PRIVACY-PRESERVING HIGH-DIMENSIONAL DATA COLLECTION WITH FEDERATED GENERATIVE AUTOENCODER

Xue Jiang
Department of Informatics
Technische Universität München
{xue.jiang}@tum.de

Xuebing Zhou
Munich Research Center
Huawei Technologies Düsseldorf GmbH
{Xuebing.Zhou}@huawei.com

Jens Grossklags
Department of Informatics
Technische Universität München
{jens.grossklags}@tum.de

ABSTRACT

Business intelligence and AI services often involve the collection of copious amounts of multi-dimensional personal data. Local differential privacy (LDP) is currently considered as a state-of-the-art solution for privacy-preserving data collection. However, existing LDP-based data collection algorithms are not applicable to high-dimensional data; not only because of the increase in computation and communication cost, but also poor data utility.

In this paper, we aim at using the idea of data synthesis to address the *curse-of-dimensionality* problem in LDP-based data collection algorithms. We propose DP-FED-WAE, an efficient privacy-preserving framework for collecting high-dimensional categorical data. With the combination of a generative autoencoder, federated learning, and differential privacy, our framework is capable to privately learn the statistical distributions of clients' local data and generate high-utility synthetic data on the server side without revealing clients' private information. By evaluating the framework on real-world datasets with 69$\sim$124 attributes, we show that our framework outperforms the LDP-based baseline algorithms in capturing complex statistical properties of real data and the generated synthetic data achieve less than 3% accuracy loss in AI training tasks. Extensive experimental results demonstrate the capability and efficiency of our framework in synthesizing high-dimensional data while striking a satisfactory utility-privacy balance.

1 INTRODUCTION

With the rapid development of network and computer technologies, large and diverse quantities of multi-dimensional person-specific data are frequently generated on local devices such as smartphones and IoT sensors. These data usually contain rich statistical information describing user profiles, which is valuable for data analysts to explore the hidden correlations and patterns of data from different perspectives. In principle, the more dimensions the data consist of, the more information can be used for describing the user groups and building effective AI services. However, since the data are generated based on individuals' ongoing behaviors, the direct collection can reveal sensitive information about them and lead to severe privacy problems (*e.g.*, Berman et al. (2018)).

Local differential privacy (LDP, Kasiviswanathan et al. (2011)), as a state-of-the-art data anonymization mechanism, has been recently deployed for privacy-preserving data collection (*e.g.*, Erlingsson et al. (2014); Bun et al. (2018); Wang et al. (2018)). However, prior research on LDP-based data collection mainly focuses on one-dimensional statistical information. Since the attributes in multi-dimensional data are usually correlated, the server is particularly interested in learning the correlations and joint distributions among attributes. Nevertheless, directly applying these LDP algorithms for estimating the joint distributions of multi-dimensional data faces the *curse-of-dimensionality* problem: the domain size increases exponentially with data dimensionality, which will lead to extremely large communication cost and storage complexity, as well as a significant degradation in data utility. To reduce the large communication overhead, later works in Ren et al. (2018) and Wang et al. (2019b) propose to separately collect data of each dimension under LDP and then use the randomized data to estimate the m-way joint distributions on the server side. However, the algorithms still suffer from high computational complexity and low data utility when m is large. Based on these facts, solutions for privacy-preserving high-dimensional data collection are still greatly needed.

Recently, data synthesis has been considered as a promising approach for addressing data privacy issues in business intelligence & AI services. In this paper, we follow the idea of data synthesis and propose DP-FED-WAE, an efficient and privacy-preserving framework for high-dimensional categorical data collection. Different from prior works on differentially-private synthetic data generation algorithms (Park et al. (2018); Torkzadehmahani et al. (2019)), which mainly focus on the *centralized setting* where the real data are already collected by the server, our framework conducts the data synthesis *without* collecting real local data. The main idea is to train a (generative) Wasserstein Autoencoder (Tolstikhin et al. (2018)) (WAE) under the federated learning (FL) setting to learn the distributions of the high-dimensional local data and then to generate high-quality synthetic data on the cloud server. Moreover, we propose a novel local randomization algorithm SIGNDS, which is applied on a client's local updates to prevent potential privacy leakages in FL and provide comprehensive privacy protection to our framework. By evaluating the framework on a number of real-world datasets containing $69 \sim 124$ classification attributes, we show that the synthetic data generated by our framework always preserve much closer joint distributions and correlations to real data in comparison to LDP-based baseline algorithms. Also, with a local privacy guarantee $\epsilon = 8$ and using the synthetic data generated by our framework for training machine learning models, we achieve less than 3% and at best even less than 1% accuracy loss, in contrast to $10\% \sim 30\%$ using the data generated by the baseline algorithms. Extensive evaluation experiments demonstrate that our framework has outperforming capability and efficiency in collecting high-dimensional data while striking a satisfactory utility-privacy balance.

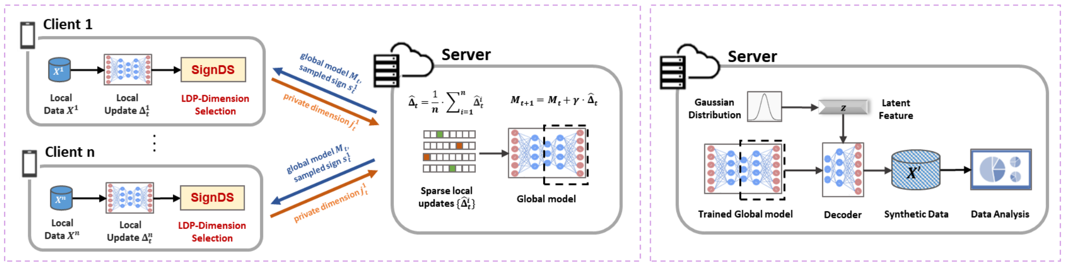
2 DP-FED-WAE FRAMEWORK

In this paper, we consider the scenario that a central server aims to estimate the correlations and joint distributions of high-dimensional personal data held by a number of local clients. To protect local privacy for each client, the data collection should be performed in a private manner, which requires the server not to have access to raw local data but only anonymized versions of data or their feature representations. The collected data and representations can be used to generate similar synthetic data on the server side for data analysis or designing new AI services.

Based on the limitations of existing LDP-based data collection algorithms, we propose DP-FED-WAE, an efficient privacy-preserving framework for collecting high-dimensional categorical data. The overall workflow is presented in Figure 1. Following the idea of data synthesis techniques, the framework utilizes generative autoencoders to learn the statistical distributions and correlations of clients' local data and then to generate high-utility synthetic data on the server side. Since we focus on the scenario where the real data are distributed on local devices and are *inaccessible* to the server, we propose to train the generative autoencoder under the FL setting, which only exchanges model parameters instead of raw local data during the training process. Furthermore, we incorporate DP into the training process to provide a comprehensive privacy protection for the framework.

In the following, we will briefly introduce the main steps of our framework.

Data Pre-processing and Design of the Generative Model Since the original data are categorical and cannot be directly used for training generative models, we firstly use *one-hot encoding* to convert the categorical records into binary vectors as the training data. Then, we design our generative model based on the dimension of training data. In this paper, we choose the Wasserstein Autoencoder (WAE) in our framework. The WAE model preserves the encoder-decoder architecture. The goal of training is to find an optimal set of parameters, which minimizes the distance between the inputs and reconstructed outputs while restricting the latent distribution to follow a certain prior distribution p_z. A detailed description of the WAE model is presented in Appendix B.1.


Client 1
X^1
SignDS
Local Data X^1
Local Update Δ_t^1
LDP-Dimension Selection
Client n
X^n
SignDS
Local Data X^n
Local Update Δ_t^n
LDP-Dimension Selection
global model M_t, sampled sign s_t^1
private dimension j_t^1
global model M_t, sampled sign s_t^1
private dimension j_t^1
Server
$\widehat{\Delta}_t = \frac{1}{n} \cdot \sum_{i=1}^{n} \widehat{\Delta}_t^i$
$M_{t+1} = M_t + \gamma \cdot \widehat{\Delta}_t$
Sparse local updates $\{\widehat{\Delta}_t^i\}$
Global model
Server
Gaussian Distribution
z
Latent Feature
Trained Global model
Decoder
X'
Synthetic Data
Data Analysis

(a) Train the Generative Model **(b) Generate Synthetic Data**

Figure 1: Overview of the DP-FED-WAE framework. The generative Wasserstein Autoencoder is trained under the federated setting, which learns the distributions of real local data. An LDP algorithm SIGNDS is applied to the local updates to provide strict local privacy guarantees. After the model is trained, the *decoder* part is used to generate high-utility synthetic data. The generated data will be used for data mining and building AI services.

Training the Generative Model Since the real data are distributed on local devices and cannot be accessed by the server, we adopt the FL mechanism to train the WAE model. Moreover, we propose a novel local randomization algorithm SIGNDS, which is applied on clients' local updates before uploading them to the server. For each local update vector, the algorithm privately selects one of the top-k dimensions by the update values and returns the selected dimension index to the server. The server then constructs sparse local updates based on these returned indices and uses their aggregation to update the global model. We prove that the randomization algorithm follows a strict ϵ-LDP definition. Details of the SIGNDS algorithm as well as the model training process are described in Appendix B.2. In comparison to previous DP-FL frameworks that apply DP perturbation on the server side (*e.g.*, Augenstein et al. (2020)), our framework ensures that the server cannot gain any information of the real local updates, which efficiently prevents local privacy leakages in FL and provides a strong *local privacy guarantee* to each client's local dataset.

Synthetic Data Generation and Post-processing Once the WAE model has been trained, the server can randomly sample latent features from p_z and feed them into the *decoder* to generate synthetic data. The decoder's output has the same dimension as the encoded input with each dimension value between 0 and 1. Finally, we convert the numerical outputs back to categorical form. Given an output vector, we split it into pieces of sub-vectors, each representing one categorical attribute. For each sub-vector, we choose the entry with the maximum value as the attribute value and finally concatenate all the attribute values into one vector to construct the final synthetic dataset.

3 EXPERIMENTS AND RESULTS

In the experiments, we use LOPUB (Ren et al. (2018)) and LOCOP (Wang et al. (2019b)) as our baseline algorithms. Both algorithms directly apply LDP on the *local data*. The randomized data are aggregated on the server side for constructing the synthetic dataset. Details of the baseline algorithms can be found in Appendix C.1. We use four real-world datasets with 69 $\sim$ 124 attributes to evaluate the performance of our framework. Detailed descriptions of the datasets, WAE model structures and parameter configurations are reported in Appendix C.2 and Appendix C.3. We, respectively, evaluate the utility of the synthetic data in terms of statistical distributions and AI training performance and briefly present some of the results in the following. Extended experimental results can be found in Appendix D.

Comparison of Joint Distributions For the analysis of joint distributions, we use the Average Variant Distance (AVD) to quantify the distribution difference between the real data and synthetic data, as suggested in Ren et al. (2018), which is defined as $AVD = \frac{1}{2}\sum_{\omega\in\Omega}|P_{real}(\omega) - P_{syn}(\omega)|$. $P_{real}(\omega)$ and $P_{syn}(\omega)$ are m-way joint distributions of real data and synthetic data. Clearly, the smaller AVD, the better the utility of the synthetic data. In Figure 2, we present the results of m-way AVD ($m \in \{2, 3, 4, 5, 6\}$) between the real data and the synthetic data under $\epsilon = 4$. It can be seen that our framework constantly outperforms the baseline algorithms for all the datasets. More

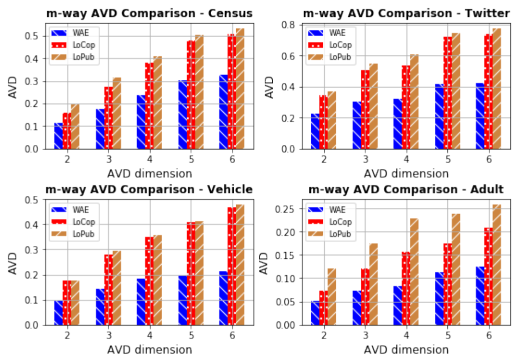


Figure 2: Average total variation distance (AVD) of m-way joint distributions ($m\{2, 3, 4, 5, 6\}$) between the real data and synthetic data generated by our framework (*WAE*) as well as by the baseline algorithms (*LoPub*, *LoCop*) with $\epsilon = 4$.

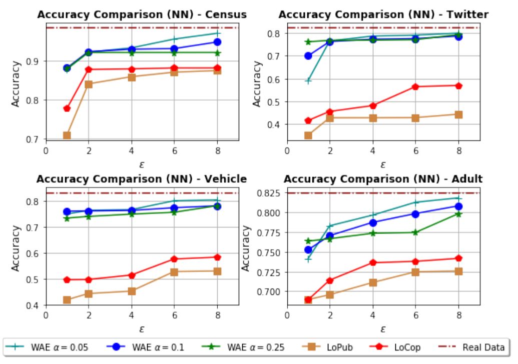


Figure 3: Classification accuracy of neural network (NN) trained with real data (*Real Data*) and synthetic data generated by our framework (*WAE*) as well as by the baseline algorithms (*LoPub*, *LoCop*) under different privacy levels.

specifically, the AVD of the baseline algorithms is close to our framework when m is small, yet gets distinctively larger with an increase of m. This indicates that our framework has a better capability in capturing information of high-dimensional joint distributions of real data.

AI Training Performance Next, we evaluate the utility of synthetic data in different AI training tasks. More specifically, we train two classification models M_{real}, M_{syn}, respectively, with real data and synthetic data, and test both models with an amount of held out real data. Then, we compare Acc_{real} and Acc_{syn}, i.e., the test accuracy of M_{real}, M_{syn}. In Figure 3, we present the evaluation results on a 2-layer Neural Network (NN) under different privacy levels. It can be seen that the Acc_{syn} of the baselines shows in general a distinctive distance from Acc_{real} on both evaluation models and only has slight improvement with larger privacy budget ϵ. In comparison, the Acc_{syn} of our method consistently outperforms the baselines for both classification algorithms. With an increase of ϵ, the Acc_{syn} gradually gets close to Acc_{real}. In particular, with $\epsilon = 8$, the reduction of Acc_{syn} is less than 1% for the **Census** and **Adult** dataset and less than 3% for the other two datasets. The results indicate that the synthetic data generated by our framework largely preserves the joint distributions and correlations of real data, and can replace real data for AI training tasks.

Extension to Other Data Types Besides collecting high-dimensional categorical data, it is also possible to modify the current WAE model to support other data types, such as image data and text data. Despite the variation of the model structures, the main idea of training the model under privacy-preserving federated learning and generating synthetic data remains unchanged. In Figure 4, we give the synthesis results evaluated on the **MNIST** (LeCun et al. (2010)), **Fashion-MNIST** (Xiao et al. (2017)), and **CelebA** (Liu et al. (2015)) datasets using WAE models with convolution layers. For each dataset, we show the synthetic data produced by generated models trained under different privacy settings. Note that the synthetic data are randomly generated and may look different from real data. It can be observed that our framework is capable of synthesizing image datasets, and generated images have better quality with an increase of ϵ.

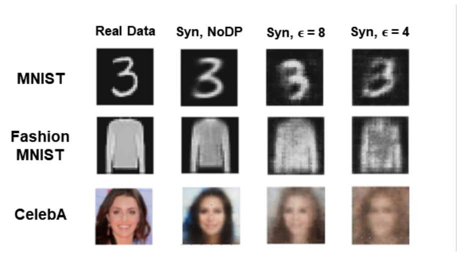


Figure 4: Results of image data synthesis.

4 Conclusion

In this paper, we follow the idea of data synthesis and propose DP-Fed-Wae, a privacy-preserving framework for high-dimensional data collection. The framework utilizes a (generative) Wasserstein autoencoder to learn the joint distributions and correlations of high-dimensional user data and generate high-utility synthetic data on the server side. Moreover, we apply a differentially-private federated learning mechanism for training the autoencoder, which not only avoids the collection of real local data but also prevents privacy leakage of local data during the training process. Experimental evaluation with real-world datasets shows that our framework significantly outperforms the LDP-based baseline algorithms for high-dimensional data collection and synthesis. The synthetic data generated by our framework preserves very similar statistical properties as real data and can replace real data for data mining and model training tasks.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

Mohammad Alaggan, Sébastien Gambs, and Anne-Marie Kermarrec. BLIP: Non-interactive differentially-private similarity computation on Bloom filters. In Andréa W. Richa and Christian Scheideler (eds.), *Stabilization, Safety, and Security of Distributed Systems - 14th International Symposium*, volume 7596 of *Lecture Notes in Computer Science*, pp. 202–216, Toronto, Canada, 2012. Springer.

Mohammad Alaggan, Mathieu Cunche, and Sébastien Gambs. Privacy-preserving wi-fi analytics. *Proceedings on Privacy Enhancing Technologies*, 2018(2):4–26, 2018.

Sean Augenstein, H. Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, and Blaise Agüera y Arcas. Generative models for effective ML on private, decentralized datasets. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, virtual, 2020. OpenReview.net.

Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*, pp. 2288–2296, Long Beach, CA, USA, 2017. Curran Associates Inc.

Gabrielle Berman, Sara de la Rosa, and Tanya Accone. Ethical considerations when using geospatial technologies for evidence generation. Technical report, Innocenti Research Briefs, 2018.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.

Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pp. 435–447, New York, NY, USA, 2018. Association for Computing Machinery.

Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 1(8), 2017.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Marco F. Duarte and Yu Hen Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, 2004.

John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil Vadhan. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, pp. 381–390, Bethesda, MD, USA, 2009. ACM.

Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1054–1067, Scottsdale, AZ, USA, 2014. ACM.

Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. Building a Rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies*, 3:1–21, 2016.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients – How easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, virtual, 2020. Curran Associates Inc.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, Montreal, Quebec, Canada, 2014. Curran Associates Inc.

Markus Herdin, Nicolai Czink, Hüseyin Özcelik, and Ernst Bonek. Correlation matrix distance, a meaningful measure for evaluation of non-stationary MIMO channels. In *2005 IEEE 61st Vehicular Technology Conference*, volume 1, pp. 136–140, Stockholm, Sweden, 2005. IEEE.

Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, Jan 2011. ISSN 1095-7111. doi: 10.1137/090756090. URL `http://dx.doi.org/10.1137/090756090`.

François Kawala, Ahlame Douzal-Chouakria, Eric Gaussier, and Eustache Dimert. Prédictions d'activité dans les réseaux sociaux en ligne. In *4ième conférence sur les modèles et l'analyse des réseaux : Approches mathématiques et informatiques*, pp. 16, France, 2013.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, Banff, AB, Canada, 2014. OpenReview.net.

Ron Kohavi. Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid. In *Proceedings of the SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 202–207, Portland, Oregon, USA, 1996. AAAI Press.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Haoran Li, Li Xiong, and Xiaoqian Jiang. Differentially private synthesization of multi-dimensional data using copula functions. In *Advances in Database Technology: Proceedings of the International Conference on Extending Database Technology*, volume 2014, pp. 475–486, Athens, Greece, 2014. NIH Public Access, OpenProceedings.org.

Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. Fedsel: Federated SGD under local differential privacy with top-k dimension selection. In *Database Systems for Advanced Applications - 25th International Conference, DASFAA 2020, Jeju, South Korea, September 24-27, 2020, Proceedings, Part I*, volume 12112 of *Lecture Notes in Computer Science*, pp. 485–501. Springer, 2020.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *2015 IEEE International Conference on Computer Vision*, pp. 3730–3738, Santiago, Chile, 2015. IEEE Computer Society.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282, Fort Lauderdale, FL, USA, 2017. PMLR.

Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *Proceedings of the International Conference on Learning Representations*, Vancouver, BC, Canada, 2018. OpenReview.net. URL `https://openreview.net/pdf?id=BJ0hF1Z0b`.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (S&P)*, pp. 739–753, San Francisco, CA, USA,, 2019. IEEE.

Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, 2018.

John C. Platt. *Fast Training of Support Vector Machines Using Sequential Minimal Optimization*, pp. 185208. MIT Press, Cambridge, MA, USA, 1999. ISBN 0262194163.

Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 192–203, Vienna, Austria, 2016. ACM.

Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A McCann, and S Yu Philip. LoPub: High-dimensional crowdsourced data publication with local differential privacy. *IEEE Transactions on Information Forensics and Security*, 13(9):2151–2166, 2018.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. Technical report, University of California, San Diego; Institute for Cognitive Science, 1985.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. Wasserstein autoencoders. In *International Conference on Learning Representations (ICLR 2018)*, Vancouver, BC, Canada, 2018. OpenReview.net.

Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. DP-CGAN: Differentially private synthetic data and label generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition – Workshops*, pp. 98–104, Long Beach, CA, USA, 2019. Computer Vision Foundation / IEEE.

Aleksei Triastcyn and Boi Faltings. Federated generative privacy. *IEEE Intelligent Systems*, 35(4): 50–57, 2020.

Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. Collecting and analyzing multidimensional data with local differential privacy. In *Proceedings of the 35th IEEE International Conference on Data Engineering*, pp. 638–649, 2019a.

Teng Wang, Xinyu Yang, Xuebin Ren, Wei Yu, and Shusen Yang. Locally private high-dimensional crowdsourced data release based on copula functions. *IEEE Transactions on Services Computing*, pp. 1–1, 2019b.

Tianhao Wang, Ninghui Li, and Somesh Jha. Locally differentially private frequent itemset mining. In *2018 IEEE Symposium on Security and Privacy (S&P)*, pp. 127–143, San Francisco, California, USA, 2018. IEEE Computer Society.

Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE Conference on Computer Communications (INFOCOM)*, pp. 2512–2520, Paris, France, 2019c. IEEE.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL `http://arxiv.org/abs/1708.07747`.

Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via Bayesian networks. *ACM Transactions on Database Systems*, 42(4):25:1–25:41, 2017.

A Preliminaries

A.1 Differential Privacy

Differential Privacy (DP) Dwork & Roth (2014) is a state-of-the-art data anonymization technique which provides strong privacy guarantees for data analysis. The mathematical definition of DP is as follows:

Definition 1 (Differential Privacy Dwork & Roth (2014))**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{O}$ satisfies ϵ-DP if for any two adjacent datasets D, D' differing from one data sample and for any measurable subset of outputs $\mathcal{Y} \subseteq \mathcal{O}$ we have:*

$$\Pr\left[\mathcal{A}(D) \in \mathcal{Y}\right] \leq e^{\epsilon} \cdot \Pr\left[\mathcal{A}(D') \in \mathcal{Y}\right] \tag{1}$$

where ϵ describes the privacy loss.

The Definition 1 is usually applied in centralized settings where the data have already been collected by a trusted server. However, in the local settings, we aim to ensure that each client's local data will not be accessed to the server. Thus, the definition of *local differential privacy* (LDP) has been proposed Kasiviswanathan et al. (2011), which provides strong local privacy guarantees for each client. The definition is as follows:

Definition 2 (Local Differential Privacy Kasiviswanathan et al. (2011))**.** *A randomized mechanism $\mathcal{A} : \mathcal{D} \to \mathcal{O}$ satisfies ϵ-LDP if and only if for any two inputs $x, x' \in \mathcal{D}$ and for any output $y \in \mathcal{O}$ we have:*

$$\Pr\left[\mathcal{A}(x) = y\right] \leq e^{\epsilon} \cdot \Pr\left[\mathcal{A}(x') = y\right] \tag{2}$$

where ϵ describes the privacy loss.

In addition, LDP also holds two widely-used properties Dwork & Roth (2014), namely *Sequential Composition* and *Robustness to Post-Processing*. The former property states that interactively applying the LDP mechanism on the same set of data yields an accumulated privacy cost. The latter property states that any deterministic or randomized function defined over an LDP mechanism also satisfies LDP.

Property 1 (Sequential Composition)**.** *Suppose n mechanisms $\{\mathcal{A}_1, \cdots, \mathcal{A}_n\}$ respectively satisfy ϵ_i-LDP, and are sequentially computed on the same set of private data D, then a mechanism formed by $(\mathcal{A}_1, \cdots, \mathcal{A}_n)$ satisfies $(\sum_{i=1}^{n} \epsilon_i)$-LDP.*

Property 2 (Robustness to Post-Processing)**.** *Let $\mathcal{A}$ be an ϵ-LDP mechanism and g be an arbitrary mapping from the set of possible outputs to an arbitrary set. Then, $g \circ \mathcal{A}$ is ϵ-LDP.*

A.2 Federated Learning

Federated learning McMahan et al. (2017) (FL) is a decentralized learning mechanism which achieves computational efficiency and privacy benefits by distributing the training task to local devices. At each global round, the server distributes the current global model to a number of local clients. Each client locally updates the global model and returns the model update to the server. On the server side, all the local updates are aggregated to update the global model, which will be distributed in the next global round. Since only model parameters are exchanged during the training process, FL allows the model trained without accessing raw local data.

Although FL provides enhanced privacy protection in comparison to centralized training, recent contributions (*e.g.*, Wang et al. (2019c); Nasr et al. (2019); Geiping et al. (2020)) point out that the mechanism still has privacy risks. In the context of FL, privacy risks can be mainly divided into *local privacy* and *global privacy* aspects. *Local privacy* risks appear when the local updates reveal insights about local data, while *global privacy* risks represent situations when the global model memorizes local data. Motivated by this, privacy enhancing techniques such as secure multi-party computation (MPC) Bonawitz et al. (2017) or differential privacy (DP) McMahan et al. (2018) are incorporated into the FL mechanism, providing protections against global and local privacy risks.

A.3 AUTOENCODERS

An autoencoder Rumelhart et al. (1985) is a type of neural network that is used to learn efficient and compressed feature representations in an unsupervised manner. The model is constituted of two main parts: an *encoder* Q_ϕ and a *decoder* G_θ. The encoder compresses the original high-dimensional input $x \sim P_x$ into the low-dimensional latent feature $z = Q_\phi(x)$ and the decoder maps z to the reconstructed output $x' = G_\theta(z)$, which is of the same shape as x. The goal of training is to find an optimized pair of encoder and decoder, which minimize the distance between x and $x' = G_\theta(Q_\phi(x))$, namely

$$\mathcal{L}_{AE} = \mathbb{E}_{x\sim P_x}[c(x, G_\theta(Q_\phi(x))] \tag{3}$$

where $c(\cdot,\cdot)$ is a metric for featuring the difference between two vectors. Commonly, we can use the mean squared error (MSE) to measure the distance between numerical input vectors and the cross entropy (CE) for binary input vectors.

B DETAILS OF THE PROPOSED FRAMEWORK

B.1 DESIGN OF THE GENERATIVE MODEL

We choose the Wasserstein Autoencoder (WAE) as the generative model in our framework, which provides better data synthesis capability in comparison to the Variational Autoencoder (VAE) Kingma & Welling (2014) and less training difficulty than the Generative Adversarial Network (GAN) Goodfellow et al. (2014). As a variant from the family of autoencoders, WAE preserves the encoder-decoder architecture. The encoder Q_ϕ compresses the original high-dimensional input $x \sim P_x$ into the low-dimensional latent feature $z = Q_\phi(x)$ and the decoder G_θ maps z to the reconstructed output $x' = G_\theta(z)$, which is of the same shape as x. The distance between the original input and the reconstructed output can be presented as $c(x, G_\theta(Q_\phi(x))$. In addition, a regularizer term $D_z(q_z, p_z)$ is applied to measure the distance between the latent space distribution q_z and certain prior distribution p_z. The final objective function of the WAE model can thus be formulated as below:

$$\mathcal{L}_{WAE} = \mathbb{E}_{x\sim P_x}[c(x, G_\theta(Q_\phi(x))] + \lambda \cdot D_z(q_z, p_z) \tag{4}$$

where λ is a hyperparameter for balancing the two terms. The goal of training is to find an optimal set of parameters for the encoder and decoder, which minimizes the distance between the inputs and outputs while restricting the latent distribution to follow the prior distribution.

We designed the WAE models with fully-connected hidden layers. We apply the *relu* activation on the output of each hidden layer for better training performance. Moreover, since the inputs are binary vectors, we use the *sigmoid* activation on the output layer, which restricts the output value within [0,1]. We use the cross entropy (CE) to measure the reconstruction cost $c(x, G(z))$ and the maximum mean discrepancy (MMD) to measure the latent space distance $D_z(q_z, p_z)$, where p_z follows the standard Gaussian distribution.

B.2 TRAINING THE GENERATIVE MODEL

B.2.1 LOCAL RANDOMIZATION WITH SIGNDS

Previous LDP-FL frameworks (*e.g.*, Dwork & Roth (2014); Duchi et al. (2018); Wang et al. (2019a)) evenly split the privacy budget across dimensions and apply the perturbation independently. However, the per-dimension privacy budget becomes extremely small for high-dimensional models, which results in a significant increase of noise. A recent work Liu et al. (2020) proposed a *two-stage* LDP-FL framework, which splits the privacy budget into a *dimension selection* (DS) stage and a *value perturbation* (VP) stage. In the DS stage, the local update is sorted by *absolute* value and one "important" dimension is privately selected from the top-k dimensions; in the VP stage, the value of

Algorithm 1: SIGNDS

```
Input: $\Delta \in \mathbb{R}^d$: local update; $k$: the number of parameters in the top-$k$ set; $\epsilon$: privacy budget; $s$:
       sampled sign
Output: $j$: selected dimension index
 1: Initialize top-$k$ status vector $z = \{0\}^d$
 2: if $s$=1 then
 3:     Sort $\Delta$ by real value and select the top-$k$
        largest values to construct set $\mathcal{S}_{topk}$
 4: else
 5:     Sort $\Delta$ by real value and select the top-$k$ smallest values to construct set $\mathcal{S}_{topk}$
 6: end if
 7: For $j \in \{1, \cdots, d\}$, if $\Delta_j \in \mathcal{S}_{topk}$, set $z_j = 1$.
 8: Sample a Bernoulli variable $x$ such that
        $$\Pr[x = 1] = \frac{e^\epsilon \cdot k}{d - k + e^\epsilon \cdot k} \quad (6)$$
 9: if $x = 1$ then
10:     Randomly sample index $j \in \{j \in \{1, \cdots, d\} | z_j = 1\}$
11: else
12:     Randomly sample index $j \in \{j \in \{1, \cdots, d\} | z_j = 0\}$
13: end if
14: Return $j$
```

the selected dimension is perturbed. Finally, a sparse local update is constructed and returned to the server. Although Liu et al. (2020) mitigates the dimension-dependency problem by only selecting one "important" dimension, the privacy budget is still consumed by the two stages. In high-privacy scenarios (where the privacy budget is small), each stage may therefore obtain only an insufficient privacy budget and cause large randomness.

Motivated by the limitations of Liu et al. (2020), we propose a *signed dimension selection* algorithm SIGNDS, as presented in Algorithm 1. The main idea is to substitute the VP stage by assigning a constant value to the selected dimension. Since the parameter values may have different signs, we introduce an extra input variable $s \in \{-1, 1\}$, which is randomly sampled by the server. Given an input local update $\Delta \in \mathbb{R}$, we first initialize a binary top-k status vector $z = \{0\}^d$ (line 1). Then, instead of sorting by absolute value as in Liu et al. (2020), we sort Δ by *original* value and construct the top-k set $\mathcal{S}_{topk}$ based on s: if $s = 1$, we select the top-k *largest* values to build $\mathcal{S}_{topk}$; otherwise, we select the top-k *smallest* values. For any dimension $j \in \{1, \cdots, d\}$, if Δ_j is in the top-k set $\mathcal{S}_{topk}$, we set $z_j = 1$ (line 7). We refer to these dimensions as *top-k dimensions* and to others as *non-top-k dimensions*. Then, a dimension index j is randomly sampled as follows:

$$j \in \begin{cases} \{j \in \{1, \cdots, d\} | z_j = 1\} & w.p. \quad p \\ \{j \in \{1, \cdots, d\} | z_j = 0\} & w.p. \quad 1 - p \end{cases} \tag{5}$$

Namely, the index j is sampled from the *top-k dimensions* with a probability p and otherwise from the *non-top-k dimensions*. We refer to p as the *top-k probability*. Finally, the dimension index j is returned to the server. At this stage, our proposed algorithm only returns the dimension index. As such, we save the privacy budget for the value perturbation in Liu et al. (2020). With the same privacy level, we can now achieve less randomness and thus higher accuracy in dimension selection.

In the following, we provide the privacy guarantee and utility analysis of Algorithm 1.

Lemma 1. *Algorithm 1 satisfies ϵ-LDP when $p \leq \frac{e^\epsilon \cdot k}{d-k+e^\epsilon \cdot k}$.*

Proof. For each client, given the sampled sign s and any output dimension index $j \in \{1, \cdots, d\}$, let $z, z' \in \{0, 1\}^d$ be the top-k status vector of any two possible local update vectors x and x'. When $p \leq \frac{e^\epsilon \cdot k}{d-k+e^\epsilon \cdot k}$ we have:

$$\frac{\Pr[j|x]}{\Pr[j|x']} = \frac{\Pr[j|z] \cdot \Pr[z|x]}{\Pr[j|z'] \cdot \Pr[z'|x']} = \frac{\Pr[j|z]}{\Pr[j|z']} \leq \frac{\Pr[j|z_j = 1]}{\Pr[j|z'_j = 0]} = \frac{p \cdot \frac{1}{k}}{(1-p) \cdot \frac{1}{d-k}} \leq e^\epsilon \tag{7}$$

Algorithm 2: Training of Generative Model

```
Input: $M_1 \in \mathbb{R}^d$: initial global model; $n$: number of per-round clients; $E$: number of local epochs;
       $\eta$: local learning rate; $k$: number of parameters in the top-$k$ set of each local update; $T$:
       number of global aggregation rounds; $\gamma$: global learning rate; $\epsilon_r$: per-round privacy budget
Output: Trained WAE model $M$

Server executes:
 1: for global round $t = 1, \cdots, T$ do
 2:     Randomly select a group of $n$ clients
 3:     for client $i = 1, \cdots, n$ in parallel do
 4:         Randomly sample a sign $s_t^i \in \{1, -1\}$ with probability Pr$[s_t^i = 1] = 0.5$
 5:         Broadcast current global model $M_t$ and the sampled sign $s_t^i$ to the client
 6:         Receive dimension index from client $j_t^i$ = LocalUpdate$(M_t, E, \eta, \epsilon_r, k, s_t^i)$
 7:         Build sparse local update $\hat{\Delta}_t^i = \{0\}^d$ and set $\hat{\Delta}_t^i[j_t^i] = s_t^i$
 8:     end for
 9:     Aggregate local updates: $\hat{\Delta}_t = \frac{1}{n}\sum_{i=1}^{n} \hat{\Delta}_t^i$
10:     Update global model $M_{t+1} = M_t + \gamma \cdot \hat{\Delta}_t$
11: end for
12: Return Global model $M = M_{T+1}$

LocalUpdate($M_t, E, \eta, \epsilon_r, k, s_t^i$):
// Run on the client side
13: Initialize local model $M_t^i \leftarrow M_t$
14: for epoch $e = 1, \cdots, E$ do
15:     $M_t^i = M_t^i - \eta \cdot \nabla\mathcal{L}(M_t^i, X^i)$
16: end for
17: Calculate local update: $\Delta_t^i = M_t^i - M_t$
18: Select dimension $j_t^i$ = SignDS($\Delta_t^i, k, \epsilon_r, s_t^i$)
19: Return $j_t^i$
```

□

In addition to the privacy guarantee, we are also interested in how to choose proper k and ϵ in order to achieve a certain top-k probability p. Let $\alpha = k/d$ be the ratio of the top-k parameters regarding the total number of parameters. An $\alpha = 1$ means to randomly select one dimension from the entire dimension group. Intuitively, the smaller α, the closer the parameter values of top-k dimensions to the real largest (or smallest) value and the better the model utility. We derive relations among ϵ, p and α as follows:

Corollary 1. *Given privacy budget ϵ, to achieve a probability p, α should satisfy $\alpha \geq \frac{p}{e^\epsilon \cdot (1-p) + p}$*

Corollary 2. *Given fixed top-k ratio α, to achieve a probability p, ϵ should satisfy $\epsilon \geq \log \frac{p \cdot (1-\alpha)}{(1-p)\cdot\alpha}$*

Proof. From Lemma 1, we have:

$$p \leq \frac{e^\epsilon \cdot k}{d - k + e^\epsilon \cdot k} = \frac{e^\epsilon \cdot \alpha}{1 - \alpha + e^\epsilon \cdot \alpha} \tag{8}$$

thus, with a fixed ϵ, we have: $\alpha \geq \frac{p}{e^\epsilon \cdot (1-p) + p}$; with a fixed α, we have: $\epsilon \geq \log \frac{p\cdot(1-\alpha)}{(1-p)\cdot\alpha}$ □

Corollary 1 states that with a fixed privacy budget ϵ, a smaller top-k ratio α leads to a decrease of top-k probability p. Namely, we have to choose a large α in order to ensure the index is more likely to be sampled from top-k dimensions. As ϵ increases, α does not differ much regarding p. In other words, we can always achieve a high top-k probability even with a small top-k ratio. Moreover, given an expected top-k probability p and a predefined top-k ratio α, the minimum required privacy budget ϵ can be calculated using Corollary 2.

B.2.2 Overall Training Process

We now describe the overall training process presented in Algorithm 2. At each global round t, the server selects a group of n clients. For each client i, the server randomly samples a sign $s_t^i \in \{-1, 1\}$ with equal probability (line 4) and sends s_t^i together with the current global model M_t to the client (line 5). On the local side, the client i in the group trains the global model for several gradient descent epochs with his local data X^i (line 14-16) and computes the local update Δ_t^i (line 17). Given the sampled sign s_t^i and the predefined privacy budget ϵ_r, the SignDS algorithm is applied on Δ_t^i which privately selects a dimension index j_t^i and returns to the server (line 18). As discussed in Lemma 1, the SignDS algorithm satisfies ϵ_r-LDP. After receiving the index j_t^i, the server builds a sparse local update $\hat{\Delta}_t^i$ and assign s_t^i to the selected dimension (line 7). Finally, the server aggregates all the sparse local updates (line 9) and updates the global model with a global learning rate γ (line 10). The updated global model M_{t+1} is distributed again to local clients to start the next round.

It should be noted that according to Property 1, if the same client repeatedly participates in the training and submits the local update for multiple global rounds, the overall privacy guarantee for the local data will be accumulated. Assume each client is allowed to participate in at most t_r global rounds. In order to ensure an overall privacy guarantee of ϵ-LDP for each client's local data after the whole training process, the per-round privacy guarantee should satisfy $\epsilon_r \leq \epsilon / t_r$. Moreover, during the training process, we monitor participating rounds of each client. If a client has reached the maximum participating rounds (which is t_r here), he is not allowed to participate in the later training process.

C Experiment Settings

C.1 Baseline Method

In the following experiments, we use LoPub Ren et al. (2018) and LoCop Wang et al. (2019b) as our baseline algorithms. In comparison to our data synthesis-based solution, both algorithms apply LDP directly on the *local data* and send the randomized data to the server. The local randomization follows the RAPPOR algorithm which firsly hash the local data into Bloom filter strings and then randomly flip the bits according to the randomized technique. Given d as the dimension of local data, h as the number of hash functions and f as the flip probability, the overall privacy for each individual client is:

$$\epsilon = 2 \cdot d \cdot h \cdot \ln((2 - f)/f) \tag{9}$$

Then, the randomized data will be aggregated on the server side for estimating the joint distributions and attribute dependencies and such information will then be finally used for constructing the synthetic dataset: LoPub generates the dependency graph based on a dependence threshold ϕ and estimates k-way joint distributions to generate the synthetic data; LoCop leverages multivariate Gaussian copula to determine attribute dependencies and generates synthetic data by only using 1- and 2-way joint distributions. For both algorithms, we use the Lasso-based regression for estimating the joint distributions. In addition, we follow Ren et al. (2018) to choose the number of hash function $h = 4$ and the dependence threshold $\phi = 0.4$.

C.2 Details of Datasets and Models

We use four open-source datasets for evaluating the performance of our framework. Each dataset contains multi-dimensional data records, which are used for classification tasks:

- The **Census** dataset Dua & Graff (2017) contains records drawn from the 1990 United States census data, which include 68 personal attributes such as gender, income and marriage status. We use the dataset for a classification task to determine the duration of people's active duty service.
- The **Twitter** dataset Kawala et al. (2013) contains records with 77 attributes such as the number of discussions, average discussion length, and the number of authors, which are used to predict the number of active discussions, namely the popularity magnitude of each instance. In our experiment, we quantify the values of each attribute into 5 bins. The goal is to classify the level of popularity of each instance.

Table 1: Datasets details

Dataset	**Type**	**Num. Records**	**Num. Attributes**	**Domain Size**
Census	Integer	2458285	69	2^{150}
Twitter	Integer	140707	78	2^{181}
Vehicle	Binary	98528	101	2^{101}
Adult	Binary	32561	124	2^{124}

Table 2: Structure of WAE models

Dataset	**Num.Params**	**Model Structure**
Census	76524	Input-Dense(96, relu)-Dense(24) -Dense(96, relu)-Output(sigmoid)
Twitter	94961	Input-Dense(128, relu)-Dense(36) -Dense(128, relu)-Output(sigmoid)
Vehicle	13093	Input-Dense(64, relu)-Dense(16) -Dense(64, relu)-Output(sigmoid)
Adult	16060	

- The **Vehicle** dataset Duarte & Hu (2004) contains data collected in wireless distributed sensor networks. Each record has 100 attributes representing data collected from different acoustic and seismic sensors. The goal is to train a classifier for vehicle type classification.
- The original **Adult** dataset Kohavi (1996) contains records with 15 personal attributes such as age, occupation, education and gender. The goal is to train a binary classifier which determines whether a person earns more than 50K a year. We use the processed version from Platt (1999) which converts the original attributes into 123 binary features.

We present details of each dataset in Table 1, which include the number of records and attributes, the length of the *one-hot* encoded input and the total domain size. Since the number of local data should be large in order to preserve data utility (which will be discussed in Appendix E.1), in the following experiments, we simulate the large-scale distributed scenario by assuming there are 5×10^4 clients, each holding 2 data records. Hence, we randomly sample 10^5 records for each dataset. For datasets with more than 10^5 records (*i.e.*, **Census** and **Twitter**), we do the sampling *without* replacement.

We vary the structure of WAE models to fit the input size of different datasets. Details of the WAE models can be found in Table 2.

C.3 Parameter Configurations

In the experiments, we assume there are 5×10^4 clients. We set the global round $T = 5000$, and $n = 10$ clients are sampled to train the WAE model in each global round; namely, each client is sampled once during the whole training process. We set the global learning rate $\gamma = 1$. For local training, each client updates the model for $E = 10$ epochs with a learning rate $\eta = 0.001$. For the local randomization, we choose the top-k ratio α from $\{0.05, 0.1, 0.25\}$. Moreover, we vary the privacy parameters ϵ in order to explore the influence of privacy on the performance of the framework. In our experiments, we choose $\epsilon \in \{1, 2, 4, 6, 8\}$.

It should be noted that ϵ here is the *overall* local privacy budget for each client. As mentioned in Section 2, if each client participates in t_r global training rounds, the per-round privacy budget should satisfy $\epsilon_r \leq \epsilon / t_r$. Since we assume that each client only participates once during the whole training process, we have $t_r = 1$ and the per-round privacy budget is equal to the overall privacy budget. Moreover, we would like to emphasize that the selected ϵs are *reasonable* local privacy guarantees for collecting d-dimensional data. Consider the privacy guarantee of the baseline algorithms (Equation (9)), with the number of hash function $h = 1$ and a flipping probability $f = 0.5$, we already have $\epsilon = 150$ for the **Census** dataset with $d = 68$. For the **Adult** dataset with $d = 124$, the overall ϵ is even 272, which is significantly larger than our setting.

Table 3: Computation time of model training and synthetic data generation

Dataset		Adult	Vehicle	Census	Twitter
Training	**Client**	1.06 s	0.93 s	1.28 s	1.25 s
	Server	3.51 ms	3.05 ms	4.95 ms	4.92 ms
Data Generation		7.38 s	7.37 s	9.44 s	10.47 s

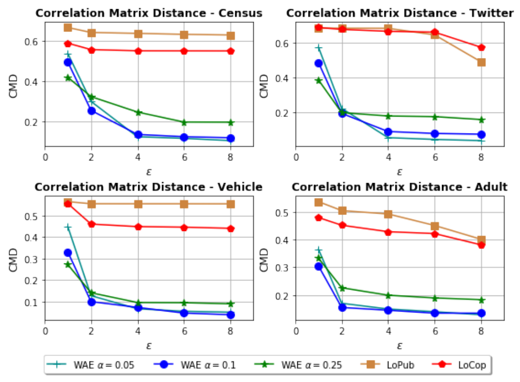


Figure 5: Averaged correlation error (CMD) between the real and synthetic data with different privacy levels.

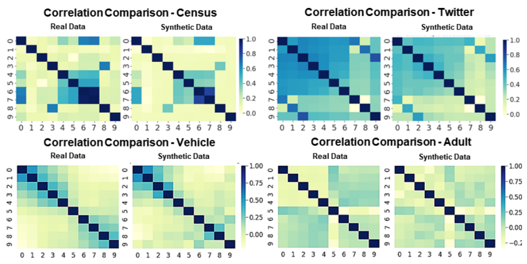


Figure 6: Correlation comparison between the real and synthetic data with $\epsilon = 8$ and $\alpha = 0.1$. For each dataset, we present the correlations of the first 10 attributes. It can be seen that the synthetic data preserves similar correlations as real data.

C.4 Computation Environments

We implement the proposed DP-Fed-Wae framework based on Tensorflow and perform all the experiments on a server with Intel E5-2470 2.40GHz CPU. In Table 3, we report the computational time of: 1) 10 epochs of local training on each client side; 2) one round of local updates aggregation and global model update on the server side; 3) generation of 10^5 synthetic records on the server side.

D Experiment Results

D.1 Comparison of Correlation

For the comparison of correlation, we respectively compute the Pearson correlation coefficient of the real and synthetic dataset and use the Correlation Matrix Distance (CMD) Herdin et al. (2005) to measure the distance between the two correlations, which is defined as follows:

$$CMD = 1 - \frac{\text{tr}\{R_{real}R_{syn}\}}{\|R_{real}\|_2\|R_{syn}\|_2} \tag{10}$$

where R_{real} and R_{syn} are correlation coefficient matrices of real and synthetic data, $\text{tr}(\cdot)$ is the matrix trace, $\|\cdot\|_2$ is the Frobenius norm. The CMD is bounded by $[0, 1]$, where zero means the two correlation matrices are identical.

For each dataset, we calculate the CMD of the synthetic data generated by both the baseline algorithms and our framework under different privacy levels and compare the results in Figure 5. It can be seen that with the same ϵ, the baseline algorithms (referred to as *LoPub* and *LoCop*) always show a much larger CMD in comparison to the results of our framework (referred to as *WAE*). Although increasing the ϵ helps to reduce the CMD, it is still insufficient for preserving the multivariate correlations of real data. On the other hand, the synthetic data generated by our framework shows a distinctive decrease with the increase of ϵ. In particular, the CMD is close to zero when $\epsilon \geq 5$, indicating that the synthetic data have similar cross-attribute correlations as real data.

We further visualize the correlation coefficient matrix of real data and synthetic data with heat maps in order to better understand the capability of our method in capturing and preserving the cross-attribute correlations. Figure 6 shows the comparison result of the different datasets with $\epsilon = 8$ and

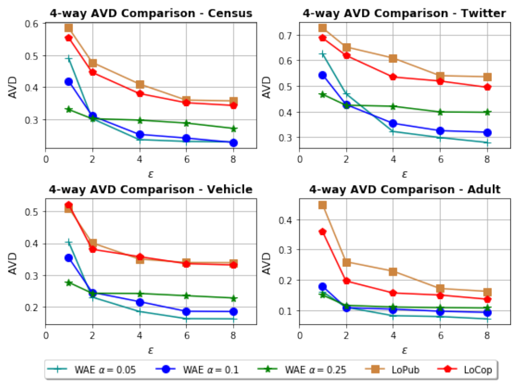


Figure 7: Average total variation distance (AVD) of 4-way joint distribution between the real and synthetic data generated by our framework (*WAE*) as well as by the baseline algorithms (*LoPub*, *LoCop*) under different privacy levels.

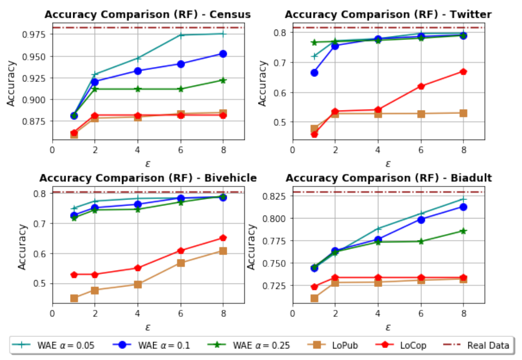


Figure 8: Classification accuracy of random forest (RF) trained with real data (*Real Data*) and synthetic data generated by our framework (*WAE*) as well as by the baseline algorithms (*LoPub*, *LoCop*) under different privacy levels.

$\alpha = 0.1$. For each dataset, we present the correlations of the first 10 attributes. From the results, it can be seen that the correlation of synthetic data is similar to the correlation of real data, indicating that the synthetic data successfully preserves the attribute correlations of real data.

D.2 Comparison of Joint Distributions (Extended)

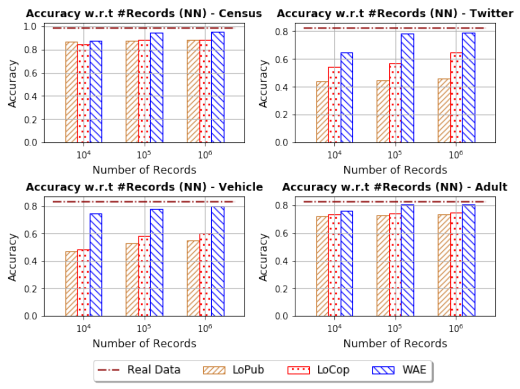
We analyze the AVD of all three algorithms with respect to the privacy level ϵ. For each dataset, we respectively compare the AVD of the synthetic data generated by the baseline algorithms and by our framework. In Figure 7, we present the results for the 4-way joint distribution with different privacy budgets. It can be seen that the AVD of all the algorithms decreases with the increase of ϵ. For all datasets, the synthetic data generated by our framework (referred to as *WAE*) constantly have smaller AVD in comparison to the baseline methods, indicating that the synthetic data generated by our framework preserves better multivariate distributions than the baseline methods. Also, we notice that the non-binary datasets *Census* and *Twitter* usually show larger AVD in comparison to the other two binary datasets. This is due to the fact that the non-binary datasets have a larger domain size, which leads to lower frequencies of the potential attribute combinations. Therefore, it is more difficult for the generative models to find meaningful mappings between the original input space and the compact latent space, which results in a comparatively larger difference between the synthetic data and real data.

D.3 AI Training Performance (Extended)

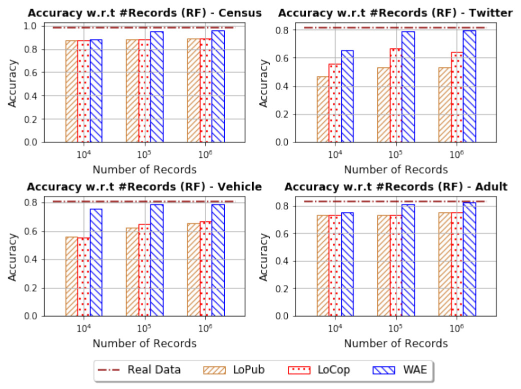
In Figure 8, we present the evaluation results on random forest (RF) under different privacy levels. Similary as in Figure 3, the Acc_{syn} of our method shows distinctive outperformance in comparison to the baseline algorithms, which indicates that the synthetic data generated by our framework preserves better utility for AI training tasks.

D.4 Impact of the Number of Records

In the above experiments, we assume a group size of 5×10^4 clients and in total 10^5 records. We further investigate how the number of records impacts the utility of synthetic data. We vary the number of records among $\{10^4, 10^5, 10^6\}$. Similar to previous experiments, we assume that each client holds two data records and only participates once during the whole training process. For the experiments with 10^4, we set the total global rounds $T = 500$ with $n = 10$ clients for each round. For the experiments with 10^6 records, we set the total global rounds $T = 5000$ with $n = 100$ clients for each round.



(a) Accuracy of neural networks



(b) Accuracy of random forests

Figure 9: Classification accuracy of (a) neural network (NN) and (b) random forest (RF) with different number of records under the privacy level of $\epsilon = 8$.

We evaluate the accuracy of both classification models (*i.e.*, 2-layer NN and RF) with respect to the number of records and present the results in Figure 9a and Figure 9b. Here we compare the results under a privacy of $\epsilon = 8$ (and $\alpha = 0.1$ for our method). Although both algorithms show higher classification accuracy with a larger number of records, the baseline algorithms still cannot achieve significant improvements even with the largest number of records. In comparison, the classification accuracy of our method constantly outperforms the baseline algorithms. In addition, we notice that the classification accuracy in the experiments with 10^4 records is distinctively lower than others. This is because the generative model is underfitted when trained on a limited number of records and thus cannot generate high-utility synthetic data. On the other hand, although a larger number of local data (*e.g.*, 10^6) ensures the generative model be fully-trained, the model performance does not improve much after achieving convergence and thus cannot reach much improvement regarding classification accuracy.

D.5 Auxiliary Data for Model Design and Pre-training

Before applying the WAE model for collecting clients' local data, the server needs to design the model structure. An appropriate model structure helps to enhance the capability of capturing the local data distributions and thus the utility of synthetic data. In our scenario, the server only knows the *basic properties* of the data to be collected, such as the number of attributes and the domain of each attribute. The server can thus use some auxiliary data to optimize the model structure. The auxiliary data here refer to certain public datasets or randomly generated data that have the same basic properties as local data. The server can use such data to simulate the data collection process and tune the model structure by evaluating the utility of the synthetic data. Moreover, the auxiliary data can also be used for pre-training the WAE model before applying the model in the data collection process, so as to improve the model convergence and the performance of synthetic data generation.

In Table 4, we compare the utility of synthetic data generated by WAE models with (w) and without (w/o) pre-training under the setting of $\alpha = 0.1$ and $\epsilon \in \{4, 8\}$. For each dataset, we randomly generate an auxiliary dataset only using the *basic properties* of real data, as mentioned before. We use the auxiliary dataset to pre-train the WAE model and apply the pre-trained model to the data collection process. We respectively evaluate the utility of synthetic data generated in both scenarios based on the classification accuracy of NNs and RFs. For both types of models, we observe that the synthetic data generated by pre-trained WAE models achieve $1 \sim 2\%$ increase in classification accuracy. The results demonstrate that using auxiliary data to pre-train the WAE model is feasible to enhance the model convergence in the data collection process and further improve the synthetic data utility.

Table 4: Classification accuracy of synthetic data generated by WAE models with (w) and without (w/o) pre-training

		Accuracy - NN		**Accuracy - RF**	
Dataset		**w/o**	**w**	**w/o**	**w**
Census	$\epsilon = 8$	0.948	0.960	0.952	0.965
	$\epsilon = 4$	0.929	0.935	0.932	0.940
Twitter	$\epsilon = 8$	0.786	0.798	0.788	0.796
	$\epsilon = 4$	0.771	0.782	0.778	0.785
Vehicle	$\epsilon = 8$	0.781	0.795	0.788	0.794
	$\epsilon = 4$	0.762	0.789	0.763	0.782
Adult	$\epsilon = 8$	0.808	0.815	0.812	0.820
	$\epsilon = 4$	0.787	0.798	0.775	0.789

E Related Work

E.1 Data Collection under LDP

Differential privacy (DP) Dwork & Roth (2014), as a strong mathematical formalization of privacy, has been used as a criterion for privacy protection in data publishing, data analysis, and machine learning (Dwork et al. (2009); Zhang et al. (2017); Abadi et al. (2016)). However, these works assume a trusted server (data curator), who first collects the original local data, then performs data analysis and releases data under differential privacy. In order to eliminate the assumptions of trustworthy servers during the data collection process, local differential privacy (LDP) Kasiviswanathan et al. (2011) has been proposed, which provides strong privacy guarantees to local data. By utilizing local randomization algorithms, the server cannot infer any individual's original data but can learn the overall statistics of the whole population. However, prior research on LDP mainly focuses on collecting one-dimensional statistical information, such as frequency estimation (Erlingsson et al. (2014); Differential Privacy Team (2017)), heavy-hitter identification (Bassily et al. (2017); Bun et al. (2018)) and itemset mining (Qin et al. (2016); Wang et al. (2018)). Regarding scenarios with multi-dimensional data, Alaggan et al. Alaggan et al. (2012) propose to encode a clients profile data into private Bloom filters and to compute in a distributed fashion the similarity between clients under DP guarantees. A follow-up work Alaggan et al. (2018) applied the private mechanism to the analysis of aggregated statistics (*e.g.*, count of distinct elements) of stream data. However, both works do not involve the estimation of joint distributions and cross-attribute correlations, which differs from our objectives.

Actually, applying the above LDP-based algorithms to estimate complex statistics of high-dimensional data will cause extremely large communication overhead as well as a degradation in data utility. Consider, for example, RAPPOR Erlingsson et al. (2014), a state-of-the-art LDP-based data collection algorithm. For d-dimensional binary data with $d = 32$, we have domain size $|\Omega| = 2^{32} \approx 4.3 \times 10^{10}$. Directly applying RAPPOR consumes a communication cost and a storage space of $O(|\Omega|)$ Ren et al. (2018). Also, for high-dimensional input domains, it is not uncommon for each client to have a unique feature combination. Therefore, it is essential to collect a large number of data in order to cover all the possible combinations in the feature domain. Given a domain size Ω, as a general rule of thumb Erlingsson et al. (2014), the number of local data N should follow $\sqrt{N}/10 \geq |\Omega|$. In the above example, $N \geq 100 \cdot 2^{64} \approx 1.8 \times 10^{21}$. All of these requirements are impractical for real-world applications.

In subsequent research, Fanti et al. Fanti et al. (2016) proposed to separately collect data of each dimension under RAPPOR and estimate the joint distributions using expectation maximization (EM). Although the algorithm significantly reduces the communication overhead between clients and the server, it only supports to estimate the joint distribution of two attributes. Based on Fanti et al. (2016), Ren et al. proposed LoPub Ren et al. (2018), which reduces d-dimensional data to k-dimensional clusters ($k < d$) using dependence graphs and estimates k-way joint distributions with an EM-based and Lasso regression-based approach. However, the algorithm still suffers from high computational complexity and low data utility when k is large. An improved scheme, LoCop Wang et al. (2019b) was further proposed, which leverages multivariate Gaussian copula to estimate cross-attribute dependencies and to construct synthetic data.

Instead of directly randomizing the local data, our framework uses deep generative models to learn the data distributions and to generate synthetic data without accessing real data, which effectively enhances data utility. In our experiments, we use LoPub and LoCop as the baselines to compare the frameworks' performance in terms of data utility.

E.2 Differentially-private Synthetic Data Generation

Differentially-private synthetic data generation has been extensively studied over recent years as an alternative solution to privacy-preserving data publishing. Previous works (Li et al. (2014); Zhang et al. (2017); Ren et al. (2018)) analyzed statistical distributions of original data under differential privacy and used them to generate synthetic data. With the development of deep learning, later works propose to use differentially-private generative models (Park et al. (2018); Torkzadehmahani et al. (2019)) to directly generate privacy-preserving synthetic data. By training deep generative models under differential privacy definition, synthetic data can provide strong privacy protection on original data while preserving high data utility. However, these prior works only focus on the centralized setting, where the server has already collected the real local data and uses them to generate synthetic data for privacy-preserving data publishing. In contrast, our approach is practical for a distributed setting, where the server cannot directly collect the real data but is interested in learning statistical information about the data.

Recent works of Augenstein et al. (2020) and Triastcyn & Faltings (2020) also investigate the synthetic data generation under the federated setting. However, both studies either have a different focus from ours or have certain limitations. The work by Augenstein et al. Augenstein et al. (2020) proposed to use generative models (*e.g.*, GAN and recurrent neural networks) to detect errors and bugs in local data. However, as they claimed, such data examination applications are based on observing the performance change of generative models, which do not require high-fidelity generation. In comparison, our framework is able to generate synthetic data with high utility and fidelity, which can replace real data in data mining and AI training tasks. On the other hand, although Triastcyn et al. Triastcyn & Faltings (2020) focused on generating and publishing synthetic data, their method is only limited to image data. In addition, they adopted a weaker measure of privacy for preserving the model utility. In comparison, our framework can be applied for both structured and unstructured data. In this paper, we mainly focus on the collection of high-dimensional structured data (especially categorical data with high sparsity) and compare the performance with LDP-based algorithms. But in Section 3 we further provide evidence that our framework can also be used on complex datasets, such as image data. Moreover, we apply strict LDP randomization on the client side, which provides strong privacy guarantees for clients' local privacy.