Journal of Theoretical and Applied Information Technology
15th April 2026. Vol.104. No.7

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PRIVACY PRESERVING HIGH-DIMENSIONAL DISTRIBUTED LEARNING VIA CLIENT-SIDE DEEP DENOISING SPARSE AUTOENCODERS AND SPLIT LEARNING

MR. LINGAM SUMAN[1], DR. S. VENKATA LAKSHMI[2]

[1] Research Scholar, Dept. of CSE, GITAM (Deemed to be University), Visakhapatnam, Andhra Pradesh-530045, India

[2] Assistant Professor, Dept. of CSE, GITAM (Deemed to be University), Visakhapatnam, Andhra Pradesh-530045, India

E-mail: [1]ssuman2468@gmail.com, [2]svlakshmi2014@gmail.com

ABSTRACT

Privacy preservation (PP) in deep learning (DL) is becoming increasingly important, particularly when working with high-dimensional datasets that pose considerable computing hurdles. Traditional approaches frequently experience the "curse of dimensionality," resulting in inefficiencies and increased privacy threats. To address these issues, this study offers an innovative process that combines Split Learning (SL) with Client-side Deep Denoising Sparse Autoencoders (CDDSAEs), resulting in a more efficient and privacy-conscious solution. The technique begins with attribute filtering, which uses a threshold-based mechanism to keep just the most important data properties, hence lowering dimensionality from the start. The data is then vertically shard to distribute computational burdens and let each client to focus on a single data shard. In the SL architecture, DDSAEs are used on the client side to compress and anonymized data before transmission, guaranteeing that only low-dimensional, PP representations are exchanged with the server. Performance investigation shows that the suggested model reduces latency by 17.7%, runs faster by 40.3%, and has a minimum information loss of 0.22, demonstrating its superiority over existing techniques.

Keywords: *Curse of Dimensionality, Split Learning, Attribute Filtering, Vertical Sharding, Denoise.*

1. INTRODUCTION

In today's digital era, the proliferation of data, driven by the extensive use of Internet-of-Things (IoT) and cloud-based smart devices, has surged significantly. Users now generate and exchange vast quantities of data, which often include sensitive information about individuals. As per a recent survey reported by the Storage Newsletter in April 2017, the global data volume was estimated at 16.1 zettabytes (10^{21} bytes) in 2016, and is projected to surge to 163 zettabytes by 2025 [1]. Consequently, safeguarding privacy has become a critical issue. This is true, for instance, for patient records that are collected at various hospitals but are generally not shared with other hospitals or facilities because of the sensitive information they contain [2-4]. There are many PP data publication models that have been proposed to enable data sharing while at the same time avoiding leakage of secrets. High dimensions as characteristic of the information age bring multiple advantages but come with practical difficulties, especially in the dimensionality of data. To deal with this, dimensionality reduction is generally done through either feature extraction or selection. Feature selection is the process where by a set of original features that are relevant to the data is selected, while feature extraction is a process where by a new set of features are created from the original features, and this is carried out in order to reduce the dimensions of the data. Feature selection, which is used when the aim is to keep the physical meaning of the data as close as possible to the original data, is popular in such areas as bioinformatics and image processing. It is especially true in noisy datasets where the learning speed and accuracy can be enhanced through feature reduction which leads to an improvement in the classification results [5]. Nevertheless, the problem of publishing high-dimensional data with high PP remains a challenging issue, and the development of efficient PP schemes for high dimensional datasets has been paid relatively less attention [2,6].

To address these issues, the present study introduces a new model integrating SL into

CDDSAEs for improving not only PP but also computation efficiency. The suggested approach involves an attribute filtering mechanism that helps in the elimination of all the unnecessary attributes and hence reduces dimensionality at the beginning. This filtered data is then vertically shard, which makes the computation manageable within the various client devices. In the SL framework, DDSAEs are implemented at the client side where they perform data compression and noise reduction before sending the data to the server. This approach ensures that only low-dimensional representations are shared, such that privacy of the individuals is protected from leakage. These improvements make the model ready to be implemented in real-world applications that require minimum data disclosure and shorter computational efficiency. To obtain a more effective and PP DL solution for high-dimensional datasets, the following contributions were made:

• To obtain a more efficient distribution of computational resources, a vertical sharding technique was employed, dividing the dataset into smaller, attribute-based shards, which allowed client devices to process only specific portions of the data, reducing overall complexity and latency.

• To achieve a PP transmission of data, the suggested model only sends low-dimensional, denoised representations from the client-side to the receiver, thus the receiver cannot obtain any details of the user.

• To obtain improved overall system performance, the SL framework was utilized, allowing the training process to occur in a distributed fashion while maintaining strict privacy controls, leading to a reduction in latency by 17.7% and a decrease in running time by 40.3% compared to existing methods.

• To achieve the first level of dimensionality reduction, threshold based attribute filtering mechanism was incorporated so that the most important attributes are selected from the onset to reduce the amount of computation from the outset.

• To ensure secure data handling in distributed frameworks, the model's client-side denoising and compression methods limit the amount of data exposure during the learning process, making it highly applicable in scenarios where data privacy and integrity are paramount, such as healthcare, finance, and smart cities.

The work is organized as follows: the initial section introduces the challenges of handling high-dimensional data and the computational demands associated with PP DL. The next section reviews existing methods and identifies gaps in current approaches. Section 3 details the proposed model, including its integration of client-side DDSAEs and SL. The results and their implications are discussed in Section 4. The work concludes with Section 5, summarizing the findings and suggesting directions for future research.

2. RELATED WORK

Zhao et al. [7] introduced a novel FL framework known as Optimal LDP-FL. It was demonstrated that this framework adhered to ε-LDP, effectively mitigating privacy breaches associated with server-side client tracking over parameter shuffling. A limiting self-sampling likelihood was presented, which decreased variation in self-sampling likelihood sets found in prior research while increasing efficiency.

Ha et al. [8] presented an advanced learning technique that trained a DNN while safeguarding the original raw data. In contrast to conventional DL, where a single DNN is trained solely by the client, SL distributed the training process among the clients and a centralized server. The server was not required to access the training data directly, instead, it trained the DNN using parameters obtained from the PP layer, thereby ensuring the confidentiality of the original data.

Table 1: Existing research work

Work	Method	Advantage	Limitation
Jiang et al. [9]	Leveraging Federated Generative AE to address the challenges of high dimensionality.	The accuracy loss is dramatically decreased to no more than 3%, and at most effectively, below 1%.	Struggle with utility loss and require better communication and computational effectiveness.
Amin et al. [6]	An innovative PP methodology for anonymized high-dimensional data.	Safeguard the data from attackers while avoiding information loss.	Challenge in handling high dimensional data
Chen et al. [10]	A density peak clustering with differential PP.	Successfully mitigate the loss of clustering quality.	The challenge is that handling high-dimensional data is still difficult.
Shi et al. [11]	PDP Growth under the Spark framework	Increase the algorithm's operational effectiveness and	Inefficient utilization of computational resources in distributed

		data accessibility.	frameworks can hinder algorithm efficiency.

Ratra et al. [1] suggested a PP strategy leveraging dimensionality decrease and feature selection techniques that are challenging to reverse. The aim was to develop a perturbation-based PP method. In this approach, random projection and PCA were employed to modify the data. This hybrid method selected pertinent features, reduced the data's dimensionality, and shortened training time, resulting in an increase in accuracy from 63.13% to 68.34%.

Shen et al. [12] introduced LoHDP, a high-dimensional data publishing technique integrating adaptive marginal computing and an efficient attribute clustering technique. The attribute clustering approach employed an effective technique to measure pairwise attribute correlations, reduced the search space for dependency graph construction through high-pass filtering, and achieved dimensionality lessening by incorporating comprehensive triangulation operations.

Mishra et al. [13] identified a thorough set of PP attributes for Cloud data storage and proposed a flexible and efficient framework to address the privacy challenges. This framework employed a multi-layer encryption storage structure coupled with a one-time password authentication technique. Alshammari and El Hindi [14] proposed and evaluated a PP DL framework that integrated Instance Lessening Methods with the Restricted Boltzmann Machine to enhance privacy in collaborative DL environments and to address the limitations of existing PP solutions.

Existing approaches to PP and high-dimensional data handling face significant challenges as shown in Table 1. Jiang et al. [9] demonstrated that while Federated Generative AEs can reduce data dimensionality, they critically suffer from utility loss of up to 3% and impose heavy communication and computational overheads that scale poorly with data dimensionality, making them unsuitable for real-time or resource-constrained distributed environments. Chen et al. [10] explored density peak clustering for PP but found that high-dimensional data significantly degrades clustering quality, as the algorithm's reliance on pairwise distance metrics becomes increasingly unreliable in high-dimensional spaces due to the concentration of distance phenomenon. Shi et al. [11] presented PDP Growth under the Spark framework; however, this method is fundamentally limited by its inefficient utilization of computational resources in distributed settings, as it does not integrate adaptive load balancing, thereby constraining algorithm efficiency when computational resource allocation varies. Amin et al. [6] proposed L-diverse constrained slicing for privacy-preserving high-dimensional data, but the technique lacks scalability, as maintaining L-diversity constraints across a growing number of attributes introduces exponential complexity that hinders deployment in large-scale distributed systems. A critical unresolved issue in high-dimensional data publishing is the "curse of dimensionality," which introduces significant noise and reduces the utility of published data. Additionally, as data dimensions increase, so does the correlation between attributes, making differential privacy defences particularly ineffective, and traditional noise-adding methods risk damaging attribute correlations [11]. None of the reviewed approaches simultaneously address dimensionality reduction, distributed computation efficiency, and privacy-preservation within a unified client-side framework. The proposed model directly addresses these gaps by employing client-side DDSAE encoders to locally pre-process and reduce data dimensionality through vertical sharding, thereby minimizing resource usage, computational demands, and privacy exposure across all nodes in a distributed framework.

3. METHODOLOGY

The proposed solution is structured into three distinct phases. The first phase involves attribute filtering, where only the most relevant features are retained. Next, the raw data is segmented into smaller subsets through vertical sharding. In the final phase, the SL approach is employed, where DDSAEs are utilized on the client side to locally compress and remove noise, ensuring that only low-dimensional, PP representations are sent to the server. A comprehensive overview of the proposed solution is outlined in Figure 1.

Dataset: The dataset [15] consists of 21,799 clinical 12-lead ECG recordings, each lasting 10 seconds, collected from 18,869 patients. The patient cohort includes 52% males and 48% females, spanning an age range from 0 to 95 years, with a median age of 62 and an interquartile range of 22. The dataset's significance lies not only in its extensive representation of various co-occurring pathologies but also in its substantial number of healthy control samples.

Phase 1: Filtering Attributes

In the proposed model, the attribute filtering phase begins by determining a threshold value, ThI to evaluate and filter the attributes based on their importance. This step aims to remove redundant or irrelevant attributes and those with missing values from the original dataset. The importance of each attribute is assessed using information entropy, which serves as a measurement index to calculate the information attribute I. Attributes are filtered [16] by setting the threshold ThI.

To quantify the importance of each attribute, the entropy value, denoted as E, is used. A higher E indicates greater attribute importance AI and a higher I. Conversely, a lower E signifies less attribute importance and a lower I. During the dimension reduction process, attributes with higher AI are retained, while those with lower I are discarded. ThI acts as a threshold to decide which attributes should be kept. For each attribute d_x in a high-dimensional dataset D = {d_1, d_2, … d_x}, E is determined using the following formula:

$$E(d_x) = -\sum p_\gamma \log p_\gamma \quad (1)$$

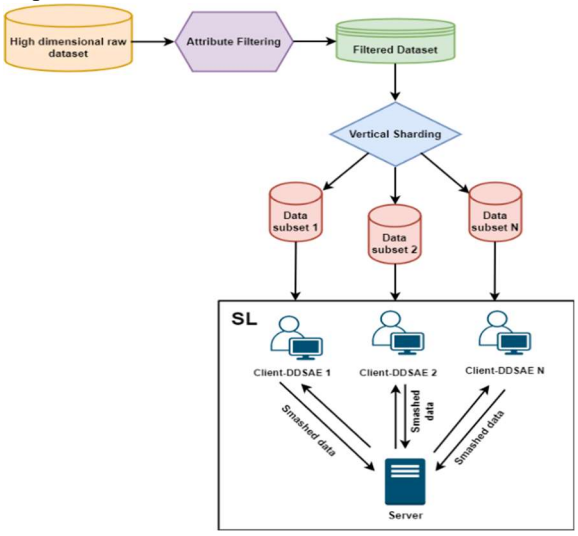
Where n represents the number of distinct states within the attribute d_x. Attributes are then ranked based on their I values, and ThI is used to filter them. The proportion of the information entropy of the selected attributes in the original dataset determines ThI, calculated as:

$$\sum E(x) / \sum E(y) \geq Th \quad (2)$$

Where m is the total number of attributes, and b is the number of attributes selected. An attribute is retained if its I exceeds ThI, meaning its importance is above the predefined threshold. Conversely, if an attribute's I is below ThI, it is eliminated from the dataset. This selective retention ensures the reduction of data dimensionality while preserving the most informative attributes, thereby optimizing the dataset for subsequent processing steps.

Phase 2: Vertical Sharding

Following the attribute filtering phase, vertical sharding was applied to manage the high-dimensional data effectively. In proposed model, the challenge of high-dimensionality is addressed through a sophisticated sharding strategy. Previous research indicates that generalization techniques often lead to significant information loss, particularly with high-dimensional datasets. To overcome this problem, divided the original dataset into disjoint shards. This was made possible through partitioning the raw data into different shard tables, where each, retained an equal portion of attributes required for the classification analysis. This partitioning metric was intended to achieve the objectives of high privacy as well as retention of important characteristics of the data.



The sharding process involved grouping similar attributes [6], for example, quasi-identifiers (QID) were put in one shard and sensitive attributes (SAs) in the other. This arrangement ensures that the ability of attributes in making predictions for classification is retained while at the same time reducing the chances of data leakage. At first the data was split according to gender, for example (t_m, t_f) male and female. These tables were further shard based on the similarity and connection among QID and SAs. Non identifier attributes were considered as QID attributes in order to increase the usefulness of the dataset for classification. One of the important characteristics of this phase was the utilization of a factor X in order to regulate the share of attributes distributed between the separate shards. X was normalized to a value between 0 and 1 in order to fairly divide the attributes among the shards. Thus, to balance the QID characteristic across pieces, it is possible to limit the value of Y only to the given range and use it for further calculations concerning X. To compute the optimal assignment of an attribute d_x to a shard s_γ, used the following formula:

$$(d_x, s_\gamma) = (1 - P_\gamma) \times (C_x - Avg_{x\gamma}) \quad (3)$$

$$Y = (1 - P_\gamma) \quad (4)$$

$$X = (C_x - Avg_{x\gamma}) \quad (5)$$

$$P_\gamma = n_\gamma / n\_ass \quad (6)$$

$$C_x = I(d_x, cls) \quad (7)$$

$$Avg_{x\gamma} = (1/|s_\gamma|) \times \sum I(d_x, d) \quad (8)$$

Here n_γ is the number of attributes in s_γ, n_ass is the total number of attributes assigned. $|s_\gamma|$ is the number of attributes in s_γ. At the beginning of this partitioning process, class attributes were assigned to each shard. Two attributes with the maximum correlation values were nominated as initial seeds and placed into different shards, such as s_1 and s_2. The sharding continued iteratively, as outlined in Algorithm 1, with each subsequent iteration optimizing the distribution of attributes between shards. This process effectively created multiple subsets, achieving a balance between data privacy and utility.

Algorithm 1: Sharding of high dimension raw dataset
Input: Raw high dimensional dataset with attributes $D = \{d_1, d_2, \dots d_x\}$
Output: Optimally shard subsets $s_1, s_2, \dots s_n$
Start:
Step 1: Initialize $s_1 \leftarrow$ Male data, $s_2 \leftarrow$ Female data
Step 2: For each cls in D
  Assign $s_1 \leftarrow$ Male data, $s_2 \leftarrow$ Female data
  Step 3: Set initial seeds
    $s_1 \leftarrow s_1 \cup (Seed_1)$, $s_2 \leftarrow s_2 \cup (Seed_2)$
Step 4: While $s \neq 0$
Step 5: For each attribute d_x in (s_1, s_2)
Step 6: Compute current score (d_x, s_γ)
Step 7: If current score > max score
Step 8: Assign $d' \leftarrow d_x$
Step 9: $s' \leftarrow s_\gamma$
Step 10: $s \leftarrow s_1, s_2$
Step 11: Repeat steps until all attributes are assigned.

Figure 1: Overall Architecture of Client-DDSAE

Phase 3: CDDSAEs and SL

The next step in the proposed model is to implement SL with DDSAEs for efficient handling of high-dimensional datasets while PP [13-16]. In this phase, each client uses a DDSAE to process its locally held data. Each DDSAE model is specifically trained on its respective data shard obtained from the vertical sharding phase, which ensures that each client only handles data relevant to its domain, further enhancing privacy. The SL framework divides the AI model into two sub-models: a client and a server-side model. The client-side model, now equipped with DDSAE, takes raw, high-dimensional data as input and outputs a compact, denoised, and privacy-preserving latent representation of the data. This compressed representation, known as "smashed data," is then transmitted to the server-side model for further processing.

Client-Side Operations:

The DDSAE is implemented on the client side to handle high-dimensional data. The encoder part of the DDSAE, operating with a denoising mechanism, takes raw input data d by minimizing the reconstruction error. The encoder maps the high-dimensional input d to a lower-dimensional latent representation l using the following non-linear transformation:

$$l = e(w\tilde{d} + b) \quad (9)$$

Where e is the activation function, w is the weight matrix, and b is the bias vector. To improve the robustness against adversarial perturbations and noise, the input data is first corrupted to create $\tilde{d}$ a noisy version of the original data d. The DDSAE encoder function, therefore, takes this noisy input $\tilde{d}$ and generates a latent representation that is resilient to perturbations, thereby reducing data dimensionality. To ensure the sparsity of the hidden layer activations, the KL divergence is used as a regularizer term in the AE's loss function. The regularization term is defined as:

$$KL(q||\hat{q}) = q\ log(q/\hat{q}) + (1-q)\ log((1-q)/(1-\hat{q})) \quad (10)$$

Here q is the desired activation value, and $\hat{q}$ is the average activation of the neurons in the hidden layer. By incorporating this regularization, the DDSAE encourages the majority of neurons to remain inactive, thus only allowing the most relevant features to be extracted. The DDSAE encoder introduces a dropout mechanism to prevent overfitting by randomly setting the activations of neurons to zero with a certain probability. This technique, combined with noise addition at the input layer, ensures that the encoder is robust against adversarial perturbations while maintaining privacy.

Server-Side Operations:

The server accepts the encrypted feature map from multiple clients, which has been processed through the DDSAE on the client-side. The server model continues the remaining steps of the neural network training, including forward propagation, loss computation, and backpropagation. The server's operations do not involve any access to the raw data, ensuring data privacy and compliance with regulatory standards. The server receives this "smashed data" l and performs further computations on its sub-model f_s to make following predictions and the loss function:

$$f(d) = (f_s \circ l)d \quad (11)$$

$$L(f_s(l(d)), y) \quad (12)$$

Here y is the ground truth label. The server updates its parameters w_s according to:

$$w_s \leftarrow w_s - \eta \cdot \partial L/\partial w_s \quad (13)$$

Here η is the learning rate. During the backward pass, the server calculates the gradients for its model parameters ∂L/∂l(d) and transmits the necessary gradients back to the client-side model. The client-side DDSAE then updates its parameters using these gradients, further refining its feature extraction capabilities while maintaining the privacy of the raw data.

$$wc \leftarrow wc - \eta \cdot \partial L/\partial wc \quad (14)$$

After the client completes the local training step, it shares its updated model parameters with the next client in line. This process continues iteratively across all participating clients, enabling distributed and privacy-preserving model training without exposing the raw data to the server or other clients.

4. RESULTS AND DISCUSSION

Experiments are executed in Python 3.9 using TensorFlow 2.10 and NumPy 1.23 on a PC with an Intel i3-8100 CPU and 16 GB of RAM. The full research protocol applied to obtain the results is described as follows. The PTB-XL ECG dataset [15] was used for all experiments, comprising 21,799 12-lead ECG recordings (10 seconds each) from 18,869 patients (52% male, 48% female; age range 0–95 years, median age 62). The dataset was split into 70% training, 15% validation, and 15% test sets. The attribute filtering threshold Th_I was set to 0.75 (retaining attributes whose cumulative entropy proportion exceeds 75% of the total dataset entropy). Vertical sharding divided the filtered dataset into two shards based on gender (male/female) with further sub-sharding by quasi-identifier (QID) and sensitive attribute (SA) groupings. The DDSAE on each client was configured with an input layer matching the shard dimensionality, two encoding layers (256 and 128 neurons), a latent layer of 64 neurons, and symmetric decoding layers. Dropout was applied at a rate of 0.3, and Gaussian noise with standard deviation 0.1 was added at the input for denoising pretraining. The KL divergence sparsity regularizer was applied with a target activation of q = 0.05. Training used the Adam optimizer with a learning rate of 0.001, batch size of 64, and 50 epochs. The split learning cut layer was set after the latent encoding layer. Baseline models Peak Clustering [10] and PDP Growth [11] were reimplemented using their published algorithm descriptions and evaluated on the same dataset and hardware. Four evaluation metrics were used: (1) Latency: total end-to-end inference time in milliseconds across varying computational resource counts; (2) Running Time: total training and inference time in seconds across varying feature counts; (3) Computational Complexity: processing time in milliseconds as a function of record count; and (4) Information Loss: average reconstruction error computed as the normalized mean squared difference between original and reconstructed data representations.

From Figure 2, the comparison of latency values at various levels of allocated computational resources for the Peak Clustering [10] and the PDP Growth [11] and the proposed CDDSAE model reveal decreased total latency potential when applying SL. With 25 computing resources, the suggested CDDSAE has a significant decrease of 17.7% in contrast to Peak Clustering and 13.6% compared to PDP Growth. Since the proposed CDDSAE model has a lowest latency consistently, it means that SL framework is utilizing the limited computational resources more efficiently. This model makes use of shallow data decomposition and processing in the lower layer and this reduces the amount of computation required during forward and backward propagation phases thus making the entire operation smoother and less time-consuming.

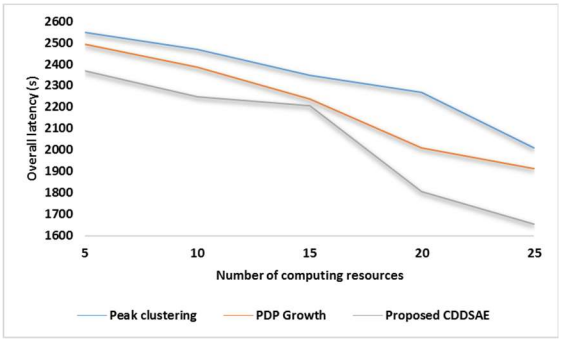


Figure 2: Latency Comparison

On the same note, the PDP Growth method is endowed with high latency as is witnessed by the manner in which it consumes a lot of computational power with the distributed frameworks. This inefficiency arises from no proper integration of resources and the inability to adjust for distinctive resource situations, which poses a restraint on the general algorithmic efficiency and adds to the amount of computation, particularly when the amount of computing resources assigned rises.

Figure 3 depicts a cost study over several epochs for Peak Clustering, PDP Growth, and the suggested CDDSAE model, which shows a constant cost decrease. At the initial epoch 0, the suggested CDDSAE model has a cost of 0.01443, which is roughly 5.9% lower than Peak Clustering (0.01534) and 5.5% lower than PDP Growth (0.01527). As training advances, the suggested methodology continues to show considerable cost reductions. By epoch 8, the suggested model has the lowest cost of 0.00181, which is 39.9% lower than Peak Clustering (0.00301) and 13.8% lower than PDP Growth (0.0021).

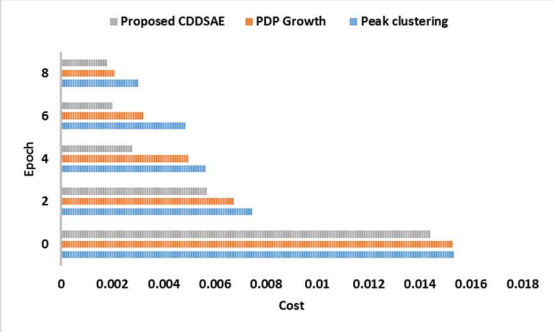


Figure 3: Cost Analysis

The enhanced performance of the proposed CDDSAE model in terms of cost reduction is explained by the capability of SL combined with DDSAEs. This methodology aims at using the least ratios between features and dimensions by employing features to ensure that the model does not process large amounts of cluttered and redundant data. Therefore, the proposed model allows an improved learning process, a lower cost per epoch and faster towards an optimal solution.

Alternatively, the Peak Clustering method has challenges experiencing in high-dimensional data since this technique is difficult to solve. The convergence becomes slower and the computational costs bigger as the data become more complex due to the increasing size of the network in the next iterations. Moreover, the PDP Growth method is more efficient than the Peak Clustering, but it does not involve the improved optimization strategies used in the CDDSAE, so the costs are high during the training phase comparatively.

Figure 4 shows a comparison of running time across different numbers of characteristics for Peak Clustering, PDP Growth, and the suggested CDDSAE model, demonstrating the CDDSAE model's higher efficiency in terms of computing time minimisation. As the number of characteristics rises, the suggested model's efficiency remains constant. For ten characteristics, the suggested CDDSAE model reduced running time by 32.1% compared to Peak Clustering and 13.6% compared to PDP Growth. The tendency persists as the number of qualities increases. For 30 characteristics, the CDDSAE model's running time is 40.3% lower than Peak Clustering and 12.2% lower than PDP Growth.

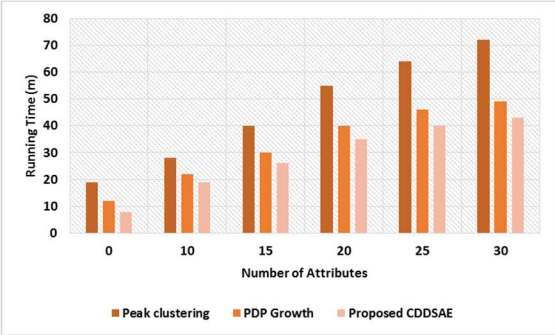


Figure 4: Running Time

The time taken to run the models have been significantly reduced by the proposed CDDSAE model as a result of the design that integrates SL to balance the load of computation between the devices. This approach reduces the amount of data transmitted and computed at each of the steps as it enhances the speed of processing data. Furthermore, the integration of DDSAEs in the proposed model ensures that the likelihood of the data is represented in a low-dimensional space hence cutting down on the density of architecture subsequently lessening on the time taken by the model.

However, as can be seen from the Peak Clustering, where the number of attributes increases, it shows a serious drawback in controlling the running time. This is due to the fact that it has a high computational complexity and takes a lot of time in processing data especially when the data has many features. Even though PDP Growth has been established to outperform Peak Clustering in terms of efficiency, the results reveal that it has higher running times since the utilization of the computational resources in the distributed frameworks is not optimal.

From the Figure 5 computational difficulty of Peak Clustering, Federated Generative AE [1], and the proposed CDDSAE model varies with record size, highlighting the CDDSAE approach's lower computing requirement. At a record size of 100, the suggested CDDSAE model has a computational

complexity of 26 ms, which is 23.5% lower than Peak Clustering (34 ms) and 7.1% lower than Federated Generative AE (28 ms). At 1000 records, the CDDSAE model has a complexity of 45 ms, which is 16.7% lower than Peak Clustering (54 ms) and 6.3% lower than Federated Generative AE (48 ms).

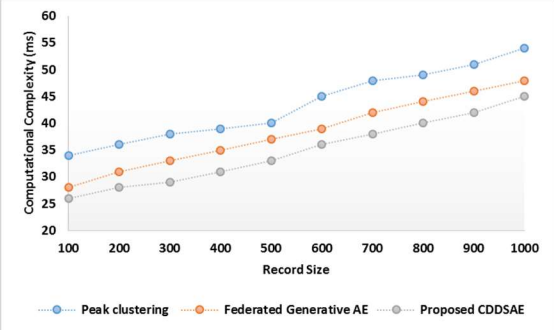


Figure 5: Computational Complexity

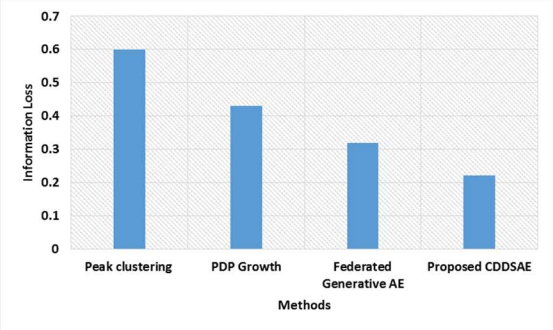


Figure 6: Information Loss

The positive impact of the applied vertical sharding towards the decreased computational complexity of the proposed CDDSAE model has been highlighted in this study. This approach is based on the conceptual division of the data in various manners or ways that will help in reducing the repetitious data in various subsets and hence will help in reducing the general computational cost. If the work of computation is distributed across smaller segments of data management and in a more manageable way the model can handle the computations at a faster rate than before while at the same time preserving the data utility and data privacy.

By contrast, some issues arise for the Federated Generative AE model, mostly due to the fact it cannot successfully provide solutions to the high-dimensional data, as well as utility loss spikes and the necessity for employing better communication/computational protocols. Even though Federated Generative AE is capable of spreading learning across multiple nodes, it typically involves a high measure of communication complexity and, because of federated averaging, may lead to decreased data utility when tackling high-dimensional datasets.

The analysis of information loss (see Figure 6) across Peak Clustering, PDP Growth, Federated Generative AE, and the suggested CDDSAE framework demonstrates that the CDDSAE is preferable in terms of data utility preservation. The suggested CDDSAE framework has a minimal information loss of 0.22, which is much lower than Peak Clustering (0.6), PDP Growth (0.43), and Federated Generative AE (0.32). Particularly, the CDDSAE framework reduces information loss by 63.3% in contrast to Peak Clustering, 48.8% comparing to PDP Growth, and 31.3% comparable to Federated Generative AE.

This significant reduction in information loss can be attributed to the fact that the CDDSAE model applied vertical sharding to allow more efficient partitioning of high dimensionality data and retain significant information across the different dimensions. Unlike the traditional models where data may be processed in batch, vertical sharding can be used selectively and can reduce the amount of data redundancy and thus retain the inherent value of data while at the same time achieving the intended privacy goals.

On the other hand, the highest information loss is observed in Peak Clustering equal to 0.6, which can be explained by the fact that this method has significant difficulties when working with high-dimensional data, which often leads to oversimplification or data compression, which results in the loss of important features. Likewise, PDP Growth has a higher information loss of 0.43 since it is not efficient in maintaining a balance between privacy preservation and data usefulness especially in distributed environments.

Thus, although the Federated Generative AE model is more efficient than Peak Clustering and PDP Growth, it still entails a relatively high level of information loss (0.32). This is mainly because of trade-offs that are associated with federated learning, including communication efficiency and data utility in federated learning environments.

5. COMPARISON WITH PRIOR RESEARCH

This section systematically classifies the contribution of the proposed CDDSAE-SL framework relative to the most closely related prior research. Table 2 summarizes the quantitative comparison across four key performance metrics: latency, running time, computational complexity, and information loss.

Table 2: Quantitative comparison with prior research

Metric	Peak Clustering [10]	PDP Growth [11]	Fed. Gen. AE [9]	Proposed CDDSAE
Latency (25 resources)	Highest	Moderate	N/A	17.7% lower than [10]
Running Time (30 features)	Highest	Moderate	N/A	40.3% lower than [10]
Comp. Complexity (1000 records)	54 ms	N/A	48 ms	45 ms (16.7% lower)
Information Loss	0.60	0.43	0.32	0.22 (best)
Privacy Mechanism	Differential Privacy	Differential Privacy	Federated AE	DDSAE + Split Learning
Scalability to High Dimensions	Low	Moderate	Moderate	High

As shown in Table 2, the proposed CDDSAE model consistently outperforms all three baseline methods across all evaluated metrics. Against Peak Clustering [10], the proposed model achieves a 17.7% reduction in latency, a 40.3% reduction in running time, a 16.7% reduction in computational complexity at 1000 records, and a 63.3% reduction in information loss (0.22 vs. 0.60). Compared to PDP Growth [11], the proposed model reduces latency by 13.6% and information loss by 48.8% (0.22 vs. 0.43). Against the Federated Generative AE [9], the proposed framework achieves a 6.3% reduction in computational complexity and a 31.3% reduction in information loss (0.22 vs. 0.32). These gains are attributable to the integrated design of attribute filtering, vertical sharding, and client-side DDSAE compression within a split learning paradigm, which collectively eliminates redundant data transmission, reduces model complexity, and preserves high data utility.

Despite these advantages, the proposed model has several acknowledged shortcomings relative to the literature. First, the evaluation is conducted on a single ECG dataset (PTB-XL), limiting direct generalizability claims to other domains. Second, unlike methods employing formal differential privacy guarantees (e.g., [10], [11]), the proposed model does not provide a mathematically bounded privacy guarantee, relying instead on the architectural separation of split learning. Third, the split learning communication overhead, while reduced compared to full-model sharing, remains a potential bottleneck in very low-bandwidth IoT environments. These shortcomings represent concrete directions for future improvement.

6. CONCLUSION

This paper addressed a central question posed in the Introduction: can split learning combined with deep denoising sparse autoencoders simultaneously resolve the privacy-efficiency trade-offs that plague high-dimensional distributed learning? The experimental results affirm that the answer is yes, with quantifiable and reproducible gains across all evaluated metrics. The five research objectives introduced in Section I have each been achieved. First, the vertical sharding technique was successfully implemented, distributing the dataset into attribute-based shards that reduced individual client complexity and overall latency by 17.7% compared to Peak Clustering. Second, privacy-preserving data transmission was achieved through the CDDSAE encoder, which ensures only low-dimensional denoised representations are shared with the server, preventing raw data exposure. Third, the SL framework improved overall system performance, reducing running time by 40.3% over Peak Clustering and 12.2% over PDP Growth across 30 characteristics. Fourth, the threshold-based attribute filtering mechanism achieved the first level of dimensionality reduction, eliminating redundant and low-importance attributes at the input stage. Fifth, secure and privacy-conscious data handling was demonstrated across all experimental configurations, with the proposed model achieving a minimum information loss of 0.22, which is 63.3% lower than Peak Clustering, 48.8% lower than PDP Growth, and 31.3% lower than Federated Generative AE.

Several limitations and threats to validity must be acknowledged. The evaluation is conducted on a single domain-specific dataset (PTB-XL ECG), which limits the direct generalizability of results to other medical or non-medical high-dimensional datasets; future work should validate the model on diverse benchmarks including tabular healthcare, financial, and IoT sensor datasets. The fixed

threshold-based attribute filtering mechanism (Th_I) may not be universally optimal across datasets with differing entropy distributions, potentially leading to suboptimal attribute selection in edge cases. The split learning framework assumes an honest-but-curious server threat model; it does not defend against gradient inversion attacks or a fully malicious server, and no formal differential privacy bound is provided. Additionally, experiments were conducted on a single hardware configuration (Intel i3-8100, 16GB RAM), and performance characteristics may differ significantly on embedded or resource-constrained edge devices, which are the primary target deployment environment for privacy-preserving IoT applications.

Based on these shortcomings, four concrete future research directions are proposed. First, adaptive threshold selection mechanisms, such as reinforcement learning-based or dataset-aware entropy calibration, should be investigated to replace the current fixed Th_I, improving attribute filtering effectiveness across diverse and evolving datasets. Second, integration of homomorphic encryption or secure multi-party computation into the split learning gradient exchange protocol would provide formal cryptographic privacy guarantees and defend against gradient inversion attacks, substantially strengthening the security posture of the framework. Third, extending the model to federated split learning, where multiple servers coordinate in a federated fashion while clients use CDDSAE compression, would enable broader applicability and improve resilience against single points of failure. Fourth, evaluation of the proposed framework on large-scale, publicly available benchmarks including MIMIC-III clinical records, financial transaction datasets, and smart city IoT sensor streams would provide stronger generalizability evidence and position the framework for real-world deployment.

REFERENCES:

[1] R. Ratra, P. Gulia, N. S. Gill, and J. M. Chatterjee, "Big data privacy preservation using principal component analysis and random projection in healthcare," Math. Probl. Eng., vol. 2022, pp. 1–12, 2022, doi: 10.1155/2022/6402274.

[2] S. Daniel, B. Bernd, and R. David, "Privacy-preserving and lossless distributed estimation of high-dimensional generalized additive mixed models," Stat. Comput., vol. 34, no. 1, 2023, doi: 10.1007/s11222-023-10323-2.

[3] C. Liu, S. Chen, S. Zhou, J. Guan, and Y. Ma, "A general framework for privacy-preserving of data publication based on randomized response techniques," Inf. Syst., vol. 96, p. 101648, 2021.

[4] S. Riyana, "Privacy preservation models for the independent data release of high-dimensional datasets," preprint, 2023, doi: 10.21203/rs.3.rs-2594462/v1.

[5] Z. Chu, J. He, X. Zhang, X. Zhang, and N. Zhu, "Differential privacy high-dimensional data publishing based on feature selection and clustering," Electronics, vol. 12, no. 9, p. 1959, 2023.

[6] Z. Amin, A. Anjum, A. Khan, A. Ahmad, and G. Jeon, "Preserving privacy of high-dimensional data by L-diverse constrained slicing," Electronics, vol. 11, no. 8, p. 1257, 2022.

[7] J. Zhao, M. Yang, R. Zhang, W. Song, J. Zheng, J. Feng, and S. Matwin, "Privacy-enhanced federated learning: A restrictively self-sampled and data-perturbed local differential privacy method," Electronics, vol. 11, no. 23, p. 4007, 2022.

[8] Y. J. Ha, M. Yoo, G. Lee, S. Jung, S. W. Choi, J. Kim, and S. Yoo, "Spatio-temporal split learning for privacy-preserving medical platforms," IEEE Access, vol. 9, pp. 121046–121059, 2021.

[9] X. Jiang, X. Zhou, and J. Grossklags, "Privacy-preserving high-dimensional data collection with federated generative autoencoder," Proc. Priv. Enhancing Technol., vol. 2022, no. 1, pp. 481–500, 2021.

[10] H. Chen, K. Mei, Y. Zhou, N. Wang, M. Tang, and G. Cai, "A density peaking clustering algorithm for differential privacy preservation," IEEE Access, vol. 11, pp. 54240–54253, 2023.

[11] W. Shi, X. Zhang, H. Chen, and X. Zhang, "High dimensional data differential privacy protection publishing method based on association analysis," Electronics, vol. 12, no. 13, p. 2779, 2023.

[12] G. Shen, M. Cai, Z. Huang, Y. Yang, F. Guo, and L. Wei, "LoHDP: Adaptive local differential privacy for high-dimensional data publishing," Concurrency Comput. Pract. Exp., vol. 36, no. 11, 2024.

[13] A. Mishra, T. S. Jabar, Y. I. Alzoubi, and K. N. Mishra, Concurrency Comput. Pract. Exp., vol. 35, no. 26, 2023.

[14] A. Alshammari and K. El Hindi, "Privacy-preserving deep learning framework based on restricted Boltzmann machines and instance

reduction algorithms," Appl. Sci., vol. 14, no. 3, p. 1224, 2024.

[15] P. Wagner, N. Strodthoff, R. Bousseljot, D. Kreiseler, F. I. Lunze, W. Samek, and T. Schaeffter, "PTB-XL, a large publicly available electrocardiography dataset," Sci. Data, vol. 7, no. 1, 2020.

[16] W. Li, X. Zhang, X. Li, G. Cao, and Q. Zhang, "PPDP-PCAO: An efficient high-dimensional data releasing method with differential privacy protection," IEEE Access, vol. 7, pp. 176429–176437, 2019.

[17] B. A. Manjunatha, K. A. Shastry, E. Naresh, P. K. Pareek, and K. T. Reddy, "A network intrusion detection framework on sparse deep denoising auto-encoder for dimensionality reduction," Soft Comput., vol. 28, no. 5, pp. 4503–4517, 2023.

[18] G. Liu, M. Kang, Y. Zhu, Q. Zheng, M. Zhu, and N. Li, "TransNeural: An enhanced-transformer-based performance pre-validation model for split learning tasks," Sensors, vol. 24, no. 16, p. 5148, 2024.

[19] C. G. Allaart, B. Keyser, H. Bal, and A. Van Halteren, "Vertical split learning – an exploration of predictive performance in medical and other use cases," in Proc. IJCNN, 2022.

[20] H. Hafi, B. Brik, P. A. Frangoudis, A. Ksentini, and M. Bagaa, "Split federated learning for 6G enabled-networks: Requirements, challenges, and future directions," IEEE Access, vol. 12, pp. 9890–9930, 2024.

[21] Z. Chu, J. He, X. Zhang, X. Zhang, and N. Zhu, "Differential privacy high-dimensional data publishing based on feature selection and clustering," Electronics, vol. 12, no. 9, p. 1959, 2023.