





New LDP Approach Using VAE

Andres Hernandez-Matamoros^(✉)  and Hiroaki Kikuchi 

Meiji University, 4-21-1 Nakano, Nakano-ku, Tokyo 164-8525, Japan
{matamoros,kikn}@meiji.ac.jp
<https://www.kikn.fms.meiji.ac.jp>

Abstract. Local Differential Privacy allows individuals to share their personal data without compromising their privacy. In traditional data collection and analysis methods, sensitive information such as names, addresses, and other identifying details may be included, making it easy to link the data to a specific individual. On the other hand, Local Differential Privacy enables data to be collected and analyzed in a way that safeguards individual privacy. This makes it possible for people to participate in data collection and analysis without the fear of being identified. While Local Differential Privacy approaches have been proposed for releasing privacy-preserving databases with statistical approximations, they have limitations when dealing with k -dimensional distribution estimations. To address this issue, we propose a solution that guarantees Local Differential Privacy based on the latent space of a Variational AutoEncoder (VAE), which is used to recover the original distribution. We tested our proposal on four real and open datasets with different characteristics, including the number of users, the number of attributes, and their cardinality. The proposed solution outperforms the well-known approach, LoPub. Our work can reduce the average variant distance by the LoPub algorithm from 0.6 to 0.1. These results suggest that the VAE can serve as a useful tool for privacy-preserving data. The source code used in this paper can be downloaded from the following link <https://github.com/phdmatamoros/New-LDP-approach-using-VAE>.

Keywords: Local Differential Privacy · Latent Space · VAE

1 Introduction

Our everyday activities involve sharing personal information with various services, such as online streaming, food delivery, social media, and filling out application forms. These services store our data on their central servers to obtain insights into their user base or train machine learning models. In recent years, the emergence of Differential Privacy [2], also known as Central Differential Privacy (CDP), aims to release databases for statistical analysis of sensitive individual data while preserving user privacy. One drawback of CDP is that users send their data without protection, entrusting it to the central server.

To address this issue, Local Differential Privacy (LDP) was introduced, where users only trust themselves. LDP is a technique that involves encoding and

introducing random noise to users' data before sending it to the central server, allowing the central server to compute the distribution of the users' information. By using this approach, it is possible to protect sensitive data while still allowing it to be used for research or other purposes. LDP is also important in situations where privacy is legally mandated, such as in the European Union's General Data Protection Regulation or the California Consumer Privacy Act.

Ren et al. proposed LoPub [3], in which users encode their data using Bloom filters and perturb the encoded data using the Randomize Response (RR) algorithm [10]. The perturbed-encoded data is then transmitted to the central server, which estimates the multi-dimensional joint distribution using the LASSO algorithm [6] and the Expectation Maximization algorithm [7]. While LoPub demonstrates good performance in datasets with low multi-dimensional joint distributions, it encounters issues with low data utility when the number of attributes is high or the cardinality of an attribute is large.

In LDP, both cardinality and attributes have an impact on the data utility and accuracy of the estimated joint distribution. Firstly, cardinality refers to the number of distinct elements in an attribute. In high-dimensional data with a large number of attributes and high cardinality, LDP approaches may experience a reduction in data utility, resulting in compromised estimation accuracy. This occurs because the introduced noise can make it challenging to identify correlations and patterns, leading to inaccurate estimates.

Secondly, highly correlated attributes can also pose challenges in LDP. When attributes exhibit strong correlations, introducing noise to one attribute can cause the noise to propagate to other attributes, leading to a further decline in data utility. This propagation of noise makes it difficult to accurately estimate joint probability distributions. To mitigate the spread of noise across attributes, Mina [1] made an assumption that the features in the dataset are both independent and categorical. Based on this assumption, a framework was developed that incorporates feature selection to generate a synthetic dataset.

We propose leveraging the latent space of a Variational Auto-Encoder (VAE) [18] within LDP to enhance privacy-preserving data analysis. VAEs have been successfully utilized in various domains and applications [16]. By incorporating the VAE's latent space, which captures meaningful representations of the data, we can improve the utility and accuracy of LDP in scenarios with a high number of attributes or large attribute cardinalities. The VAE acts as a denoising and reconstructing tool, enabling precise estimation of joint probability distributions. This approach offers a more effective and privacy-preserving solution compared to traditional LDP methods, especially in complex, high-dimensional datasets.

To evaluate the performance of our VAE based approach in comparison to a baseline method introduced by Ren [3], we conducted experiments on four publicly available datasets featuring varying numbers of users and cardinalities. Our approach follows the data encoding and perturbation method proposed by [3], but with the incorporation of VAE on the central server. Table 1 highlights the distinctions between the [3] method and our proposed approach. The key contributions of this paper can be summarized as follows:

- We propose a novel LDP approach that utilizes a VAE on the central server to estimate joint probability distributions. In our approach, we use a VAE to learn a latent representation of the data that is shared across all parties. This allows us to estimate the joint probability distribution of the data.
- We explore the impact of attribute cardinality on the reconstruction error during VAE training. We find that as the attribute cardinality increases, the reconstruction error also increases. This is because it becomes more difficult for the VAE to learn a latent representation that is able to capture the diversity of the data.
- We compare our proposed approach to [3], a baseline algorithm for LDP. We find that our proposed approach outperforms [3] in terms of accuracy with same privacy budget.
- We demonstrate the effectiveness of VAE in estimating the joint probability distribution through experiments on four diverse datasets. We find that VAE is able to accurately estimate the joint probability distribution of the data in all four datasets.

Table 1. Difference between LoPub and ours

	LoPub [3]	Ours
User	Hash F. Randomize Response	Hash F. Randomize Response
Central Server	LASSO regression	Latent space (VAE)

The paper is structured as follows: Sect. 2 presents the preliminaries, Sect. 3 describes our proposed approach, and the paper concludes with the Experiments and Conclusion sections.

2 Preliminaries

2.1 Generalizing the Problem

In LDP approaches, the users encode and perturb their data before share their information with a central server. By doing this, the users preserve their anonymity. In this work, we follow the user steps proposed by LoPub [3]. We generalize the LDP problem; a dataset U with N users could be represented as $U = \{u^1, u^2, u^3, \dots, u^N\}$.

The users have the same number of attributes k , and each attribute has a specific domain. Thinking about the n^{th} user has a k -dimensional vector $(u_1^n, u_2^n, u_3^n, \dots, u_k^n)$, the domain of each attribute $j \in \{1, \dots, k\}$ is denoted $\Omega_j = \{\omega_i^1, \dots, \omega_i^{|\Omega_j|}\}$. The cardinality $|\Omega_j|$ means the number of elements in the attribute j .

2.2 Local Differential Privacy (LDP)

LDP proposes that for any user n , a randomization mechanism Ψ satisfies ϵ -LDP if and only if for any two records u^n, w^n , and for any outputs $\tilde{u}_\tau \in \text{Range}(\Psi)$, the probability computed over Ψ 's and $\epsilon > 0$; privacy budget holds

$$\Pr[\Psi(u^n) = \tilde{u}_\tau] \leq e^\epsilon \Pr[\Psi(w^n) = \tilde{u}_\tau] \quad (1)$$

On Eq. (1), we can figure how important is the privacy budget. A smaller ϵ means stronger privacy protection, and viceversa.

2.3 Privacy Analysis

User privacy is preserved by claiming the privacy of local randomizers, which all user run on data records separately. Local perturbation of a specific attribute value can achieve ϵ -LDP, where $\epsilon = 2h \ln \frac{2-f}{f}$, with h being the number of hash functions in the Bloom filter [9] and f the flip bit probability. Based on the sequential composition theorem [14], the local transformation of a k -dimensional data record achieves ϵ -LDP, where:

$$\epsilon = 2kh \ln \frac{2-f}{f},$$

with k being the number of attributes in the original data. Because all users perform the same transformation independently, the above ϵ -LDP guarantee applies to all distributed users.

2.4 Lopub Scheme

The LDP approach relies on the participation of two components: users and a central server. In this work, we utilize the algorithm proposed by Ren [3] to encode and perturb users' data. Our proposal involves replacing the LASSO and EM algorithms with VAE in the central server.

User. This section explains how users encode and perturb their data, an approach consisting of two main steps:

- Encoding user information. The user input is represented using a Bloom filter (BF) \mathcal{H} , a technique used to test whether an element is a member of a set; it is a probabilistic data structure proposed by Bloom [9]. To encode each u_j^n , the user incorporates BF using a set \mathcal{H}_j of hash functions that are designed for U_j , where U_j is the j^{th} attribute of U . Specifically, the user applies h_j hash functions $\mathcal{H}_{j,1}, \dots, \mathcal{H}_{j,h_j}$ from \mathcal{H}_j to map u_j^n to a length- m_j bit string s_j^n , where m_j is the length of the BF. Therefore, u_j^n is inserted into a length m_j bit BF using h_j hash functions from \mathcal{H}_j , represented as $H_j(\omega)$, where

$s_j^n[b]$ denotes the b^{th} bit of the bit string s_j^n . The length of BF m_j for U_j is computed as:

$$m_j = \frac{\ln \frac{1}{p}}{\ln 2^2} |\Omega_j|, \quad (2)$$

where $|\Omega_j|$ is the cardinality of the attribute U_j and p is the false positive probability; in our experiments we set $p = 0.022$.

- Perturbing the data. Randomized Response (RR) is a method proposed by Warner [10] that allows interviewee to give their answers while maintaining confidentiality. Randomly whether the question is to be answered truthfully is unknown to the interviewer. RR is applied after encoding each step, where each bit $s_j^n[b]$ ($b = 1, 2, \dots, m_j$) is randomly flipped using the following rule:

$$\hat{s}_j^i = \begin{cases} s_j^n & \text{with probability of } 1 - f, \\ 1 & \text{with probability of } f/2, \\ 0 & \text{with probability of } f/2 \end{cases} \quad (3)$$

Where $f \in [0, 1]$ is the probability of flipping a bit randomly. Once the randomized BF s_j^n is obtained, the n^{th} user combines s_1^n through s_k^n to create a bit vector $(s_1^n || \dots || s_k^n)$, which consists of $(\sum_{j=1}^k m_j)$ bits. This resultant vector is transmitted to the server.

Central Server. After users encode and perturb their data, they send their data to the central server, which receives the distribution of users with random noise added by RR. For each bit b in each attribute j , the central server counts the number of frequencies of the perturbed value \hat{s}_j^i as $\hat{y}_j[b] = \sum_{i=1}^N \hat{s}_j^i[b]$. Next, the original count $y_j[b]$ is estimated as

$$y[b] = \frac{\hat{y}[b] - \frac{fN}{2}}{1 - f},$$

where after the original count is computed, the candidate bit matrix is created using a candidate set of Bloom filters \mathcal{H} , as $M = [\mathcal{H}_1(\Omega_1) \times \mathcal{H}_2(\Omega_2) \times \dots \times \mathcal{H}_d(\Omega_d)]$, where d is the number of attributes. As we illustrate, the block diagram in Fig. 1 reviews how it works from the previous steps applied by the central server and gives an example with $k = 2$ -way, estimating the distribution of two attributes in the following steps. To estimate the distribution from the noise data using a regression technique, $y = M\beta$. LASSO, is a linear regression technique that performs regularization order to improve prediction accuracy; it was introduced by [8]. If the reader wants to read more about the whole process, please refer to [3].

3 Proposed Scheme

3.1 VAE Preliminaries

Auto-Encoders (AE) were introduced by Hinton in 1986 [17]. They are designed to encode input data into an essential representation and then decode it back

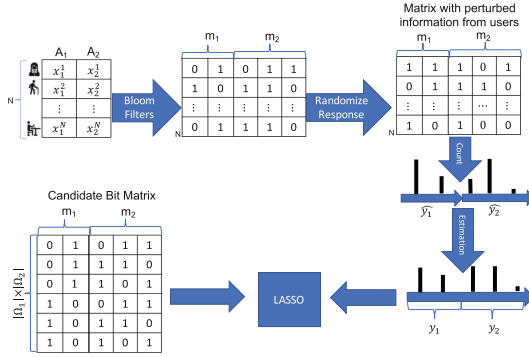


Fig. 1. Central server block diagram proposed by [3] computing the joint distributions of two attributes for N users.

to create a reconstructed input that is as similar as possible to the original input. AE consists of two parts: the encoder and the decoder. The input to the encoder is the data Γ . The output of the encoder is called Y , which is the reduced representation of Γ in a latent space. Next, the decoder is adjusted to reconstruct the data Γ . Finally, the decoder reconstructs the original data Γ from Y by minimizing the Euclidean distance between Γ and Γ' .

Later, in 2013, Kingma proposed a variation of AE called Variational Auto-Encoder (VAE) [18]. The main difference between AE and VAE is that the encoder in AE outputs latent vectors, whereas VAE imposes a constraint on this latent distribution, forcing it to be a normal distribution. VAE has two main stages: training and testing. In the training stage of VAE, a reconstruction error function $RE(\Gamma, \Gamma')$ is defined as follows:

$$RE(\Gamma, \Gamma') = \sqrt{\sum (\Gamma - \Gamma')^2} + MMD(V, Sample_z), \quad (4)$$

where V is the output of VAE's encoder and $Sample_z$ is drawn from $\mathcal{N}(0, 1)$, the Maximum Mean Discrepancy (MMD) distance measures the distance between the feature maps of two probability distributions. A smaller distance suggests that the two distributions are more alike.

3.2 VAE Model

Training. During the training step, the Algorithm 1 is applied to synthesizing two datasets.

- X contains the encoded information of each attribute.
- X' is X after being perturbed using RR.

Using these datasets, our model trains to create a latent space for each attribute available in the dataset. VAE is trained on two datasets: a perturbed dataset X' and a non-perturbed dataset X , where $RR(X) = X'$ (Fig. 2). These

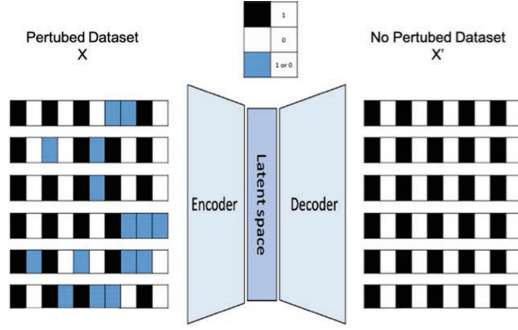


Fig. 2. How train VAE.

Algorithm 1. Creating Datasets

```

Require:  $j^{th}$  attribute
Require:  $t = 1000$ 
Require:  $X_j = []$ 
Require:  $X'_j = []$ 
Require:  $L_j = []$ 
Require:  $f$ 
  for element in  $j$  do
     $\mathcal{H}_j$ 
    for each  $p = 1, 2, \dots, t$  do
       $X_j \leftarrow [X_j, \mathcal{H}_j(\textit{element})]$ 
       $X'_j \leftarrow [X'_j, RR(\mathcal{H}_j(\textit{element}), f)]$ 
       $L_j \leftarrow [L_j, \textit{element}]$ 
    end for
  end for
return  $X_j, X'_j, L_j$ 

```

- ▷ Attribute on Dataset
- ▷ Perturbed times
- ▷ No perturbed
- ▷ Perturbed
- ▷ Attributes Label
- ▷ Flip Bit Probability
- ▷ The Bloom filters on j^{th} attribute
- ▷ Append
- ▷ Append
- ▷ Append

datasets are created by the central server using the method described in Algorithm 1. For this experiment, the central server creates one thousand artificial users per attribute in both X and X' datasets. The datasets are then split into training and validation sets with a ratio of 90% and 10%, respectively.

The Algorithm 2 invokes Algorithms 1 and 3. Given a specific value of f , Algorithm 1 uses it to create datasets X and X' for each attribute. Algorithm 3 is used to train the VAE. A summary of the VAE is available in Appendix A.1. In Algorithm 3, a simplified algorithm for training the VAE is presented. For further details on how to train the VAE, please refer to [11].

The outputs of Algorithm 2 are the encoder E_j and the latent space Y_j for attribute j . The latent space models the cardinality for each element in the attribute. Examples of a 2D latent space for the “Marital Status” and “Sex” attributes in the Adult dataset are shown in Fig. 5. In our experiments, we set the latent space as 4D.

Latent Space Evaluation. Once the model has been trained on synthetic datasets, it is evaluated using real datasets. To achieve this, the records of perturbed data are transformed into the latent space by the VAE encoder. Sub-

Algorithm 2. Main algorithm

```

Require:  $t = 1000$  ▷ Perturbed times
Require:  $k$  ▷ Number of attributes on the original dataset
Require:  $f$  ▷ Flip Bit Probability Value
  for each  $j=1, \dots, k$  do
     $X_j, X'_j, L_j \leftarrow \text{Creating Datasets}(j, f, t)$  ▷ Algorithm 1
     $E_j, V_j \leftarrow \text{TrainingVAE}(X_j, X'_j)$  ▷ Algorithm 3
  end for
return  $E_j, V_j, L_j$ 

```

Algorithm 3. Training VAE

```

Require:  $X_j, X'_j$  ▷ Created by Algorithm 1
Require: Encoder of VAE please refer to Appendix A.1
Require: Decoder of VAE please refer to Appendix A.1
Require: epochs=1000
Require: Optimizer Adam, lr=0.0001
   $patience = 0$ 
  for each  $epoch = 1, \dots, \text{epochs}$  do
     $Y_j \leftarrow \text{Encoder}_j(X'_j)$ 
     $W_j \leftarrow \text{Decoder}_j(Y_j)$ 
     $Re_j \leftarrow RE(X_j, W_j)$  ▷ Eq. 4
    Using lr update internal parameters of VAE
     $Re_{epoch} \leftarrow \text{average of } Re_j$ 
    if  $Re_{epoch} \geq Re_{epoch-1}$  then
       $patience \leftarrow patience + 1$ 
    end if
    if  $patience=16$  then
      finish training
    end if
  end for
return  $\text{Encoder}_j, Y_j$ 

```

sequently, the approach calculates the Euclidean distance between the user's coordinates and the latent space created during the training step for a specific attribute.

For simplicity, the latent space Y_j which belongs to the j^{th} attribute using f value will be represented as Y in the following expressions. Y is a matrix with s rows and d columns, where s is the number of elements of latent space and d is the dimension of the latent space. In our experiments the dimension of latent space d is four and the number of elements of latent space s is $900|\Omega_j|$ for each j^{th} attribute.

$$Y = \begin{bmatrix} Y_{1,1}, Y_{1,2}, \dots, Y_{1,d} \\ Y_{2,1}, Y_{2,2}, \dots, Y_{2,d} \\ Y_{3,1}, Y_{3,2}, \dots, Y_{3,d} \\ \dots \\ Y_{s,1}, Y_{s,2}, \dots, Y_{s,d} \end{bmatrix}$$

The latent space evaluation consists of two steps, as shown in Fig. 3;

- The first step is performed by the encoder E_j , which transforms i^{th} user's record into the latent space. E_j outputs a vector V of size d , where each component V_i corresponds to a dimension in the latent space.

$$E_j(i^{th} \text{ user's record}) = (V_1, \dots, V_d)$$

- We compare the vector V with the latent space Y_j , which was computed during the training stage. Our proposed method involves computing the Euclidean distance between V and each row in the matrix Y_j . By identifying the index of the row that exhibits the closest similarity to V , we evaluate this index in L_j which is created using Algorithm 1, to determine the potential element in the attribute.

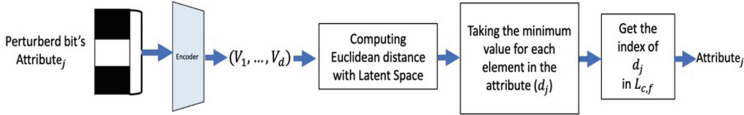


Fig. 3. Latent Space Evaluation.

Algorithm 4. Inference Attribute

```

Require:  $E_i, Y_i, L_i$ 
Require:  $D = [ ]$  ▷ Recovered Dataset
Require: fpb
  for each  $u = 1, 2, \dots, Users$  do
    for each  $j=1, \dots, k$  do
       $V \leftarrow E_j(j)$  ▷ Transforming  $V$  into Latent Space
       $d_j = dis(V, Y_j)$  ▷ Computing Euclidean distance
       $j^* = argmin(d_j)$ 
       $D(u, j) \leftarrow L_j(j^*)$ 
    end for
  end for
return  $D$ 

```

After Algorithm 4 finishes, we obtain D , where each row represents an anonymous user and each column represents a possible attribute. This matrix can be used to estimate the joint distribution of the original dataset. To calculate the joint probability distribution of two or more attributes for the users in D , follow these steps:

- Calculate the total number of users in the D .
- Count the frequency of each combination of attribute values for the users of interest. It is the joint frequency distribution.
- Divide the joint frequency of each combination by the number of users to obtain the joint probability of that combination.

4 Experiments

4.1 Experimental Method

We tested our approach on four open datasets from different areas. The Nursery dataset [15] was originally created in the 1980s to evaluate applications

for nursery schools in Europe. The NHANES dataset [12] was used in the PWS Cup 2021 [13] to provide anonymized healthcare data. The Adult dataset [4] is one of the most popular datasets used to measure the performance of CDP and LDP approaches. The Bank dataset [5] contains information about marketing campaigns. Table 2 summarizes the datasets, showing the number of users, attributes, their cardinality, and their size after encoding.

The default parameters for our approach are as follows: we use $h = 5$ hash functions for all four datasets. The value of m varies depending on the dataset’s cardinality and could be calculated using the Eq. 2.

Table 2. Statistics of Datasets

Dataset	Users	Attributes	Cardinality		m	
			min	max	min	max
Adult	45223	8	2	16	8	64
Bank	45212	10	2	12	8	47
Nursery	12960	9	2	5	8	20
NHANES	4190	5	2	6	8	23

4.2 Results

VAE Reconstruction Error (RE). We evaluated the Reconstruction Error (RE) during the training stage of the VAE using Eq. 4. The results are presented in Figs. 4(a)–(b). Figure 4(a) displays the results for the Adult dataset, where the attribute *marital – status* exhibits a lower reconstruction error compared to the attribute *sex*. A comparison between their latent spaces is depicted in Figs. 5(a) and (b). For simplicity, the latent spaces are shown in 2D, although 4D were used in the experiments.

Figure 4(b) shows the results for the NHANES dataset, with the attribute *Education* having a lower RE than the others. The attribute *Qm* demonstrates the highest reconstruction error.

Joint Probability. We randomly selected k -way joint probabilities of attributes one hundred times. To analyze the joint distributions, we used the Average Variance Distance (AVD) metric to quantify the difference between the real and computed data. The AVD distance, as used by [3], is defined as follows:

$$AVD = \frac{1}{2} \sum_{\omega \in \Omega} |P(\omega) - Q(\omega)|. \quad (5)$$

Figures 6(a)–(d) show the results of VAE and LASSO using color and grayscale, respectively. The x -axis corresponds to the AVD distance, while the y -axis shows the algorithms with a flip bit probability $f = 0.5$. LASSO regression,

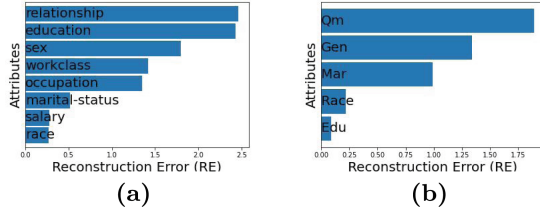


Fig. 4. Reconstruction Error with $f = 0.5$ (a) Adult, (b) NHANES Datasets.

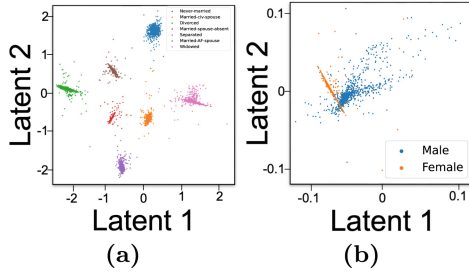


Fig. 5. Latent Space for Adult dataset with $f = 0.1$ on training stage (a) Marital Status, (b) Sex Attributes.

proposed by [3], was used to recover the original distribution in LDP schemes. In summary, the comparison of LASSO and VAE models on different datasets revealed interesting insights. In the NHANES dataset, LASSO struggled to capture complex patterns, as indicated by increasing AVD distances with higher values of k -way. On the other hand, VAE consistently outperformed LASSO, suggesting its ability to effectively capture latent representations and reproduce patterns in the NHANES dataset.

Similarly, on the Adults dataset, LASSO exhibited decreasing predictive accuracy with higher model complexity, while VAE consistently outperformed LASSO with lower AVD distances. This indicates that VAE’s capacity to capture latent representations and generate data is advantageous for the Adults dataset.

The Nursery dataset posed challenges for LASSO, as it struggled to accurately predict values with higher values of k -way. In contrast, VAE significantly outperformed LASSO on the Nursery dataset, indicating its superior ability to capture complex relationships and reproduce values accurately.

The Bank dataset showed relatively good performance for LASSO, with low AVD distances across all values of k -way. VAE slightly outperformed LASSO, indicating its capability to capture and reproduce underlying patterns in the Bank dataset.

In summary, VAE consistently outperformed LASSO in terms of AVD distances across different datasets and values of k -way. VAE’s ability to capture latent representations and generate data allows it to capture complex patterns and relationships better, resulting in improved predictive accuracy.

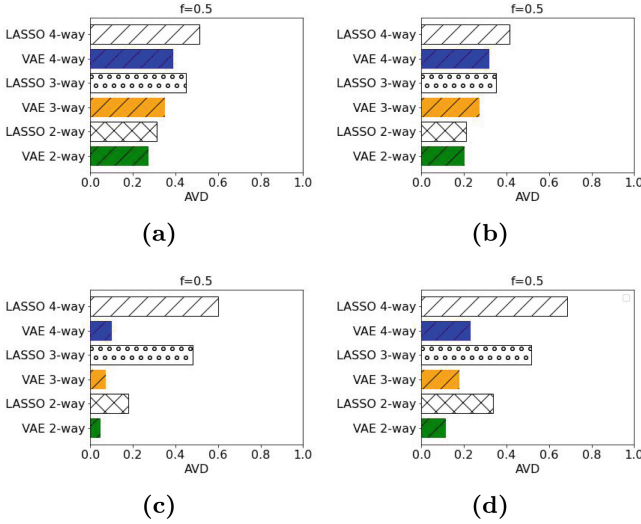


Fig. 6. Accuracy (a) Adult, (b) Bank, (c) Nursery, (d) NHANES Datasets.

Figures 7(a)–(d) display the results of VAE and LASSO with $f = 0.5$ when varying the number of users N . The dotted line represents the results of Lasso, while the solid line represents the results of VAE. The blue color represents the results of k -way=2, yellow represents k -way=3, and green represents k -way=4. Then, we discuss the performance for each dataset

In the Adult dataset shown in Fig. 7(a), VAE performs better than LASSO for all values of k -way when $N > 15000$. In the case where k -way is two and $N < 15000$, LASSO and VAE have similar performance. However, when $N = 5000$, LASSO shows better performance than our approach, but the difference between the two is minimal.

In the Bank dataset shown in Fig. 7(b), VAE outperforms LASSO for k -way equal to four when $N > 10000$. For k -way equal to three and $N > 15000$, VAE also outperforms LASSO. However, for k -way equal to two, VAE performs better when $N < 30000$, after which LASSO and VAE exhibit similar performance.

In the Nursery dataset shown in Fig. 7(c), VAE outperforms LASSO for k -way={3,4}, regardless of the number of users. For k -way=2 and $N > 3000$, VAE also outperforms LASSO.

Finally, in NHANES dataset, VAE outperforms LASSO for k -way={2, 3, 4}, regardless of the number of users.

The difference in AVD values across the Dataset is related to the cardinality of the attributes. For instance, the Nursery dataset has more attributes than the Adult dataset, but the Adult dataset exhibits the maximum cardinality, as shown in Table 2. The experimental results demonstrate that VAE outperforms LASSO for all four datasets analyzed in this paper when the number of users is greater than half the size of the original dataset.

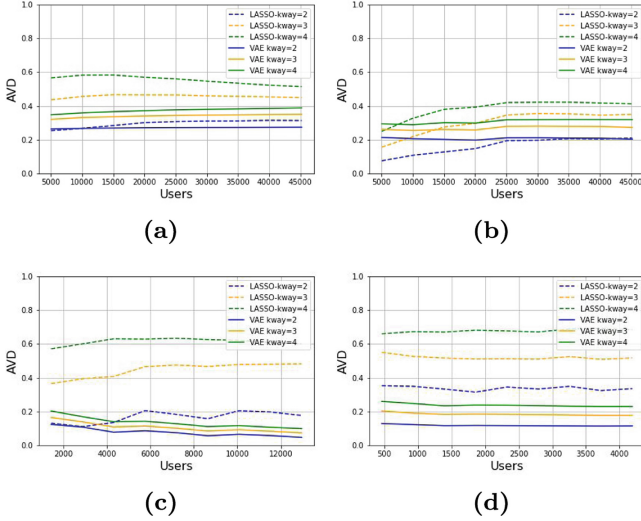


Fig. 7. Accuracy versus N users with $f = 0.5$ (a) Adult, (b) Bank, (c) Nursery, (d) NHANES Datasets.

5 Conclusions

In conclusion, this work proposes the use of the latent space of a VAE in the central server of the LDP scheme to calculate the joint probability. The approach is tested on real datasets with varying numbers of users and attribute cardinalities using a single VAE model. The results show that VAE outperforms LASSO regression, as it allows each attribute to have its own independent latent space, preventing noise from one attribute from affecting others. The AVD of VAE exhibits stable behavior across different numbers of users, indicating that the LDP model using VAE can be applied to extract information from datasets that increase over time. Future work includes investigating the relationship between attribute cardinality and corresponding latent space to develop an improved VAE model, which could be used to create synthetic datasets by computing correlations between attributes.

Acknowledgements. This work was supported by JST, CREST Grant Number JPMJCR21M1, Japan.

A Appendix

A.1 VAE's Summary

This appendix provides an overview of the VAE architecture. For more detailed information on these concepts, please refer to [11].

Table A. Encoder Summary		Table B. Decoder Summary	
Layer(Type)	Output Shape	Layer(Type)	Output Shape
Conv1d-1	$[-1, 16, (input_{dim} - 1)]$	Linear-1	$[-1, 1, 32 \times (input_{dim} - 4)]$
ReLU-2	$[-1, 16, (input_{dim} - 1)]$	ReLU-2	$[-1, 1, 32 \times (input_{dim} - 4)]$
BatchNorm1d-3	$[-1, 16, (input_{dim} - 1)]$	ConvTranspose1d-3	$[-1, 1, (input_{dim} - 4)]$
Conv1d-4	$[-1, 16, (input_{dim} - 2)]$	ReLU-4	$[-1, 1, (input_{dim} - 3)]$
ReLU-5	$[-1, 16, (input_{dim} - 2)]$	BatchNorm1d-5	$[-1, 1, (input_{dim} - 3)]$
BatchNorm1d-6	$[-1, 16, (input_{dim} - 2)]$	ConvTranspose1d-6	$[-1, 1, (input_{dim} - 2)]$
Conv1d-7	$[-1, 32, (input_{dim} - 3)]$	ReLU-7	$[-1, 1, (input_{dim} - 2)]$
ReLU-8	$[-1, 32, (input_{dim} - 3)]$	BatchNorm1d-8	$[-1, 1, (input_{dim} - 2)]$
BatchNorm1d-9	$[-1, 32, (input_{dim} - 3)]$	ConvTranspose1d-9	$[-1, 1, (input_{dim} - 1)]$
Conv1d-10	$[-1, 32, (input_{dim} - 4)]$	ReLU-10	$[-1, 1, (input_{dim} - 1)]$
ReLU-11	$[-1, 32, (input_{dim} - 4)]$	BatchNorm1d-11	$[-1, 1, (input_{dim} - 1)]$
BatchNorm1d-12	$[-1, 32, (input_{dim} - 4)]$	ConvTranspose1d-12	$[-1, 1, input_{dim}]$
Linear-13	$[-1, 64]$		
ReLU-14	$[-1, 64]$		
BatchNorm1d-15	$[-1, 64]$		
Linear-16	$[-1, 16]$		
ReLU-17	$[-1, 16]$		
Linear-18	$[-1, 4]$		
Linear-19	$[-1, 4]$		

References

1. Alishahi, M., Moghtadaiee, V., Navidan, H.: Add noise to remove noise: local differential privacy for feature selection. *Comput. Secur.* **123**, 102934 (2022). <https://doi.org/10.1016/j.cose.2022.102934>. ISSN: 0167-4048
2. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. *Found. Trends Theoret. Comput. Sci.* **9**(3–4), 211–407 (2014)
3. Ren, X., et al.: *LoPub*: high-dimensional crowdsourced data publication with local differential privacy. *IEEE Trans. Inf. Forensics Secur.* **13**, 2151–2166 (2018). <https://doi.org/10.1109/TIFS.2018.2812146>
4. Adult: UCI Machine Learning Repository (1996)
5. Moro, S., Rita, P., Cortez, P.: Bank Marketing. In: UCI Machine Learning Repository (2012)
6. Zou, H., Hastie, T., Tibshirani, R.: On the “degrees of freedom” of the lasso, Institute of Mathematical Statistics, 35, *The Annals of Statistics*
7. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc.: Ser.* **39**, 1–22 (1977)
8. Santosa, F., Symes, W.W.: Linear inversion of band-limited reflection seismograms. *SIAM J. Sci. Statist. Comput.* **7**(4), 1307–1330 (1986). <https://doi.org/10.1137/0907087>
9. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* **13**(7), 422–426 (1970)
10. Warner, S.L.: Randomized response: a survey technique for eliminating evasive answer bias. *J. Am. Statist. Assoc.* **60**, 63–69 (1965)
11. Alzubaidi, L., Zhang, J., Humaidi, A.J., et al.: Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *J Big Data* **8**, 53 (2021). <https://doi.org/10.1186/s40537-021-00444-8>
12. Kikuchi, H.: PWS Cup: Data Anonymization Competition ‘Diabetes’ (2021). <https://github.com/kikn88/pwscup2021>. Accessed 10 May 2023
13. PWS: PWS (2021). <https://www.iwsec.org/pws/2021/cup21.html>. Accessed 10 May 2023

14. McSherry, F.D.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: Proceedings of the ACM SIGMOD, pp. 19–30 (2009)
15. Rajkovic, V.: Nursery, UCI Machine Learning Repository (1997)
16. Bengio, Y., Yao, L., Alain, G., Vincent, P.: Generalized denoising auto-encoders as generative models. In: Advances in Neural Information Processing Systems (2013)
17. Rumelhart, D.E., Hinton, G.E., Williams, R.J. : Learning internal representations by error propagation. In: Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations, pp. 318–362. MIT Press (1987)
18. Diederik, P.: Kingma. Auto-Encoding Variational Bayes, ICLR, Max Welling (2014)