of iterations (e.g., r = 5) in order to expedite the attainment of the projection result. Users can adjust the iteration numbers for each stage considering the data utility and runtime requirements. In addition, if the data size is sufficiently small, users can opt for equal iteration numbers across both stages.

3.4. The Shapley Value

Consider an $n \times m$ dataset $X(n, m)$ with n records and m attributes; each attribute's data are $X(n, x^{(k)})$ $(1 \leq k \leq m)$. We assume a utility function $\mathcal{U}(\mathcal{S})$ that evaluates the utility of a coalition $\mathcal{S}$ with $\mathcal{S} \subseteq \left\{X(n, x^{(1)}), \ldots, X(n, x^{(m)})\right\}$. The Shapley value is used to measure the marginal utility improvement contributed by $X(n, x^{(k)})$ averaged over all possible coalitions $\mathcal{S}$ [40].

Given a dataset $X(n, m)$, the Shapley value SV_k of $X(n, x^{(k)})$ $(1 \leq k \leq m)$ is calculated as follows:

$$SV_k = \frac{1}{m} \sum_{\mathcal{S} \subseteq \{X(n,m)\} | X(n,x^{(k)}) \notin \mathcal{S}} \frac{\mathcal{U}\left(\mathcal{S} \cup \left\{X(n, x^{(k)})\right\}\right) - \mathcal{U}(\mathcal{S})}{\binom{m-1}{|\mathcal{S}|}}, \tag{11}$$

where $\mathcal{U}\left(\mathcal{S} \cup \left\{X(n, x^{(k)})\right\}\right)$ represents the data utility when the data contain $X(n, x^{(k)})$. The Shapley value can be employed to assess data utility by allocating the privacy budget [41]. Here, $\mathcal{U}(\cdot)$ denotes the data utility function, which corresponds to the classification success rate, defined as the ratio of correctly classified samples to the total number of samples in a given dataset.

3.5. Problem Definition

This method is based on an edge computing network denoted as $G = (C, V, E)$, where C represents the cloud node, V denotes the set of edge nodes, and E represents the set of connections between nodes. Each edge node $(V_i \in V)$ functions as a base station and is accompanied by a corresponding edge server. The base station and its associated edge server are connected via high-speed optical cables, ensuring negligible communication

delays. Moreover, each base station provides wireless connectivity to user equipment (UE) that chooses to participate in data aggregation. Let UE_j denote the UE and D_j represent the data stored on that equipment, where j is an integer ranging from 1 to n_u (the total number of UE). The size of D_j is denoted as $|D_j|$.

Users within the coverage of the base station can upload their equipment data D_j from UE UE_j to the corresponding edge server, where $UE_j \in S_i^t$ (S_i^t represents the set of UE selected by the i-th base station at time t to upload data). Each edge server aggregates user data into dataset Z_i, which is then uploaded to the cloud server for collection and integration in preparation for subsequent data analysis. Assume that the user data consist of an original dataset $X(n_j, d)$ with dimensions represented by $n_j \times d$, where each sample in equipment data D_j corresponds to a dimension attribute, and there are time stamps ranging from $t = 0$ to $t = T$. Firstly, the dynamic data are divided into initial time windows denoted as w_t ($1 < t < T$), with the unit window size defined as $|w|$. Secondly, the prediction of the data volume in the time window is utilized to change the window size dynamically for each UE's data uploading, which determines S_i^t. Thirdly, dimensionality reduction is applied to each window's dataset, resulting in a transformed dataset denoted as $Y(n_j, l)$, where $l < d$, and $UE_j \in S_i^t$. Finally, while ensuring individual privacy requirements are met, DP noise perturbation is introduced after dimensionality reduction for each window's data. However, it should be noted that both dimensionality reduction and DP noise perturbation may lead to reduced utility when aggregating dynamic high-dimensional data in edge computing scenarios. Moreover, reusing a dimensionality reduction model for each window could impact efficiency. Therefore, effectively allocating the privacy budget to minimize utility loss in the transformed dataset $Y(n_k, l)$ after dimensionality reduction is an urgent problem requiring resolution.

4. Edge-Based Differential Privacy Data Aggregation Method

In this section, we present the proposed EDP-PUDL method, an edge-based differential privacy data aggregation approach that utilizes LSTM and progressive UMAP with DP for aggregating dynamic high-dimensional data. We commence with an overview of the proposed method, followed by a detailed explanation and ultimately an analysis of its privacy.

4.1. Overview of the Proposed Method EDP-PUDL

In our proposed EDP-PUDL method, the dynamic data in each UE are segmented using a dynamic time window approach based on LSTM, and the window's data's dimensions are reduced through progressive UMAP. For privacy budget allocation, two aspects are primarily considered: one is allocating the privacy budget based on the amount of data in the equipment covered by each base station to ensure the availability of aggregated data; the other is calculating the Shapley value of specific equipment's attribute data within the window for privacy budget allocation to minimize the loss of data utility.

4.2. EDP-PUDL

The main process of the EDP-PUDL method presented in this paper is outlined in Figure 2 and Algorithm 1. Firstly, various UE in an IoT system generate their actual dynamic high-dimensional data. Secondly, a dynamic time window method based on LSTM was designed to select UE to collect data at each time stamp. Thirdly, progressive UMAP is utilized to reduce data dimension and present a privacy budget allocation method to perturb the data using DP noise. Lastly, DP-perturbed dynamic low-dimensional data are uploaded to edge servers and aggregated in a cloud server.

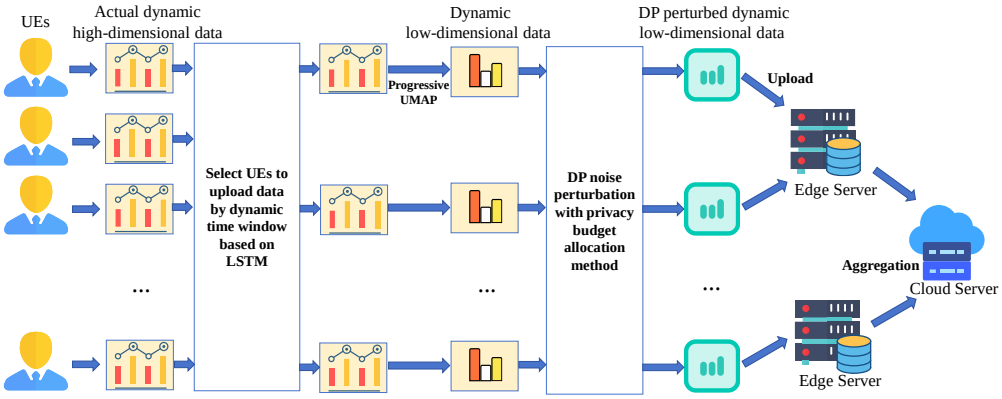

Figure 2. The EDP-PUDL process.

Algorithm 1 Edge-based differential privacy data aggregation method.

Input: Edge computing network $G = (C, V, E)$, given finite time range T, initial window size $|w|$, and local dataset D_j in each UE UE_j.
Output: Privatized aggregated dataset Z.

1: Initialize dataset D_j in each UE when $t = 0$;
2: $t \leftarrow 1$;
3: **while** $t \leq T$ **do**
4: **while** $UE_j \in UE(i,t)$ (in parallel) **do**
5: **if** $|D_j| \neq 0$ **then**
6: Apply dynamic time window based on LSTM to calculate S_i^t and perform dimension reduction: $Y^t(n_j, l) \leftarrow X^t(n_j, d)$, by Algorithm 2;
7: Calculate privacy budget $\epsilon_j^t(k)$ using Algorithm 3;
8: Add Laplacian noise: $Y'^t(n_j, k) \leftarrow Y^t(n_j, k) + Lap_j(\Delta f / \epsilon_j^t(k))$;
9: Merge data: $Y'^t(n_j, l) \leftarrow \sum_{k=1}^{l} Y'^t(n_j, k)$;
10: Upload $Y'^t(n_j, l)$ to the edge server;
11: Process all $Y'^t(n_j, l)$ and aggregate them in the edge server to obtain Z_i^t;
12: Upload data Z_i^t in each edge server to the cloud server;
13: Process all Z_i^t and aggregate them in cloud server to obtain Z^t;
14: **end if**
15: **end while**
16: $t \leftarrow t + 1$;
17: **end while**
18: $Z \leftarrow \sum_{t=1}^{T} Z^t$;
19: **return** Z.

In Algorithm 1, the initialization stage (step 1) involves the original high-dimensional dataset without any privacy processing. Step 2 indicates the next time stamp. Steps 3 to 17 represent the edge server aggregation phase, where the secure collection of dynamic high-dimensional datasets from equipment covered by base stations takes place. Steps 4 to 15 demonstrate that UE with a time window covered by a base station synchronously upload their datasets, which are securely collected by an edge server. The datasets are not collected if no data are generated within an equipment's time window coverage. Steps 5 to 14 describe the secure aggregation process of user data by the edge server when there is a need to upload data from the user's equipment. Here, step 6 applies the dynamic time window method based on LSTM to obtain the UE selection set S_i^t. $X^t(n_j, d)$ in step 6 represents the dataset before dimensionality reduction of user j's equipment's data in window w_t, and $Y^t(n_j, d)$ represents the dataset after dimensionality reduction. Step 7 involves the privacy budget allocation of adding noise to the data. For the dimensionally reduced data, their privacy budget allocation is calculated based on the Shapley value

of their attribute, enabling DP noise addition for different attributes. Step 8 introduces $Y'^t(n_j,k)$ as the noisy version of attribute k's data in window w_t for equipment j, while $Y^t(n_j,k)$ refers to its original non-noisy counterpart. Additionally, Laplacian noise is denoted as $\mathrm{Lap}_j(\Delta f/\epsilon_j^t(k))$ to perturb $Y^t(n_j,k)$. Step 9 merges all attribute data within equipment j into a combined dataset represented by $Y'^t(n_j,l)$. In Step 11, the i-th edge server processes $Y'^t(n_j,l)$ via cleaning and transforming, then aggregates the data within window w_t, resulting in aggregate data denoted as Z_i^t. In step 13, the cloud server collects and merges aggregated datasets from all edge servers, forming a merged dataset Z^t.

4.2.1. Progressive UMAP with Dynamic Time Window Based on LSTM

The proposed approach for dynamic data dimensionality reduction is progressive UMAP with dynamic time windows based on LSTM, as illustrated in Algorithm 2. Step 2 sets the first upload time stamp t_1. Steps 3 to 8 show that the method is performed in the first time window. The progressive UMAP dimension reduction parameters (ops) and iteration times (r) are set in steps 4 and 5; by applying the progressive UMAP method [23], the dimension of UE data in this window is reduced from d to l in step 6. It utilizes LSTM for the dynamic time window to update window size in step 7. Steps 9 to 16 represent that the method is performed in subsequent windows. Firstly, new dimension reduction parameters (ops) and iteration times (r) are set, which are significantly smaller than those used in the first window. Then, it updates the window size for UE_j by using the previous predicted value of the data volume and calculates the UE selection set S_i^t of each edge server at time stamp t in step 12. Progressive UMAP is utilized to reduce data dimensions in step 13. Lastly, it utilizes LSTM for the dynamic time window based on LSTM to update window size in step 14.

Algorithm 2 Progressive UMAP with dynamic time windows based on LSTM.

Input: The local dataset D_j in each UE UE_j, and the high-dimensional dataset $X^t(n_j,d)$ before the UE data are reduced in window $w_{j,t}$.

Output: Low-dimensional dataset $Y^t(n_j,l)$ of UE data after dimensionality reduction in window $w_{j,t}$, $l < d$.

1: Initially divide time window $w_{j,t}$ according to the initial window size $|w_{j,1}|$;
2: $t_1 \leftarrow |w_{j,1}|$;
3: **if** $t = t_1$ **then**
4: Set parameter $ops \leftarrow ops_1$;
5: Set the number of iterations $r \leftarrow r_1$;
6: Progressive UMAP dimension reduction: $Y^t(n_j,l) \leftarrow X^t(n_j,d)$;
7: Utilize the dynamic time window based on LSTM to update window size $|w_{j,t+1}|$;
8: **end if**
9: **if** $t \neq t_1$ **then**
10: Set parameter $ops \leftarrow ops_2$, where $ops_2 << ops_1$;
11: Set the number of iterations $r \leftarrow r_2$, where $r_2 << r_1$;
12: Calculate the UE selection set S_i^t using updated window size $|w_{j,t}|$;
13: Progressive UMAP dimension reduction: $Y^t(n_j,l) \leftarrow X^t(n_j,d)$;
14: Utilize the dynamic time window based on LSTM to update window size $|w_{j,t+1}|$;
15: **end if**
16: **return** $Y^t(n_j,l)$.

(1) Stacked LSTM model

The LSTM network is utilized for data prediction. The logical architecture diagram of the LSTM cell is shown in Figure 3 (left part). Owing to the presence of the forgetting gate in the LSTM unit, the network can rapidly assimilate novel features from data, dynamically update its parameters, and ensure high precision when predicting streaming data. Employing stacked hidden layers further enhances model depth, leading to more accurate

outputs. We employ an LSTM model with stacked layers for the sequential processing of window batches, maintaining the same architectural design as in our previous work [26], as shown in Figure 3 (right part).

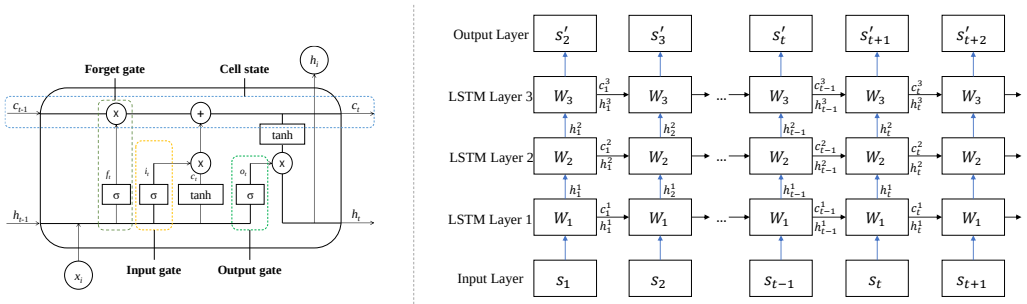

Figure 3. The structures of LSTM cell and stacked LSTM model.

The stacked LSTM architecture can be defined as an LSTM model comprising multiple layers, where the upper LSTM layer generates a sequence of outputs instead of a single-value output to be passed on to the lower LSTM layer. The output of the hidden layer is not only propagated forward through time but also utilized as one of the inputs for the subsequent hidden LSTM layer. Specifically, at each time stamp t, in the l-th layer, the hidden state h_t^l is updated via the fusion of data s_t, input gate i_t^l, forget gate f_t^l, output gate o_t^l, memory cell c_t^l, and the hidden state in the previous time stamp h_{t-1}^l. The updated equations are as follows:

$$i_t^l = \sigma\left(W_i^l \cdot \left[h_{t-1}^l, s_t\right] + b_i^l\right), \tag{12}$$

$$f_t^l = \sigma\left(W_f^l \cdot \left[h_{t-1}^l, s_t\right] + b_f^l\right), \tag{13}$$

$$o_t^l = \sigma\left(W_o^l \cdot \left[h_{t-1}^l, s_t\right] + b_o^l\right), \tag{14}$$

$$c_t^l = f_t^l * c_{t-1}^l + i_t^l * \tanh\left(W_c^l \cdot \left[h_{t-1}^l, s_t\right] + b_c^l\right), \tag{15}$$

$$h_t^l = o_t^l * \tanh\left(c_t^l\right), \tag{16}$$

where W_i^l, W_f^l, W_o^l, and W_c^l are weight matrices; b_i^l, b_f^l, b_o^l, and b_c^l are bias vectors; σ is a sigmoid activation function; tanh is a hyperbolic tangent function; and $*$ means the elementwise product.

The stacked LSTM is employed to predict the data volume of each time window, thereby facilitating the adjustment of the window size.

(2) Dynamic time window based on LSTM

For each UE, the data volume of each time window is predicted by utilizing the stacked LSTM model to train the data in previous windows. Then, it updates the window size based on the predicted value. The time window size is updated as follows:

$$l_{j,t} = \lambda \cdot \frac{s'_{j,t} - s_j}{s_j} \cdot |w_j|, \tag{17}$$

$$L_{j,t} = \begin{cases} |w_j| & |\frac{s'_{j,t}-s_j}{s_j}| > \theta \\ |w_j| + l_{j,t} & |\frac{s'_{j,t}-s_j}{s_i}| \leq \theta, \end{cases} \tag{18}$$

where $L_{j,t}$ is the updated window size for UE_j at time t, $|w_j|$ is the fixed part of the window (i.e., the initial window size) for UE_j, $l_{j,t}$ is the variable part of the window of UE_j at time t, s_j is the average statistic value of the data volume in UE_j, and $s'_{j,t}$ is the predicted value of

the data volume in UE_j at time t. λ is the smoothing factor that prevents $l_{j,t}$ from being too large or too small, which makes $l_{j,t}$ a positive integer, and θ is the comparison parameter.
(3) UE selection for dimensionality reduction

A smaller data volume poses a higher risk of privacy leakage. Therefore, the time window sizes of UE are updated dynamically so that not all UE may be selected in the same time stamp.

Firstly, UE is selected at each time stamp when the windows arrive, and the UE selection set S_i^t is obtained. Secondly, on the data in each window $w_{j,t}$, progressive UMAP with DP is performed to reduce dimensions while preserving privacy, $Y'^t(n_j, l) \leftarrow X^t(n_j, d)$. Thirdly, by using the private low-dimensional data of each window, a dynamic time window based on LSTM is applied to update the next window size $w_{j,t+1}$. Repeat the above steps until time T.

All UE covered in access point i ($UE(i,t)$) participates in the UE selection. The UE selection set S_i^t is determined by the updated window size of the UE. When windows arrive at time t, $|w_{j,t}| = L_{j,t}$ and UE_j are selected and added to set S_i^t. As an example, in Figure 4, the time window sizes of three UE (UE_1,UE_2, UE_3) are updated. UE_2 is selected to upload data without UE_1 and UE_3 when $t = 5$, due to UE_1 and UE_3 not having enough data at that time. UE_1 and UE_2 are selected without UE_3 when $t = 9$, due to UE_3 not having enough data at that time.

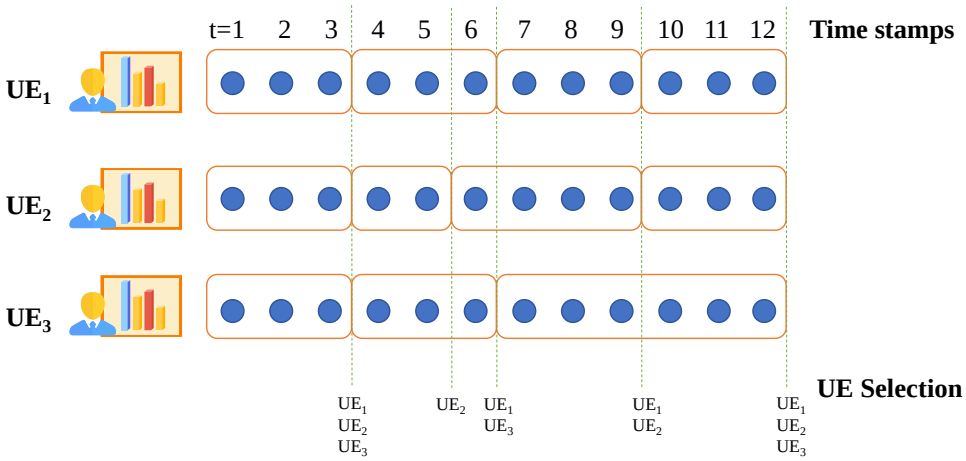

Figure 4. An example of UE selection.

4.2.2. Privacy Budget Allocation in Edge-Based Data Aggregation

Algorithm 3 presents the allocation of the privacy budget, which involves two main steps: firstly, the privacy budget ϵ_j^t is computed for the reduced dataset in window w_t, which needs to be assigned to UE j; secondly, the privacy budget $\epsilon_j^t(k)$ is determined and allocated to each attribute's data.
(1) Privacy budget allocation for UE

The privacy budget allocated to the UE in edge computing is represented by ϵ_j^t. The calculation of ϵ_j^t is as follows:

$$\epsilon_j^t = \alpha \cdot e^{\omega_j^t}, \tag{19}$$

where α serves as the regulatory factor. If a customized total privacy budget value is set, the size can be adjusted accordingly to ensure that data privacy is not compromised by exceeding the total privacy budget. The assigned weight value ω_j^t can be computed using the following formula:

$$\omega_j^t = 1 - \frac{|D_j'(t)|}{\sum_{j=1}^{|S_i^t|} |D_j'(t)|}, \tag{20}$$

where $|D'_j(t)|$ is the amount of data of the j-th UE after dimensionality reduction in window w_t.

Algorithm 3 Privacy budget allocation in edge-based data aggregation.

Input: Low-dimensional dataset $Y^t(n_j, l)$ that of data D_j after dimensionality reduction in window w_t.
Output: The privacy budget $\epsilon_j^t(k)$ is allocated to each attribute's data of $Y^t(n_j, l)$.

1: Calculate the amount of data $|D'_j(t)|$ on UE UE_j in window w_t after dimensionality reduction;
2: Calculate budget allocation weight ω_j^t based on data $|D'_j(t)|$ using Equation (20);
3: Calculate the privacy budget ϵ_j^t based on ω_j^t using Equation (19);
4: Calculate the Shapley value $SV_j^t(k)$ for the k-th attribute's data in $Y^t(n_j, l)$ using Equation (11);
5: Calculate preliminary privacy budget $\epsilon_j^{o,t}(k)$ for each attribute based on $SV_j^t(k)$ using Equation (22);
6: **if** $\epsilon_j^{o,t}(k) \geq \epsilon_j^t$ **then**
7: $\quad \epsilon_j^t(k) \leftarrow \epsilon_j^t$;
8: **else if** $\epsilon_j^{o,t}(k) < \epsilon_j^t$ **then**
9: $\quad \epsilon_j^t(k) \leftarrow \epsilon_j^{o,t}(k)$;
10: **end if**
11: **return** $\epsilon_j^t(k)$.

According to Equations (19) and (20), the allocation of the privacy budget for a dataset is determined by its size. This is because larger datasets require more noise interference to ensure data privacy (more noise implies a reduced privacy budget). Conversely, smaller datasets cannot tolerate excessive noise that may render the data unusable. Therefore, allocating privacy budgets based on data volume ensures both overall data privacy and availability.

(2) Privacy budget allocation for attributes

After dimensionality reduction, various attributes' data still have different utility in the dataset. To further reduce the loss of data utility, the Shapley value is calculated to evaluate the utility of each attribute's data to allocate the privacy budget. According to Equation (11), the Shapley value $SV_j^t(k)$ of the k-th ($1 \leq k \leq l$) attribute of UE_j in window w_t is calculated as follows,

$$SV_j^t(k) = \frac{1}{l} \sum_{\mathcal{S} \subseteq \left\{Y^t(n_j,l)\right\} | Y^t(n_j,x^{(k)}) \notin \mathcal{S}} \frac{\mathcal{U}\left(\mathcal{S} \cup \left\{Y^t(n_j, x^{(k)})\right\}\right) - \mathcal{U}(\mathcal{S})}{\begin{pmatrix} l-1 \\ |\mathcal{S}| \end{pmatrix}}, \tag{21}$$

where $\mathcal{U}(\mathcal{S})$ is a utility function that evaluates the utility of a coalition $\mathcal{S}$ with $\mathcal{S} \subseteq \left\{Y^t(n_j, x^{(1)}), \ldots, Y^t(n_j, x^{(l)})\right\}$. $\mathcal{U}\left(\mathcal{S} \cup \left\{Y^t(n_j, x^{(k)})\right\}\right)$ represents the data utility when the data contain $Y^t(n_j, x^{(k)})$. Here, $\mathcal{U}(\cdot)$ denotes the data utility function, which corresponds to the classification success rate.

Based on the Shapley value, the initial privacy budget is determined and allocated to each attribute's data as $\epsilon_j^{o,t}(k)$. Subsequently, the final privacy budget is calculated and assigned to each attribute's data as $\epsilon_j^t(k)$ using the following equations:

$$\epsilon_j^{o,t}(k) = \beta \cdot e^{SV_j^t(k)}, \tag{22}$$

$$\epsilon_j^t(k) = \min(\epsilon_j^{o,t}(k), \epsilon_j^t), \tag{23}$$

where β is the regulatory factor, acting the same as α.

According to Theorem 2, $\epsilon_j^t = \max \epsilon_j^t(k)$, so $\epsilon_j^t(k) \leq \epsilon_j^t$. If $\epsilon_j^{o,t}(k) \geq \epsilon_j^t$, which is calculated using Equation (22), $\epsilon_j^t(k) = \epsilon_j^t$.

To ensure data privacy and enhance data availability, Laplacian noise, denoted as $Lap_j(\Delta f/\epsilon_j^t(k))$, is perturbed for each attribute in the dataset by allocating a privacy budget $\epsilon_j^t(k)$. The allocation of the privacy budget $\epsilon_j^t(k)$ depends on the Shapley value of an attribute. The attribute data with a higher Shapley value are allocated a larger privacy budget. In other words, attributes data with higher utility experience less noise perturbation, thereby reducing the loss of data availability.

(3) A calculation example of privacy budget allocation

The calculation of a privacy budget is exemplified in Table 2. It considers two UE, UE_1, and UE_2, within a given window w_t. After dimensionality reduction, the datasets in UE_1 and UE_2 are denoted as $D_1'(t)$ and $D_2'(t)$. Both datasets have the same three attributes ($l = 3, k = 1, 2, 3$) but differ in sample sizes ($n_1 \neq n_2$), resulting in unequal cardinalities ($|D_1'(t)| \neq |D_2'(t)|$). Assuming that $|D_1'(t)| = 10$ and $|D_2'(t)| = 20$, the various attributes of Shapley values are 0.9, 0.6, 0.3, 0.3, 0.3, and 0.9. Since there is no predefined total budget available by default for this scenario, we set $\alpha = \beta = 1$ as the default value.

Table 2. An example of privacy budget allocation for data aggregation.

UE_j	$\lvert D_j'(t)\rvert$	ω_j^t	ϵ_j^t	k	$SV_j^t(k)$	$\epsilon_j^t(k)$
UE_1	10	0.66	1.93	1	0.9	1.93
				2	0.6	1.82
				3	0.3	1.35
UE_2	20	0.33	1.39	1	0.3	1.35
				2	0.3	1.35
				3	0.9	1.39

By examining the example in Table 2, it is evident that after reducing the dimensions of data $D_j'(t)$, UE UE_j in window w_t has a smaller amount of data, with $|D_1'(t)| = 10$, compared to equipment UE_2, which has a larger amount of data, with $|D_2'(t)| = 20$. Consequently, the privacy budget $\epsilon_1^t = 1.93$ allocated for equipment UE_1 is higher than the $\epsilon_2^t = 1.39$ assigned to equipment UE_2. This implies that a smaller amount of data $|D_j'(t)|$ receives a greater privacy budget ϵ_j^t, resulting in less noise disturbance. Such an allocation ensures that larger datasets are not vulnerable to privacy breaches due to insufficient preservation while smaller datasets do not suffer from the poor availability caused by excessive preservation measures. Considering that different attributes within the dataset contribute differently to utility, attribute utility is assessed using Shapley value calculations for allocating privacy budgets. As observed in the table, $SV_1^t(1) < SV_1^t(2) < SV_1^t(3)$ and, consequently, $\epsilon_1^t(1) < \epsilon_1^t(2) < \epsilon_1^t(3)$. This indicates that attributes with higher utility receive a larger privacy budget (resulting in less noise), thereby ensuring the high availability of the data. To guarantee data privacy, the threshold for each attribute's privacy budget should not exceed the overall dataset's privacy budget ($\epsilon_j^t(k) \leq \epsilon_j^t$).

4.3. Privacy Analysis

This part proves that the proposed EDP-PUDL method satisfies DP through the following theorems.

Theorem 4. *The proposed EDP-PUDL method satisfies ϵ-DP and adds Laplacian noise to the user data after dimensionality reduction for data aggregation when UE data are uploaded to an edge server.*

Proof of Theorem 4. After the dimensionality reduction of the data, Laplacian noise is added to the data $Y_{n_j,k}$ for each attribute on the low-dimensional dataset $Y_{n_j,l}$ according to

the Shapley value $SV_j(k)$. According to Definition 3, the sensitivity is $\max\limits_{1\le a\le n_j}\sqrt{\sum_{k=1}^{l} Y_{a,k}^2}$, where a is a sample variable; the computation is as follows:

$$\begin{aligned}\| f(D_j) - f(D_j^1) \|_2 &= \| M \cdot Y_{n_j,l} - M^1 \cdot Y_{n_j,l} \|_2 \\ &= \| (M - M^1) \cdot Y_{n,l} \|_2 \\ &\le \max_{1\le a\le n_j} \sqrt{\sum_{k=1}^{l} Y_{a,k}^2} \\ &= \Delta f.\end{aligned}$$

where D_j' is D_j's neighbor datasets, M and M' is two input matrix, both differ one element $m_{u,v}$, corresponding to the user u, v.

The privacy budget allocated to each attribute data of UE_j in the dataset in window w_t is $\epsilon_j^t(k) = \beta \cdot \min(\epsilon_j^t, e^{SV_j^t(k)})$. According to Definition 2, each attribute satisfies $\epsilon_j(k)$-DP. Based on Theorem 2, the privacy budget $\epsilon_i^t = \max(\epsilon_j^t)$ should be allocated to all UEs covered by each base station in each time window after dimensionally reduced data. The privacy budget for the cloud aggregate dataset that needs to be allocated to each time window is $\epsilon^t = \max(\epsilon_i^t)$. Furthermore, according to Theorem 3, the total privacy budget is calculated as $\epsilon = \sum_{t=1}^{T} \epsilon^t$. As it holds that $\epsilon_j^t(k) \le \epsilon_j^t \le \epsilon_i^t \le \epsilon^t \le \epsilon$, finally, EDP-PUDL satisfies ϵ-DP. □

5. Experimental Analysis

The performance of the proposed method EDP-PUDL was evaluated against that of several representative methods through a comprehensive set of experiments conducted on multiple public datasets. Firstly, we introduce the experimental environment setup and performance evaluation metrics. Secondly, we present and analyze the experimental results under different parameter settings.

5.1. Experimental Step

All methods were implemented in Python 3.7 on a machine with AMD Ryzen 7 6800H CPU 3.20 GHz, NVIDIA GeForce RTX 3060 GPU, 16GB RAM, and Windows 11. Open datasets were selected for experimental datasets: Adult (https://archive.ics.uci.edu/ml/datasets/adult/, accessed on 10 January 2024), ACS (https://usa.ipums.org/usa/, accessed on 10 January 2024), TPC-E (http://www.tpc.org/, accessed on 10 January 2024), and NLTCS (https://www.icpsr.umich.edu/web/NACDA/studies/9681/, accessed on 10 January 2024). The Adult dataset contains 45,222 entries of personal information from the U.S. census, the ACS dataset contains 47,461 items of personal information from IPUMS-USA's ACS sample set, the TPC-E dataset contains the records of 40,000 transactions from an online processing program, and the NLTCS dataset contains the records of 21,574 individuals who participated in the National Long-term Care Survey. The statistics of the datasets are summarized in Table 3.

Table 3. Datasets' characteristics.

Dataset	Cardinality	Dimensions
Adult	45,222	15
ACS	47,461	23
TPC-E	40,000	55
NLTCS	21,574	16

We set up five edge nodes, each covering 5 to 10 pieces of UE, and we randomly selected equipment to upload data. To mitigate the risk of privacy breaches while ensuring

data availability, we enforced a minimum variance threshold of 10% between the privacy-processed data and the original data, alongside maintaining a misclassification rate below 0.5 in all cases. Thus, the privacy budget ϵ ranged from 0.1 to 1.6, and the initial window size ranged from 50 to 100. The value of k was set to 3 by default, $ops_1 = 15{,}000$, $ops_2 = 1000$ in progressive UMAP, and the number of iterations $r_1 = 100$ and $r_2 = 10$. The value in the experiment diagram is the average value of 50 repetitions for each experiment.

5.2. Evaluation Metrics

To evaluate the privacy preservation effect of the algorithm and to measure the data availability, we used the variance in the statistics of the data in their windows (σ^2), the cumulative value of the average L_2 -Error ([16]) for all time windows at time T (TL_2Error), the cumulative value of the average variable distance [16] for all time windows at time T ($TAVD$), and the SVM misclassification rate. The variance in the data statistics was used to measure the balance of the data between windows as follows,

$$\sigma^2 = \frac{\sum_{j=1}^{N}(s_j - s)^2}{N} \tag{24}$$

where s_j is the data statistics in each window, and s is the average value of the data statistics.

The data availability was evaluated using $TL2Error$ for the binary datasets (i.e., ACS and NLTCS) and $TAVD$ for the nonbinary datasets (i.e., Adult and TPC-E). $TL2Error$ and $TAVD$ were calculated as follows:

$$TL2Error = \sum_{t=1}^{T} L2Error(t) \tag{25}$$

where $L2Error(t)$ refers to the L_2-Error of the data in time window w_t.

$$TAVD = \sum_{t=1}^{T} avd_t \tag{26}$$

where avd_t refers to the average variable distance of the data in time window w_t.

In addition, the classification results were evaluated using an SVM classifier, and classification performance was measured using the misclassification rate (the second attribute of each dataset was set to the classification attribute), which reflected the data availability.

5.3. Evaluation Results

In this subsection, the results and an analysis of the comparison experiments with existing algorithms are given, and the performance of the proposed method is analyzed in detail. Firstly, we experimentally compared the three models for dividing data streams using the variances in the dynamic time window based on LSTM (DTW-LSTM), fixed time window (FTW), and dynamic time window based on data speed (DTW-speed) [42]. Then, we employed representative dimensionality reduction methods based on the dynamic time window of LSTM. We employed linear dimensionality reduction techniques, namely, principal component analysis (PCA) [43] and random projection, and a nonlinear dimensionality reduction technique, t-distributed stochastic neighbor embedding (t-SNE) [44], to reduce the dimensionality of the data, and we added DP noise to each window's data after dimensionality reduction, i.e., PCA-DP, DPPro [16], and tSNE-DP. In addition, DP noise was added to attribute data on average after progressive UMAP as another comparison method, i.e., PU-AveDP.

5.4. Variance

The DTW-LSTM model was compared with the FTW and DTW-speed models on the Adult dataset. In the FTW model, the window size remains constant and equal to the initial window size, while in the DTW-speed model, the current window size dynamically adjusts

based on data speed calculated from the previous windows. We incrementally increased the window size from 50 to 100 in steps of 10 for all three methods.

As depicted in Figure 5, our proposed DTW-LSTM model consistently achieved significantly lower variance across all cases than both the FTW and DTW-speed models, outperforming them by 48.8% and 23.7%, respectively, on average. Moreover, as the initial window size increased, DTW-LSTM demonstrated superior performance in terms of variance compared to both the FTW and DTW-speed models due to its ability to leverage larger amounts of historical data to improve prediction accuracy using LSTM.

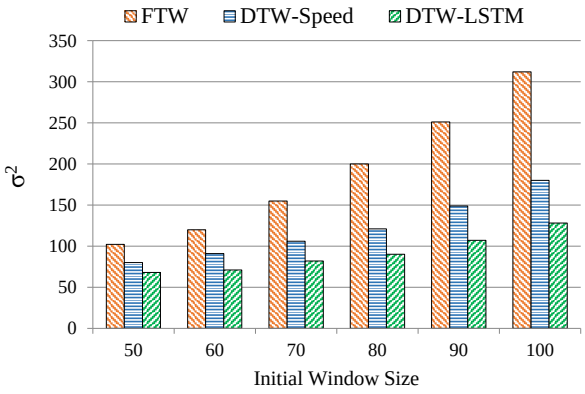

Figure 5. Variance in Adult dataset.

5.5. TAVD and TL2Error

This subsection compares the *TAVD* and *TL2Error* of the PCA-DP, tSNE-DP, and EDP-PUDL methods over time T on different datasets. The number of attributes after dimension reduction was set to 3, the privacy budget ϵ was set to range from 0.1 to 1.6, and the initial window size was 50 to 100.

First, *TAVD* and *TL2Error* on the Adult, TPC-E, ACS, and NLTCS datasets were compared by changing the total privacy budget ϵ (the initial window size $|w| = 70$) for PCA-DP, tSNE-DP, and EDP-PUDL. As shown in Figure 6, EDP-PUDL had a lower *TAVD* for all privacy budgets ϵ compared with the other methods on the Adult and TPC-E datasets. On the Adult dataset, EDP-PUDL outperformed the other methods by 20.6% and 10.5% on average. As shown in Figure 7, EDP-PUDL outperformed the other methods for all privacy budgets ϵ on the ACS and NLTCS datasets. On the ACS dataset, EDP-PUDL outperformed other methods by 42.9% and 20.7% on average. Compared with PCA-DP, tSNE-DP, and DPPro, EDP-PUDL had smaller errors and higher data availability.

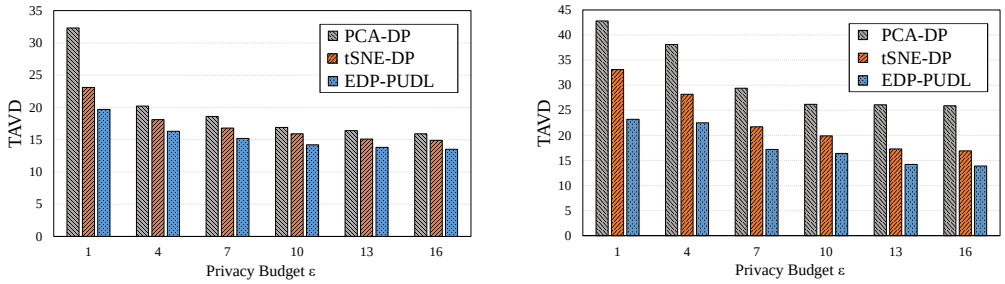

Figure 6. *TAVD* with various privacy budgets on the Adult and TPC-E datasets.

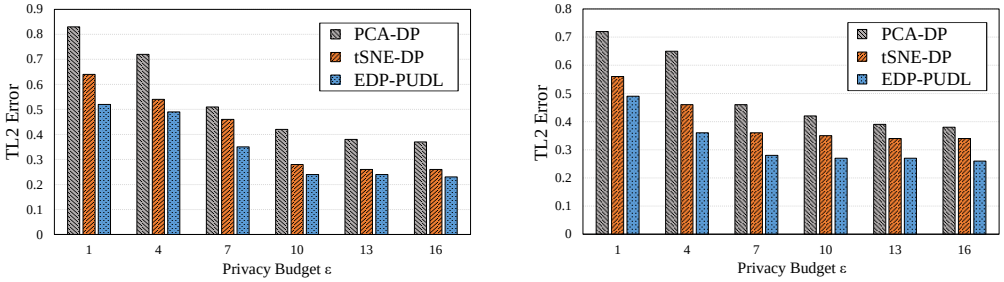

Figure 7. *TL2Error* with various privacy budgets on the ACS and NLTCS datasets.

Secondly, the comparison of *TAVD* and *TL2Error* was conducted on the Adult, TPC-E, ACS, and NLTCS datasets by varying the initial window size (with a total privacy budget $\epsilon = 7$) for PCA-DP, tSNE-DP, and EDP-PUDL. As depicted in Figure 8, EDP-PUDL exhibits lower *TAVD* values across all initial window sizes $|w|$ than the other methods on the Adult and TPC-E datasets. Specifically, on the Adult dataset, EDP-PUDL outperformed the other methods, with an average improvement of 18.2% and 9.4%. Furthermore, as shown in Figure 9, EDP-PUDL demonstrates lower values of *TL2Error* for all initial window sizes $|w|$ compared to the other three methods on the ACS and NLTCS datasets. On the ACS dataset, EDP-PUDL surpassed the other methods by 39.8% and 27.2%. These results indicate that when it comes to dataset reduction and noise processing tasks, EDP-PUDL yields less error and higher data availability than PCA-DP and tSNE-DP.

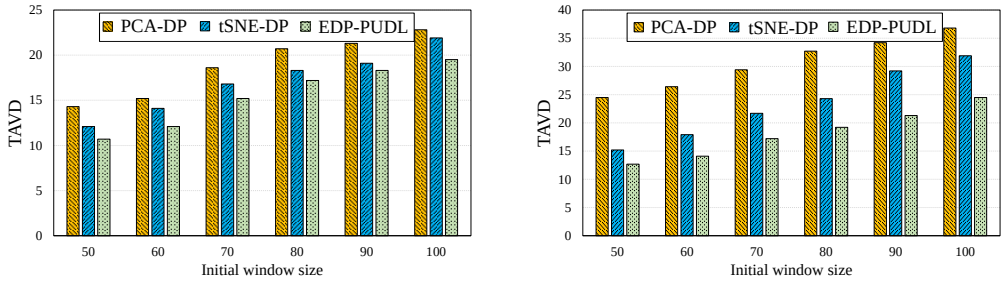

Figure 8. *TAVD* with various initial window sizes on the Adult and TPC-E datasets.

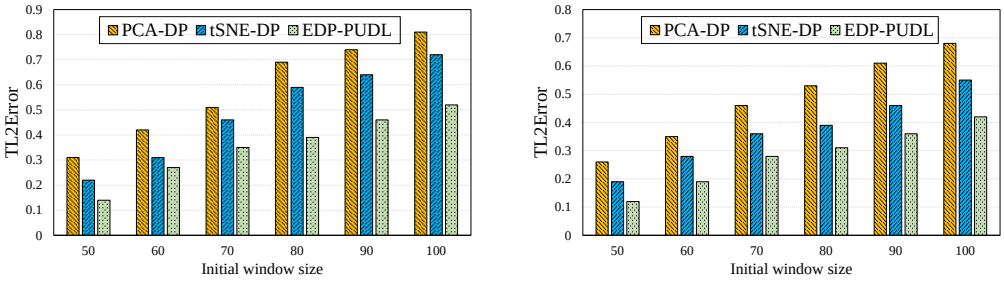

Figure 9. *TL2Error* with various initial window sizes on the ACS and NLTCS datasets.

According to Theorem 1, it can be observed that as the privacy budget increases, each method introduces a smaller amount of noise. Simultaneously, the L_2-Error, average variation distance, and SVM misclassification rates decrease until the noise becomes negligible and does not significantly impact the data.

5.6. Misclassification Rates

Firstly, the misclassification rates of PCA-DP, tSNE-DP, DPPro, PU-AveDP, and EDP-PUDL were compared on the Adult and TPC-E datasets by varying the size of the privacy budget. As shown in Figure 10, for all values of the privacy budget ϵ, EDP-PUDL exhibited

a lower classification rate than the other four methods on both the Adult and TPC-E datasets. Specifically for the Adult dataset, EDP-PUDL outperformed the other four comparison algorithms by average margins of 18.3%, 13.5%, 15.5%, and 11.5%. Moreover, in contrast to the PCA-DP, tSNE-DP, DPPro, and PU-AveDP aggregated datasets, EDP-PUDL lost less data utility while maintaining higher data availability.

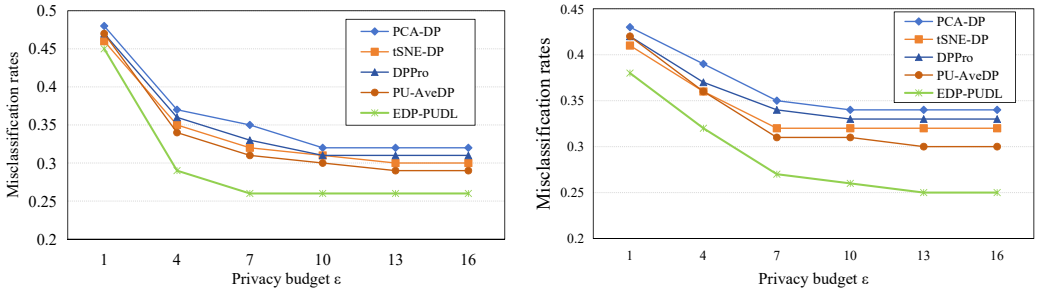

Figure 10. SVM misclassification rates on the Adult and TPC-E datasets.

The misclassification rates of PCA-DP, tSNE-DP, DPPro, PU-AveDP, and EDP-PUDL were compared on the ACS and NLTCS datasets by varying the initial window sizes. As shown in Figure 11, EDP-PUDL outperformed all other methods, with lower classification rates across all initial window sizes $|w|$ for both the ACS and NLTCS datasets. Specifically, on the ACS dataset, EDP-PUDL achieved a significantly lower misclassification rate than the other four methods, with an average improvement of 22.3%, 8.8%, 15.9%, and 11.8%. Compared with PCA-DP, tSNE-DP, DPPro, and PU-AveDP, EDP-PUDL lost less data utility and had higher data availability.

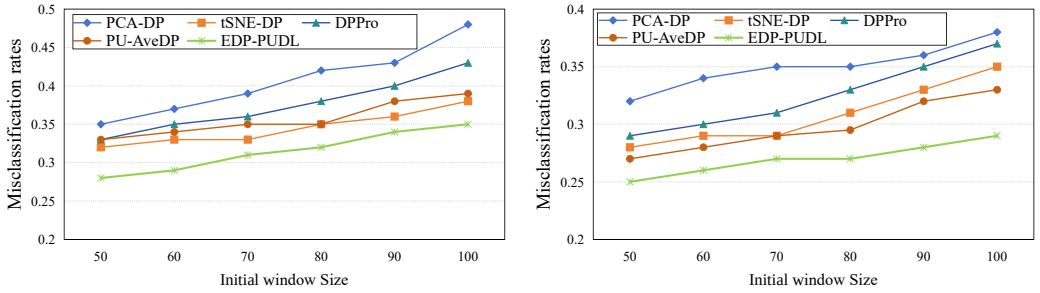

Figure 11. SVM misclassification rates on the ACS and NLTCS datasets.

5.7. Runtime

This subsection compares the running times of PCA-DP, tSNE-DP, DPPro, PU-AveDP, and EDP-PUDL on different datasets to evaluate efficiency. Dimensionality reduction was performed to obtain a target dimension of 3 ($l = 3$).

The running time of EDP-PUDL was shorter than those of PCA-DP, tSNE-DP, and DPPro on various datasets, as demonstrated in Table 4. PU-AveDP exhibited a shorter runtime than EDP-PUDL due to its average privacy budget allocation without considering the computation of the Shapley value; however, it incurred a higher availability loss for PU-AveDP.

Table 4. Runtime (s).

Method	Adult	TPC-E	ACS	NLTCS
PCA-DP	78.2	102.1	86.5	80.2
tSNE-DP	119.9	145.1	132.5	128.4
DPPro	86.2	118.5	92.2	89.9
PU-AveDP	68.0	98.3	78.3	71.0
EDP-PUDL	69.6	100.9	82.5	75.2

5.8. Discussions of Results

We first compared the variance with different window-dividing strategies to verify data privacy. Then, the TAVD, TL2Error, and misclassification rates were employed to evaluate the data availability of the methods based on the representative linear and nonlinear dimensionality reduction functions. At last, the efficiencies of all compared methods were assessed based on their runtime. As shown in the experimental results, our proposed EDP-PUDL outperformed the representative methods in variance, TAVD, TL2Error, and misclassification rates while exhibiting superior runtime performance.

Our proposed EDP-PUDL employs LSTM to predict the dynamics of data and adaptively adjusts the window size, thereby reducing the variance across different windows. Thus, it mitigates the risk of privacy leakage while utilizing DP. Progressive UMAP is superior in effectiveness and efficiency of dimensionality reduction for dynamic high-dimensional data, while the privacy budget allocation method proposed in this paper effectively mitigates data availability loss. Therefore, EDP-PUDL exhibits superior performance in terms of evaluation metrics.

In comparative experiments, we employed four public high-dimensional datasets containing sensitive personal information. We will apply our proposed method to other datasets that necessitate privacy preservation, such as facial image data used for face recognition, image data utilized in the detection of key industrial equipment, etc. Additionally, the efficiency of runtime can be further improved through adaptively adjusting the parameters of progressive UMAP by monitoring the data utility of each window, while ensuring data privacy and data availability.

6. Conclusions and Future Work

In this paper, we proposed an edge-based differential privacy data aggregation method, EDP-PIDL, for dynamic high-dimensional data aggregation in distributed applications. EDP-PUDL can be effectively employed to ensure data secure aggregation to provide analysis services in IoT applications, particularly when low latency is required. For example, in intelligent medical systems, it is important to securely and efficiently collect patient data from users for comprehensive analysis to facilitate accurate diagnosis and effective treatment. It is imperative to address the challenges posed by the privacy leakage and low efficiency resulting from data dynamics and high dimensionality. Thus, EDP-PUDL initially divides the data via a model of dynamic time windows based on LSTM and applies the progressive UMAP method to reduce the dimensionality of each window's data. To ensure the preservation of user privacy, DP noise is introduced before the user uploads data to the edge server. However, the application of DP noise and the dimensionality reduction technique may lead to a decrease in data availability. In order to mitigate the loss of data availability, the allocation of the privacy budget is determined based on the dataset size within each window and the attribute utility of its data. Through privacy analysis and experimental comparative analysis involving variance, total L2-Error, total average variable distance, misclassification rate, and runtime with related algorithms on dynamic high-dimensional datasets, we found that our proposed method exhibits superior performance in terms of data privacy, data availability, and operational efficiency. It provides an effective solution for secure data aggregation in distributed applications.

This paper focused on security aggregation services provided to users without considering users' personalized privacy requirements for data. However, certain users may possess individualized privacy preferences regarding the data they upload. Therefore, in future work, we aim to develop an efficient and secure solution that caters to the diverse privacy requirements of different users regarding data uploading.

Author Contributions: The authors confirm their contributions to this paper as follows: study conception and design: Q.C. and Z.N.; data collection: M.L., P.X. and W.L.; analysis and interpretation of results: Q.C., X.Z. and W.L.; draft manuscript preparation: Q.C., Z.N. and X.Z. All authors have read and agreed to the published version of this manuscript.

Funding: This work was supported by the National Nature Science Foundation of China under grant No. 91546108 and No. 71490725, the Anhui Provincial Science and Technology Major Projects under grant 201903a05020020, the Anhui Provincial Natural Science Foundation under grant No. 1908085QG298, the Fundamental Research Funds for the Central Universities under grant No. PA2023IISL0093, No. JZ2019HGTA0053, and No. JZ2019HGBZ0128, and the Open Research Fund Program of Key Laboratory of Process Optimization and Intelligent Decision-making, Ministry of Education.

Data Availability Statement: The data supporting our conclusions are included within this article. Any inquiries regarding the data can be directed to the corresponding author.

Acknowledgments: We thank the anonymous reviewers and journal editors for their invaluable assistance. Furthermore, we would like to acknowledge the support provided by the National Nature Science Foundation of China (NSFC), Department of Science and Technology of Anhui Province, and Hefei University of Technology.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding this study.

References

1. Sha, K.; Yang, T.A.; Wei, W.; Davari, S. A survey of edge computing-based designs for IoT security. *Digit. Commun. Netw.* **2020**, *6*, 195–202. [CrossRef]
2. Behrouz, P.; Nima, J.N. Data aggregation mechanisms in the Internet of things: A systematic review of the literature and recommendations for future research. *J. Netw. Comput. Appl.* **2017**, *97*, 23–34. [CrossRef]
3. Moon, J.; Hong, D.; Kim, J.; Kim, S.; Woo, S.; Choi, H.; Moon, C. Enhancing Autonomous Driving Robot Systems with Edge Computing and LDM Platforms. *Electronics* **2024**, *13*, 2740. [CrossRef]
4. Yousefi, S.; Karimipour, H.; Derakhshan, F. Data Aggregation Mechanisms on the Internet of Things: A Systematic Literature Review. *Internet Things* **2021**, *15*, 100427. [CrossRef]
5. Sabah Salih, H.; Jaber, M.M.; Ali, M.H.; Abd, S.K.; Alkhayyat, A.; Malik, R.Q. Application of edge computing-based information-centric networking in smart cities. *Comput. Commun.* **2023**, *211*, 46–58. [CrossRef]
6. Alwarafy, A.; Al-Thelaya, K.A.; Abdallah, M.; Schneider, J.; Hamdi, M. A Survey on Security and Privacy Issues in Edge-Computing-Assisted Internet of Things. *IEEE Internet Things J.* **2021**, *8*, 4004–4022. [CrossRef]
7. Zhang, J.; Chen, B.; Zhao, Y.; Cheng, X.; Hu, F. Data Security and Privacy-Preserving in Edge Computing Paradigm: Survey and Open Issues. *IEEE Access* **2018**, *6*, 18209–18237. [CrossRef]
8. Dwork, C.; McSherry, F.; Nissim, K.; Smith, A. Calibrating Noise to Sensitivity in Private Data Analysis. In Proceedings of the Third Conference on Theory of Cryptography, New York, NY, USA, 4–7 March 2006; pp. 265–284. [CrossRef]
9. Stephanie, V.; Chamikara, M.; Khalil, I.; Atiquzzaman, M. Privacy-preserving location data stream clustering on mobile edge computing and cloud. *Inf. Syst.* **2022**, *107*, 101728. [CrossRef]
10. Zhang, G.; Zhang, S.; Man, Z.; Cui, C.; Hu, W. Location Privacy Protection in Edge Computing: Co-Design of Differential Privacy and Offloading Mode. *Electronics* **2024**, *13*, 2668. [CrossRef]
11. Ren, X.; Yu, C.M.; Yu, W.; Yang, X.; Zhao, J.; Yang, S. DPCrowd: Privacy-Preserving and Communication-Efficient Decentralized Statistical Estimation for Real-Time Crowdsourced Data. *IEEE Internet Things J.* **2021**, *8*, 2775–2791. [CrossRef]
12. Zhang, H.; Li, K.; Huang, T.; Zhang, X.; Li, W.; Jin, Z.; Gao, F.; Gao, M. Publishing locally private high-dimensional synthetic data efficiently. *Inf. Sci.* **2023**, *633*, 343–356. [CrossRef]
13. Zhou, F.; Wu, Q.; Wu, P.; Xu, J.; Feng, D. Privacy-preserving and verifiable data aggregation for Internet of Vehicles. *Comput. Commun.* **2024**, *218*, 198–208. [CrossRef]
14. Liu, P.; Wu, D.; Shen, Z.; Wang, H.; Liu, K. Personalized trajectory privacy data publishing scheme based on differential privacy. *Internet Things* **2024**, *25*, 101074. [CrossRef]
15. Bozdal, M.; Ileri, K.; Ozkahraman, A. Comparative analysis of dimensionality reduction techniques for cybersecurity in the SWaT dataset. *J. Supercomput.* **2024**, *80*, 1059–1079. [CrossRef]
16. Xu, C.; Ren, J.; Zhang, Y.; Qin, Z.; Ren, K. DPPro: Differentially Private High-Dimensional Data Release via Random Projection. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 3081–3093. [CrossRef]
17. Li, W.; Zhang, X.; Li, X.; Cao, G.; Zhang, Q. PPDP-PCAO: An Efficient High-Dimensional Data Releasing Method With Differential Privacy Protection. *IEEE Access* **2019**, *7*, 176429–176437. [CrossRef]
18. Chaudhuri, K.; Sarwate, A.D.; Sinha, K. A Near-Optimal Algorithm for Differentially-Private Principal Components. *J. Mach. Learn. Res.* **2013**, *14*, 2905–2943. [CrossRef]
19. Wang, S.; Chang, J.M. Differentially Private Principal Component Analysis Over Horizontally Partitioned Data. In Proceedings of the IEEE Conference on Dependable and Secure Computing, Kaohsiung, Taiwan, 10–13 December 2018; pp. 1–8. [CrossRef]
20. Chanyaswad, T.; Liu, C.; Mittal, P. RON-Gauss: Enhancing Utility in Non-Interactive Private Data Release. *Proc. Priv. Enhancing Technol.* **2019**, *2019*, 26–46. [CrossRef]

21. Law, M.H.C.; Jain, A.K. Incremental nonlinear dimensionality reduction by manifold learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 377–391. [CrossRef]
22. McInnes, L.; Healy, J.; Saul, N.; Großberger, L. UMAP: Uniform Manifold Approximation and Projection. *J. Open Source Softw.* **2018**, *3*, 861. [CrossRef]
23. Ko, H.K.; Jo, J.; Seo, J. Progressive Uniform Manifold Approximation and Projection. In Proceedings of the Eurographics Conference on Visualization, Norrköping, Sweden, 25–29 May 2020; pp. 133–137. [CrossRef]
24. Wu, N.; Bao, X.; Wang, D.; Jiang, S.; Zhang, M.; Zou, J. Task Offloading in Real-Time Distributed Energy Power Systems. *Electronics* **2024**, *13*, 2747. [CrossRef]
25. Almeida, A.; Brás, S.; Sargento, S.; Pinto, F.C. Time series big data: A survey on data stream frameworks, analysis and algorithms. *J. Big Data* **2023**, *10*, 83. [CrossRef]
26. Chen, Q.; Ni, Z.; Zhu, X.; Xia, P. Differential privacy histogram publishing method based on dynamic sliding window. *Front. Comput. Sci.* **2023**, *17*, 174809. [CrossRef]
27. Yan, X.; Zeng, B.; Zhang, X. Privacy-Preserving and Customization-Supported Data Aggregation in Mobile Crowdsensing. *IEEE Internet Things J.* **2022**, *9*, 19868–19880. [CrossRef]
28. Regan, R.; Josphineleela, R.; Khamruddin, M.; Vijay, R. Balancing data privacy and sharing in IIoT: Introducing the GFL-LFF aggregation algorithm. *Comput. Netw.* **2024**, *247*, 110401. [CrossRef]
29. Liu, Y.; Wang, H.; Peng, M.; Guan, J.; Xu, J.; Wang, Y. DeePGA: A Privacy-Preserving Data Aggregation Game in Crowdsensing via Deep Reinforcement Learning. *IEEE Internet Things J.* **2020**, *7*, 4113–4127. [CrossRef]
30. Tian, X.; Song, Q.; Tian, F. Multidimensional Data Aggregation Scheme for Smart Grid with Differential Privacy. *Int. J. Netw. Secur.* **2018**, *20*, 1137–1148. [CrossRef]
31. Tang, W.; Ren, J.; Deng, K.; Zhang, Y. Secure Data Aggregation of Lightweight E-Healthcare IoT Devices With Fair Incentives. *IEEE Internet Things J.* **2019**, *6*, 8714–8726. [CrossRef]
32. Wang, T.; Mei, Y.; Jia, W.; Zheng, X.; Xie, M. Edge-based differential privacy computing for sensor–cloud systems. *J. Parallel Distrib. Comput.* **2020**, *136*, 75–85. [CrossRef]
33. Shang, S.; Li, X.; Gu, K.; Li, L.; Zhang, X.; Pandi, V. A Robust Privacy-Preserving Data Aggregation Scheme for Edge-Supported IIoT. *IEEE Trans. Ind. Inform.* **2024**, *20*, 4305–4316. [CrossRef]
34. Lyu, L.; Nandakumar, K.; Rubinstein, B.; Jin, J.; Bedo, J.; Palaniswami, M. PPFA: Privacy Preserving Fog-Enabled Aggregation in Smart Grid. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3733–3744. [CrossRef]
35. Imtiaz, S.; Horchidan, S.F.; Abbas, Z.; Arsalan, M.; Chaudhry, H.N.; Vlassov, V. Privacy Preserving Time-Series Forecasting of User Health Data Streams. In Proceedings of the IEEE International Conference on Big Data, Atlanta, GA, USA, 10–13 December 2020; pp. 3428–3437. [CrossRef]
36. Dwork, C.; Kenthapadi, K.; McSherry, F.; Mironov, I.; Naor, M. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, 28 May–1 June 2006; pp. 486–503. [CrossRef]
37. Zhu, T.; Li, G.; Zhou, W.; Yu, P.S. Preliminary of Differential Privacy. In *Differential Privacy and Applications*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 7–16. [CrossRef]
38. Zhu, T.; Li, G.; Zhou, W.; Yu, P.S. Differentially Private Data Publishing and Analysis: A Survey. *IEEE Trans. Knowl. Data Eng.* **2017**, *29*, 1619–1638. [CrossRef]
39. McSherry, F.D. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, Providence, RI, USA, 29 June–2 July 2009; pp. 19–30. [CrossRef]
40. Liu, J.; Lou, J.; Liu, J.; Xiong, L.; Pei, J.; Sun, J. Dealer: An end-to-end model marketplace with differential privacy. *Proc. VLDB Endow.* **2021**, *14*, 957–969. [CrossRef]
41. Gough, M.B.; Santos, S.F.; AlSkaif, T.; Javadi, M.S.; Castro, R.; Catalão, J.P.S. Preserving Privacy of Smart Meter Data in a Smart Grid Environment. *IEEE Trans. Ind. Inform.* **2022**, *18*, 707–718. [CrossRef]
42. Zhang, Z.; Wang, H.; Xue, W. Approach for data streams clustering over dynamic sliding windows. *Comput. Eng. Appl.* **2011**, *47*, 135–138.
43. Shlens, J. A Tutorial on Principal Component Analysis. *Int. J. Remote Sens.* **2014**, *51*, 1–12. [CrossRef]
44. Cai, T.T.; Ma, R. Theoretical Foundations of t-SNE for Visualizing High-Dimensional Clustered Data. *J. Mach. Learn. Res.* **2022**, *23*, 1–54. [CrossRef]