Turkish Journal of Computer and Mathematics Education Vol.12 No. 10 (2021), 7165-7173
Research Article

Differentially Private Data Release: Bias Weight Perturbation Method - A Novel Approach

Dr. Priyank Jain[1], Dr. Manasi Gyanchandani[2], Shriya Sahu [3],Rohit Mishra[4]

Department of Information Technology[1], Department Of Computer Science and Engineering[2, 3, 4]
IIIT Bhopal[1], MANIT Bhopal[2, 3, 4]

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 28 April 2021

Abstract:

Differential privacy plays the important role to preserve the individual data. In this research work, discussing a novel approach of releasing private data to the public, which is differentially private, called Bias Weight Perturbation Method. The approach follow here align with principle of differential privacy, it also used concept of statistical distance and statistical sample similarity to quantify the synthetic data generation loss, which is then used to validate our results. Our proposed approach make use of the deep generative models for providing privacy and it further produce synthetic dataset which can be released to public for further use.

Keywords: Differential Privacy, Deep Learning, Synthetic Data

Introduction:

In today's world, as technology is emerging, data has become the prime source of fuel for these technologies to work. To do every type of statistical analysis and experiment we required a wide variety of data, most of the data is related to individual, government records, a private firm, etc. There are many methods available that are used to privatize the concerned data to protect the privacy threat involve,but most of them are not useful in terms of the utility purpose. In 2006 [3,4] Dwork et al. work describe the concept of Ɛ-differential privacy, a mathematical definition for the privacy loss associated with anydata release drawn from a statistical database.The intuition behind the Ɛ-differential privacy is that a person's privacy cannot be compromised by a statistical release if their data are not in the database. Therefore, with differential privacy, each roughly has the same privacy that would result from having their data removed. That is, the statistical functions that operate on the database should not overly rely on the data of any one individual.

In the recently, deep learning emerges as a prominent field of research,promises many possibilities that were not viable before. But deep learning techniques require huge volume of data to training purpose. The data may contain private information,any individually does not want to share it, on the other hand deep learningtechniques require all the feature for training purpose,it also perform better in their task.. To address this dilemma, differential privacy methods come to rescue. To achieve privacy in data, we are proposed here a new method, where we are using the autoencoders model to recreate the data and use the model's trained bias weight to generate a new copy of data under different epsilon value of differential privacy.

Related Work:

The literature in computer science regarding anonymized data release in extensive, we discuss directly relevant work here.

Dwork[3,4] et al. gave the concept of differential privacy, since then a lot of work has been done regarding data release using differential privacy concept.When it comes to perturbation (slight disturbance from normal state) mainly three types of operation done to achieve privacy when performing different task such as asking query from database or training a deep

learning model. First, Output perturbation technique use, in which the noise is added to the output provided by the algorithm on the database. Dwork et al. used the Laplace distribution to add the noise, and also show the method to add Gaussian noise to output, the amount of noise added to output control by the sensitivity of data. The Second method is objective perturbation, this method first given by chaudhari [2] et al. in 2011, in which they perturbed the **objective function ?** and optimized the function to perform machine learning task.

The third method, gradient perturbation technique adam etal [1]. where they perturbed the gradient of the loss function while updating weights of deep neural network during training of the model.This method is very famous in these days currently and implemented by various companies[14] to achieve differential privacy in their various deep learning task. Based on this method, a variational autoencoder deep learning method proposed by Acs G. et al. [5] they trained their network to generate synthetic data which also gurantee privacy. Another work given by **Abay** et al. [6] used autoencoder trained over original data which produce the data similar to original data satisfying differential privacy.Both the work in [5,6] uses the gradient perturbation method. Other techniques like in Zhang[7]et al. also popular for releasing private dataset.

Related Theory:

In our proposed work following termologies used.

Autoencoder: Autoencoder [8] is an artificial feed-forward neural network that makes use of unsupervised learning to learn the latent features of the input data.

These latent features generally have less dimensionality than the original data. It is a nonlinear dimensionality reduction technique, where the model tries to learn the latent representation of original data.These latent representation hold the core nature of data distribution after the model is trainded over data. The loss function's use in these modelsis mean squared error and weight update of layers followsthe back propagation method.

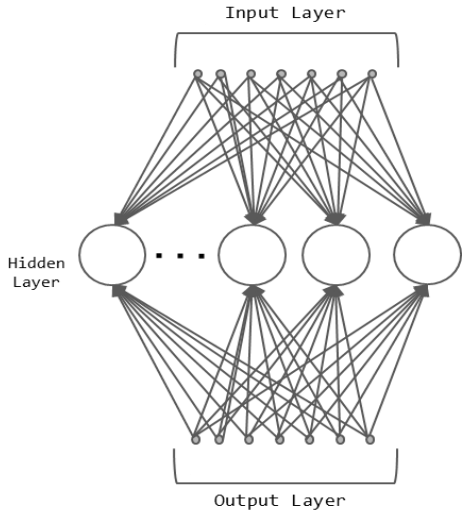


Fig 1.Vanilla Autoencoder.

Over the recent year, new optimization functions discovered which provide better weight update method hence better learning and lower loss.Many regularization techniques also used in these models to penalize the weights for overfitting.

Other versions of the autoencoder model also exist, such as variational autoencoder[9] which is a generative model. This model tries to learn the distribution of the data, hence capable of generatinga new type of synthetic data,which seems very close to real data.

Differential Privacy: The definition of Ɛ, δ differential privacy given by Dwork [3,4] et al, is that:

M:$X^n \rightarrow$ T satisfies (Ɛ, δ)differential privacy if for all neighboring database x,x'$\in X^n$such that distance (x,x')=1 and for all S $\subseteq$T

$$Pr[M(x)\in S] \leq e^{\varepsilon} Pr[M(x')\in S] + \delta$$

M is a Randomized function that takes data as input and returnsa randomized response over range T.

ϵis the measure of information leakage andδ is close to zero. For example, suppose a point does not exist in the original data, then there is no way it will exist in the neighboring datasets, so in that case value of δ, seems useful.

After observing the output of mechanism α,the privacy loss is given by the function

$$Loss=\ln(Pr[M(x)=\alpha]/Pr[M(x')=\alpha])$$

If the loss is positive then α is most likely from database x and if the quantity is negative then α is most likely from database x'.

The Ɛ, δ differential privacy ensures that for all adjacent x,x' the absolute value of the privacy loss will be bounded byεwith probability at least 1− δ.

Sensitivity: The sensitivity of functionm is defined as a maximum absolute distance between two neighboring pairs (x,x').

$$s=||m(x)-m(x')||$$

Where ||.|| is L2 norm.

The sensitivity of the function also control the amount of noise addition to the data as it is directly proportional to the value of standard deviation of the distribution which is used to calculate amount of noise to be added.

Gaussian Mechanism: The Ɛ, δ differential privacy of function m over database x is given by

$$M(x) = m(x) + noise$$

The noise is given by normal distribution $N(0,\sigma^2)$ where $\sigma^2\ Ɛ^2 \geq 2\ln(1.25/\delta)s^2$ for Ɛ[0,1].

Composition: Let M1,M2,……,Mk be sequence of Ɛ, δdifferential privacy mechanism then M1:k is (kƐ, k δ)differentially private. The composition theorem is a very powerful tool to ensure privacy, suppose a dataset is released with some noise added according to the DP(Ɛ, δ), after that again some transformation and operations performed in the dataset and some noise is added according to DP(Ɛ, δ), then finally released dataset , will be (2Ɛ,2δ) differentially private.

Hellinger Distance: The Hellingerdistance[10] is used to quantify the distance between two probability distributions.

TheHellinger distance for two discrete probability distributionsP, Q is defined as:

$$H(P,Q) = \frac{1}{\sqrt{2}} \left\| \sqrt{P} - \sqrt{Q} \right\|_2.$$

H(P,Q) is zero when P,Q both represent the same distribution.H(P,Q) is one when P assignsa probability to 1 to each observation when Q assignsa probability to zero to that observation, which means no similarity at all.

Bhattacharyya coefficient: Bhattacharyya coefficient[11] is used to approximate the similarity between two statistical samples.

$$BC(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} \sqrt{p_i q_i},$$

The Hellinger distance is related to Bhatta -charyya coefficient by given relationship:

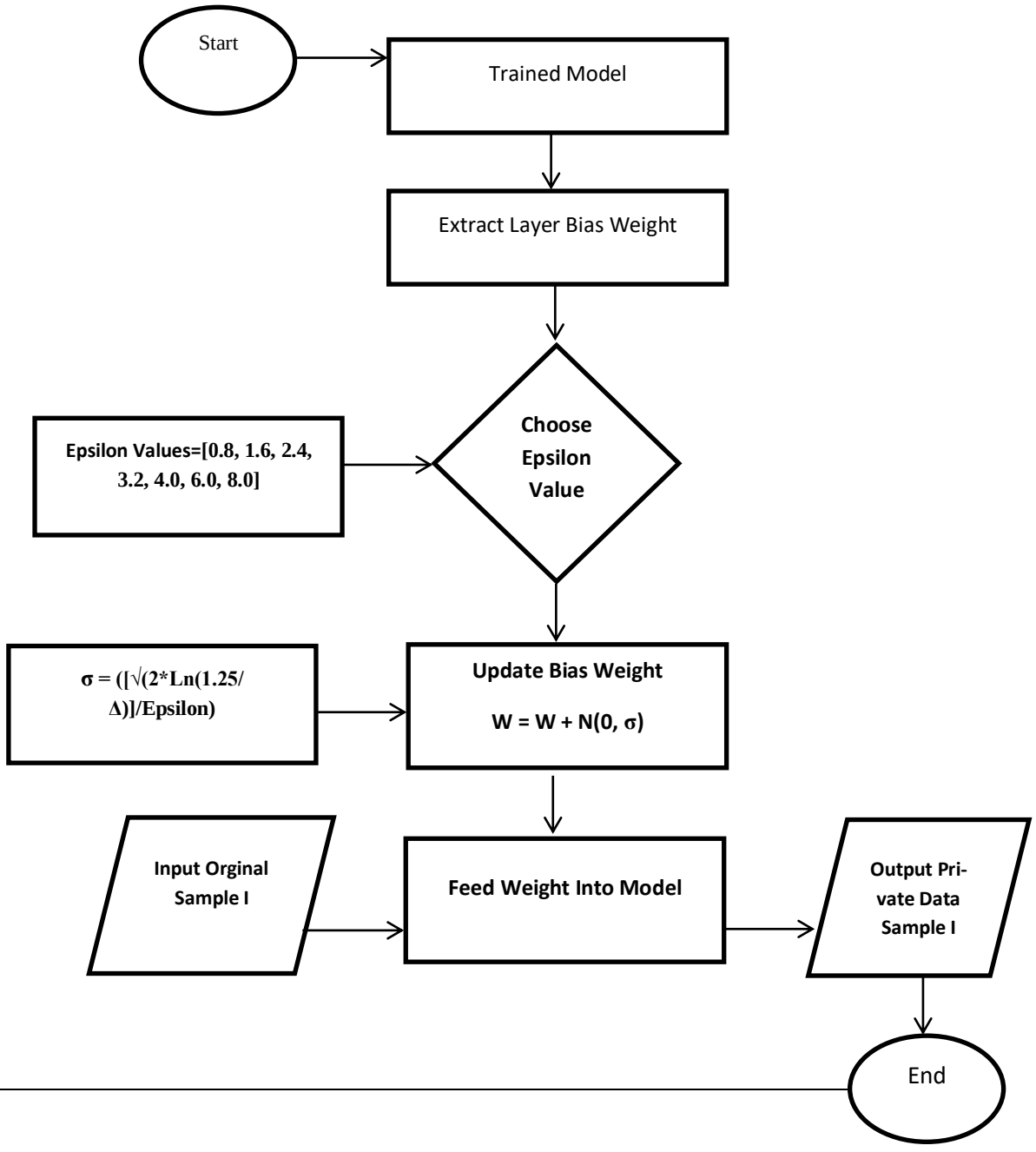
$$H(P,Q) = \sqrt{1 - BC(P,Q)}.$$

Methodology:

In this section we will present our proposed approach in detail.

First we trained our autoencoder model, after the model training we are extracting the bias weight of model's hidden layer and output layer as shown in fig 2. This proposed work has performed three operations, in first operations we have only perturbed the hidden layer bias weight, keeping all other weight of model unchanged, in second operation, we have only perturbed output layer bias weight and in third operations, we have perturbed the bias weights of hidden and output layers. In this way we created 3 different autoencoder models that have same weights as of our original model expect the bias weights. We have taken the 7 different epsilon values, after applying gaussian noise according to given epsilon values total 21 private combination of data generated. In this work perturbation noise is added from the Gaussian distribution which has zero mean and its standard deviation is controlled

Fig 2: Flow Chart of the Proposed Method.



by epsilon value according to formula given in result section.
To verify the similarity of generated datawith the original dataset, Histogram of original image and generated data is used for the Hellinger distance and Bhattacharyya coefficient calculation. The detailed comparison of data produced by experiment is given in result section of this paper, it is showing the effect of bias weight perturbation visually.
This proposed work does not require the need for retraining the model for different epsilon values as in existing method [1], one needs to retrain the model if they need privacy of different epsilon value.This proposed work executed fastly and it is also cost effective when comes to private data release and can also be applicable for private machine learning methods.

Dataset used:

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST(National Institute of Standards and Technology). The digits have been size-normalized and centered in a fixed-size image.

Results and Discussion:

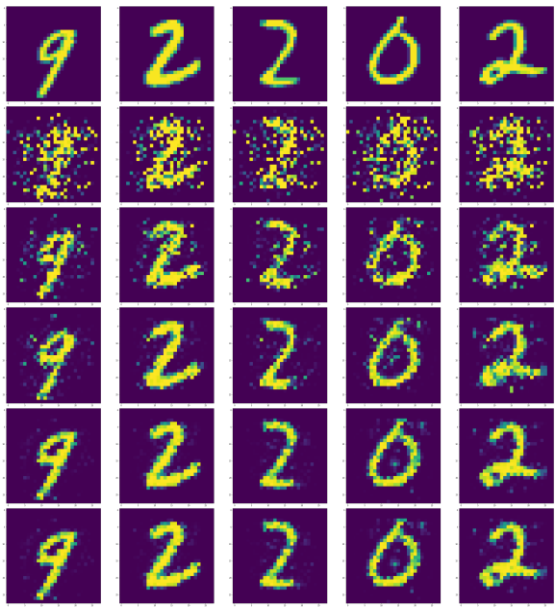
We have used the autoencoder model with one layer of hidden unit equals to 64, the activation function for the hidden layer is relu and the activation function for the outer layer is chosen to be sigmoid. The dataset values are normalized before feeding into the model. The loss function used for the model is mean squared error and the optimizer algorithm used is Adam[13] with default learning rate. The platform used for the experimentation is Google Colab.

For Bias Weight Perturbation:

Normal distribution with zero mean and Standard deviation, $\sigma = ([\sqrt{(2*\ln(1.25/\delta)}]/epsilon)$ is used , where δ value is 0.00001.
The Epsilon values used for experiments are[0.8, 1.6, 2.4, 3.2, 4.0, 6.0, 8.0].The sensitivity of the dataset always taken 1.
In this section we have plotted images of different samples from data we created according to different epsilon values. We can see from there that when we take epsilon values a lowi.e more privacy , the image reconstruction is very lossy and when we take epsilon values high i.e low privacy images start appear to be reasonable. Table 1,2,3 represent Hellinger distance and corresponding Bhattacharyya coefficient between the generated sample image and original sample image.In our proposed method these distances and coefficient gives the sense that how close a produced image from the autoencoder model is with the original image.
The reconstruction of images from the model shown in Fig2. Corresponds to the bias weight perturbation of the hidden layer, it can be seen clearly that , image produced for lower epsilon values are very distorted than the images produced at higher epsilon values, it is due to the fact that ,bias weight of the hidden layer effect the reconstruction of the images more as the hidden layer units represent the intrinsic, internal representation of the original images, so any distortion in output of these units more likely to distort the whole image reconstruction more.
For images shown in Fig3. It can be seen clearly that reconstruction is far more smoother than the images in Fig2. The noise is being added to the bias weight of the the output layer, and its effect is less as when noise added to the bias weight of hidden layer. It can be also verified through the hellinger distance values in Table1 and Table2.
For images shown in Fig4. The reconstruction loss is more than shown in Fig2, Fig3. Here when we are adding noise to the bias weight of hidden and output , the noise added is more than the individually, because now for same epsilon value the noise added is twice the noise added individually in bias weight of hidden and output layers. The reason for that is composition theorem, for example if we want to add the noise for epsilon value 0.8,then we have to

add the noise in bias weight of hidden and output of epsilon value 0.4 each, so then according to composition theorem total noise added will be according to epsilon value 0.8.
The other thing to notice here which is obvious with the nature of deep learning methods, that if we see the values of Bhattacharyya coefficient in Table 3. We found that for lesser noise, similarity does not increase as it should be , for example Bhattacharyya coefficient value for epsilon value of 1.6 is lower than epsilon value of 0.8 in Table 3. The decrease in similarity conforms with the nonlinear nature of the autoencoder method.

Fig.2: (Top) original sample images, from second row to bottom sample images generated by hidden layer bias weights perturbationwith epsilon values = [0.8, 1.6, 2.4, 3.2, 4.0, 6.0, 8.0].



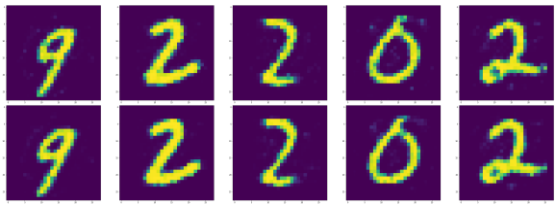


Fig.3: (Top) original sample images, from second row to bottom sample images generated by output layer bias weights perturbation with epsilon values = [[0.8, 1.6, 2.4, 3.2, 4.0, 6.0, 8.0].

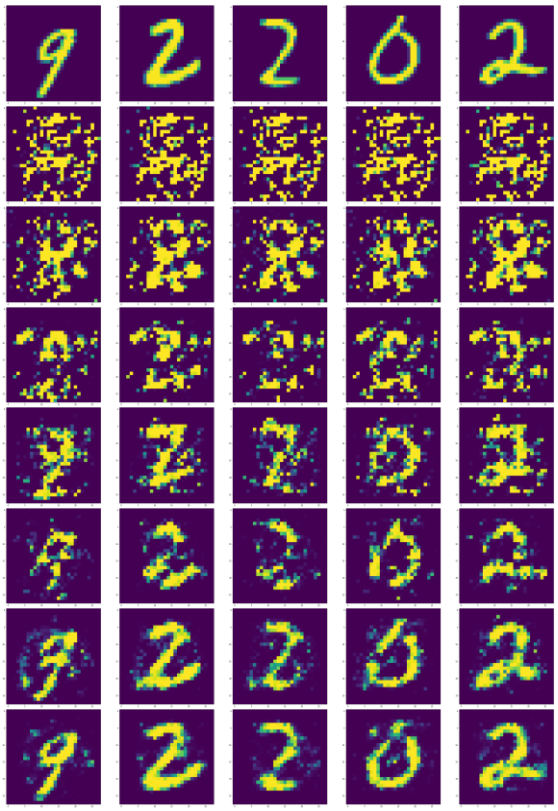


Fig 4: (Top) original sample images, from second row to bottom sample images generated by hidden and output layer bias weights perturbation with epsilon values = [0.8, 1.6, 2.4, 3.2, 4.0, 6.0, 8.0].

Table.1 Sample Image of Number9

Hidden Layer Bias Weight Perturbation		
EPSILON	HELLINGER DISTANCE	BHATTACHARYYA COEFFICIENT
0.8	0.10864	0.96906
1.6	0.10776	0.96925
2.4	0.09952	0.97096
3.2	0.09144	0.97250
4	0.09734	0.97139
6	0.08218	0.97411
8	0.08001	0.97446

Table.2 Sample Image of Number9

Output Layer Bias Weight Perturbation		
EPSILON	HELLINGER DISTANCE	BHATTACHARYYA COEFFICIENT
0.8	0.16755	0.95279
1.6	0.12510	0.96521
2.4	0.11106	0.96853

3.2	0.08301	0.97397
4	0.10981	0.96880
6	0.09440	0.97195
8	0.07300	0.97553

Table.3 Sample Image of Number9

Hidden and Output Layer Bias Weight Perturbation		
EPSILON	HELLINGER DISTANCE	BHATTACHARYYA COEFFICIENT
0.8	0.19541	0.94268
1.6	0.28183	0.90143
2.4	0.18131	0.94799
3.2	0.07671	0.97498
4	0.15027	0.95828
6	0.15492	0.95686
8	0.07008	0.97595

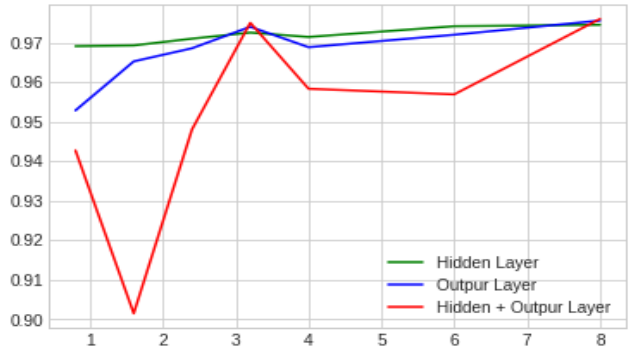


Fig.5: Above figure shows the samples similarity of sample image of number 9 according to Bhattacharyya Coefficient.

*To calculate the Hellinger distance and Bhattacharyya coefficient, the image histogram is formed with 16 bins. No of bins is chosen randomly.
*The values shown in Table[1,2,3] are the best of 10 runs. For each epsilon value experiment is repeated for 10 times.
*The autoencoder is nonlinear dimensionality reduction model, the output function also follows non linearity with bias weight, and therefore, even if value of epsilon is larger it may not produce larger similarity[Bhattacharyya coefficient] as it should be. These irregularities can be seen in fig.5 .

Conclusion:

In this paper, we showed that perturbation of bias weight with the Gaussian noise controlled through different epsilon value under differential privacy principle can be used to release data under privacy constraint. The effect of bias weight perturbation of trained model is nonlinear which align with nature of output function of model. The above method also can be used to perturbation of certain features of data, by adding noise to weights related to those features.

References:

1. RaefBassily, Adam Smith, and AbhradeepThakurta. 2014. Private empiricalrisk minimization: Efficient algorithms and tight error bounds. In2014 IEEE 55thAnnual Symposium on Foundations of Computer Science.464–473.

2. AmalikaChaudhuri, Claire Monteleoni, and AnandDSarwate. Differentially privateempirical risk minimization.Journal of Machine Learning Research, 12(Mar):1069–1109, 2011.
3. CynthiaDwork, Frank McSherry, KobbiNissim, and Adam Smith. Calibrating noise tosensitivity in private data analysis.InProceedings of the Third Conference on Theoryof Cryptography, TCC'06, pages 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.
4. CynthiaDwork and Aaron Roth. The algorithmic foundations of differential privacy.Foundations and TrendsR©in Theoretical Computer Science, 9(3–4):211–407, 2014
5. Acs, G., Melis, L., Castelluccia, C., Cristofaro, E.D.: Differentially private mixture of generative neural networks. CoRR abs/1709.04514 (2017), http://arxiv.org/abs/1709.04514
6. Abay, Nazmiye& Zhou, Yan &Kantarcioglu, Murat &Thuraisingham, Bhavani& Sweeney, Latanya. (2018). Privacy Preserving Synthetic Data Release Using Deep Learning.510-526. 10.1007/978-3-030-10925-7_31.
7. Zhang, Jun &Cormode, Graham &Procopiuc, Cecilia &Srivastava, Divesh& Xiao, Xiaokui. (2017). PrivBayes: Private Data Release via Bayesian Networks. ACM Transactions on Database Systems. 42. 1-41. 10.1145/3134428.
8. Chapter 14,Autoencoder,Ian Goodfellow, YoshuaBengio, & Aaron Courville (2016). *Deep Learning*.MIT Press.
9. .Chapter 20.10.3, VariationalAutoencoder, Ian Goodfellow, YoshuaBengio, & Aaron Courville (2016). *Deep Learning*.MIT Press.
10. Wikipedia contributors. (2020). Hellinger distance –- Wikipedia, The Free Encyclopedia.
11. Wikipedia contributors. (2020). Bhattacharyya distance –- Wikipedia, The Free Encyclopedia.
12. http://yann.lecun.com/exdb/mnist/
13. Kingma, D. P., & Ba, J. L. (2015). Adam: a Method for Stochastic Optimization. International Conference on Learning Representations, 1–13.
14. https://en.wikipedia.org/wiki/Implementations _of_differentially_private_analyses