2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD) | 979-8-3503-3168-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/CSCWD57460.2023.10152834

DLMT: Outsourcing Deep Learning with Privacy Protection Based on Matrix Transformation

Dongdong Zhao[1,2], Ying Chen[1], Jianwen Xiang[1,2,*], Huanhuan Li[3]
[1]*School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan, China*
[2]*Chongqing Research Institute, Wuhan University of Technology, Chongqing, China*
[3]*China University of Geosciences, Wuhan, China*
[zdd, chenying98, jwxiang]@whut.edu.cn, julylhh@gmail.com

***Abstract*—In recent years, deep learning has been applied in a wide variety of domains and gains outstanding success. In order to achieve high accuracy, a large amount of training data and high-performance hardware are necessary for deep learning. In real-world applications, many deep learning developers usually rent cloud GPU servers to train or deploy their models. Since training data may contain sensitive information, training models on cloud servers will cause severe privacy leakage problem. To solve this problem, we propose a privacy-preserving deep learning model based on matrix transformation. Specifically, we transform original data by adding or multiplying a random matrix. The obtained data is significantly different from the origin and it is hard to recover original data, so it can protect the privacy in original data. Experimental results demonstrate that the models trained with processed data can achieve high accuracy.**

***Index Terms*—Deep Learning, Privacy Protection, Data Security, Matrix Transformation**

I. Introduction

On the benefit of the advance of neural network theory and computer hardware, deep learning has been widely used in many fields such as object detection [1], [2], image classification [3] and word representation [4]. In general, in order to get a high accuracy model, massive amounts of training data is essential. Besides, training a model usually is time and resource consuming. In real-world applications, developers tend to utilize high-performance GPU cloud servers to train or deploy their models. Before training on cloud servers, the developers need to upload their data in advance. However, in some cases, these data may contain some sensitive information like personal medical records, human face images and financial data. Since the cloud server is not totally controlled by the data owner, if the data owner uploads his original data to the cloud server, he can't prevent the leakage of the data which would cause severe privacy problem. Moreover, even the data owner conducts the training process locally but uploads the trained model to a cloud server, attackers can still infer some useful information about the training data from model parameters [5]. In federated learning [6], some data owners train a network together by sharing their data or model parameters [7] trained locally. This strategy will also cause privacy concerns. If one of the participants is malicious or the attacker can intercept the network between data owners and the server, then the privacy in training data will be at great risk of leakage [8], [9]. Therefore, it is meaningful to propose privacy-preserving deep learning models.

Presently, several methods for protecting data privacy during deep learning have been proposed. Part of them try to combine deep learning with homomorphic encryption [10]–[12] to protect data privacy. However, due to the high computation and communication complexity, these methods are usually hard to apply in the applications involving a large amount of data. Methods based on differential privacy [13]–[15] try to protect privacy by adding proper noises into gradients or data. But these methods usually have to add large noises to achieve high level of privacy protection, and this often results in a non-negligible degradation of accuracy. Some researchers try to process data and local gradients into another form before uploading them to an untrusted server to achieve privacy preserving purpose. In [16], a suite of image disguising mechanisms was proposed to protect image data. Unfortunately, these mechanisms will also have non-negligible degradation of accuracy to achieve high level of privacy protection.

To summarize, most of existing methods could not achieve high level of privacy protection efficiently with a negligible degradation of accuracy. In this paper, we propose a new privacy-preserving deep learning method with high accuracy based on matrix transformation. Specifically, we treat each training image data as a pixel matrix, and add or multiply it with a random matrix element by element. Data owners can either train a model locally using the processed data and upload the model to cloud servers for inference, or directly submit the data to the servers for training. As a result, the processed data is significantly different from the origin, and it is difficult to recover original data. Hence, the proposed method can protect the original image data. Overall, the contributions of this paper are listed below:

1) We propose a privacy-preserving deep learning method with high accuracy based on matrix transformation.
2) We study the effectiveness of two matrix transformations, i.e. matrix addition and matrix multiplication.
3) We analyze the security of the proposed method and demonstrate that it is difficult for adversaries to recover original data.
4) We conduct several experiments to investigate the per-

*Corresponding author

formance of the proposed method. Results show that the proposed method could achieve promising accuracy and the balance between privacy and accuracy can be well controlled through parameters.

The rest of this paper is organized as follows: Section II reviews related works about privacy-preserving deep learning models. Section III describes proposed method in detail and analyzes the security and efficiency. In Section IV, we first test the accuracy of our method with CIFAR-10 and MNIST datasets. After that we present the comparisons results of DLMT with state-of-the-art method. In the end, the conclusion for DLMT and future work can be found in Section V.

II. Related work

Presently, several methods for protecting data privacy during deep learning have been proposed. In general, these methods can be divided into three categories: methods based on homomorphic encryption [10]–[12], [18]–[22], methods based on differential privacy [13], [14], [23], [24] and methods based on data processing [16], [17], [25].

Methods based on homomorphic encryption: In 2016, Gilad-Bachrach et al. [10] proposed CryptoNets which combine homomorphic encryption with deep neural network. In their scheme, the data owner encrypts the data into ciphertext and then sends it to a cloud server. The server performs the inference on ciphertext and returns the encrypted label to the data owner. After that the data owner can decrypt it to get the correct label. The shortage of CryptoNets is that when the network contains too many non-linear layers, it will become ineffective and its accuracy will decrease greatly. In order to solve this problem, Chabanne et al. [11] tried to replace activation functions by their polynomial approximations. As a result, their method can be applied to a deeper network. Gradients-encrypted schemes can be found in [12], [18]. In [12], gradients-encrypted asynchronous SGD (Stochastic Gradient Descent) was proposed to protect gradients during federated learning through additively homomorphic encryption. Hesamifard et al. [19] combined deep neural networks with HE (Homomorphic Encryption) friendly activation functions to keep high accuracy. Zhu et al. [20] used the Paillier homomorphic cryptosystem to encrypt data in deep learning. In [21], the authors reformulated batch normalization to make it compatible with the use of FHE (Fully Homomorphic Encryption). Li et al. [22] designed a new scheme that uses multi-key fully homomorphic encryption to protect data.

Methods based on differential privacy: In 2016, an algorithm called DPSGD (Differentially Private stochastic gradient descent) was proposed by Abadi et al. [13] to protect the gradients. To achieve high level of privacy protection, the algorithm usually has to add large noises into gradients which will cause a great degradation of accuracy. In [14], Geyer et al. proposed a method to protect each participant's gradients during federated learning based on the differential privacy model. In [23], the author studied the m-neighborhood notion based on differential privacy to protect sensitive information in image data. Chamikara et al. [24] studied an image processing method for protecting human face data with differential privacy.

Methods based on data processing: In 2018, Sharma et al. [16] proposed a method based on block-wise permutation and RMT (Randomized Multidimensional Transformations). They first partition original image data into blocks and then shuffle the blocks. To provide better protection for data, they multiply the pixel matrix of images with a random orthogonal or projection matrix in each block. To achieve high level of privacy protection, Sharma's scheme will also have non-negligible degradation of accuracy. Sirichotedumrong et al. [17] introduced a new method for privacy-preserving deep learning. In their method, each image pixel is XORed with 255 with the probability 50%, and the result is then submitted to the server for training or testing. However, since there are about half of image pixels remain unchanged, some information about original data would be recovered. In [25], the authors introduced a transformation network to protect visual information in images.

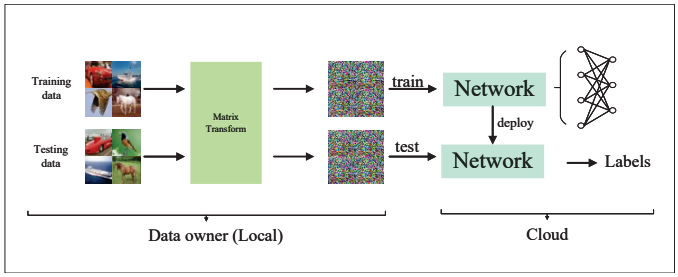


Fig. 1. The framework of DLMT.

III. The proposed method

In this section, we will describe the problem we want to solve and the method we use to solve this problem. We also analyze the security and complexity of our method in Section III-D and Section III-E.

A. Problem description

In traditional framework, the data owner usually trains the model M with his private data (x, y). In real-world deep learning applications, training a model usually is time and resource consuming. Without high-performance GPU hardware, training a model with high accuracy will be extremely slow. To reduce the cost of developing a model, many developers usually will choose to rent a cloud GPU server rather than buy a new GPU server to train their models. To train models on cloud GPU servers, the developers need to upload their training data (x, y) in advance. This operation will bring severe privacy leakage problem since the training data x may contain sensitive information and cloud servers are not completely safe. In order to protect the data (x, y), we propose to utilize one method F to preprocess it and hide the sensitive information in x. After that, the data owner can only upload new data $(F(x), y)$ into unsafe cloud servers

for training. In cloud servers, the training program can train model M with processed data $(F(x), y)$. To satisfy the security requirement, there are two key problems that need to be solved: 1. The processed data $F(x)$ can not reveal main information in original data like object category since these information usually is sensitive and needs to be protected from adversaries; 2. The process must be hard to reverse. If the process is easy to reverse, with processed data, adversaries can easily recover the original data which will directly leak main sensitive information in original data.

Algorithm 1 DLMT

Input: Original dataset $D = \{(A_1, y_1), \ldots, (A_N, y_N)\}$; The dimension of $A : W \times H \times C$; MAX_V; $RISE\_V$;
Output: private dataset D';

1: Initialize random matrix R, each value in R is in the interval $[1, MAX\_V]$;
2: **for** $i \leftarrow 1$ to N **do**:
3: **Matrix Transformation**: Transform A_i into A'_i using Eq. (1)-(5);
4: **return** $D' = \{(A'_1, y_1), \ldots, (A'_N, y_N)\}$.

B. Main framework of DLMT

Training models with original data on cloud servers will cause severe privacy leakage problem. To provide protection for these original data, we propose a privacy-preserving deep learning model called DLMT based on matrix transformation. To satisfy the security requirement, our idea is that: instead of training models with original data on the cloud server, the data owner needs to preprocess the data locally to hide sensitive information and then send the processed data to the server for training and testing. By this way, even if the processed data or trained model parameters are disclosed to adversaries, they can't recover the original data and the privacy is under protection. During preprocessing, we propose to use Matrix Transformation (MT) to achieve this goal. Each image data needs to be processed with a same random matrix. After that, the data owner can upload the processed image data to the cloud server for training and testing. Alternatively, he can also train the models locally with processed data and send the parameters to cloud server for further federated learning. The details of DLMT can be found in Algorithm 1 and the framework of DLMT is shown in Fig. 1. In algorithm 1, the data owner should transform their original data with one kind of Matrix Transformation methods in Section III-C. In this step, each training and testing data must be transformed with a same random matrix R and the dimension of R is as same as the original data. Notice that, if we choose to use Eq. (4) or Eq. (5) to transform original data, the MAX_V for MAT and MMT are usually different and we use MAX_V_ADD and MAX_V_MUL to differentiate them. When all data is processed with Matrix Transformation, the algorithm will return the private dataset which is used for training and testing.

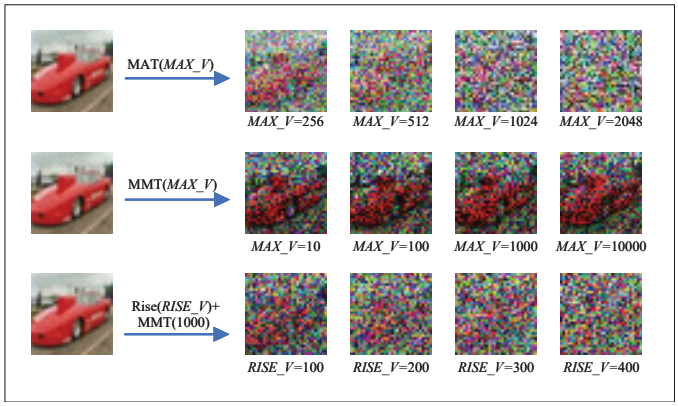


Fig. 2. Converted images with different MT methods.

C. Matrix Transformation

For an image A, we treat it as a pixel matrix with dimension of $W \times H \times C$, where W is the width, H is the height and C is the number of channels. MT randomly generates a matrix R with the same dimension, and adds or multiplies A with R element by element. Each value in R is a random integer in the interval $[1, MAX\_V]$ and MAX_V is a positive integer. We call the two methods as Matrix Addition Transformation (MAT) and Matrix Multiplication Transformation (MMT), respectively.

For MMT, in order to enhance the difference between small values and large values in A and hide sensitive information better, we add all values in A with a constant number $RISE\_V$ before multiplication and call this method "Rise+MMT". Furthermore, we can combine MAT with MMT to get two hybrid transformations: MAT+MMT, MMT+MAT. In summary, the details of five MT methods can be found in Eq. (1)-(5). To differentiate the random matrix R in MAT and MMT, we use R_{add} and R_{mul}, respectively. In Eq. (1)-(5), $i \in [1, W]$, $j \in [1, H]$, $k \in [1, C]$.

MAT :

$$A'[i,j,k] = A[i,j,k] + R[i,j,k] \tag{1}$$

MMT:

$$A'[i,j,k] = A[i,j,k] * R[i,j,k] \tag{2}$$

Rise+MMT:

$$A'[i,j,k] = (A[i,j,k] + RISE\_V) * R[i,j,k] \tag{3}$$

MAT+MMT:

$$A'[i,j,k] = (A[i,j,k] + R_{add}[i,j,k]) * R_{mul}[i,j,k] \tag{4}$$

MMT+MAT:

$$A'[i,j,k] = A[i,j,k] * R_{mul}[i,j,k] + R_{add}[i,j,k] \tag{5}$$

Fig. 2 shows the converted images with different MT methods. As shown in Fig. 2, the original image contains a red car. In MAT transformation, when the MAX_V_ADD

reaches 1024, from the converted image we can hardly recognize it contains a red car. In MMT transformation, when we change MAX_V_MUL from 10 to 10000, though the converted image is quite different from the original image, from converted image we can still know that it may contain a red car. So it seems like MAT can provide better protection for original image than MMT. Similarly, in Rise+MMT, when the $RISE\_V$ reaches 300, the converted image nearly contains no information about a red car.

In real applications, uploading original data into cloud servers to train a deep neural network will cause severe privacy leakage problem. In order to protect these data, the data owner can use DLMT to preprocess their data. As shown in Fig. 2, the processed images are quite different from the original images. If we choose proper parameter settings, DLMT can well hide main information in original data. After that the data owner can upload the processed data into cloud servers to train their models. Even if the processed data is leaked, the adversary can hardly know the main information from the processed data. And DLMT can satisfy the requirement: the processed data can not reveal main information. In Section III-D, we will prove that DLMT can also satisfy the second requirement that the process must be hard to reverse.

D. Security analysis

In this section, we analyze security of DLMT and prove that our method is hard to reverse. In DLMT, the data owner only uploads the processed data into cloud servers for training or testing, and adversaries could get nothing but processed data. The matrix R used to transform original data is not open to public.

For MAT, each value in the random matrix R of MAT is an integer value in $[1, MAX\_V\_ADD]$. Suppose that with a certain transformed value $A'[i,j,k]$, its original pixel value $A[i,j,k]$ has $G[i,j,k]$ different possible values and $G[i,j,k]$ is calculated by the following equation:

$$G[i,j,k] = \begin{cases} 256 & (A'[i,j,k] > 256) \\ A'[i,j,k] & (A'[i,j,k] \leq 256) \end{cases} \tag{6}$$

And the original pixel matrix A has

$$\prod_{i=1}^{W}\prod_{j=1}^{H}\prod_{k=1}^{C} G[i,j,k] \tag{7}$$

possible alternatives. For MMT, we first define a function $T(x)$ and $T(x)$ returns the number of x's positive factors less than 256. For example, $T(3) = 2$ and $T(15) = 4$. With a certain transformed value $A'[i,j,k]$, $G[i,j,k]$ is calculated by the following equation:

$$G[i,j,k] = \begin{cases} 256 & (T(A'[i,j,k]) > 256) \\ A'[i,j,k] & (0 < T(A'[i,j,k]) \leq 256) \\ 1 & (T(A'[i,j,k] = 0) \end{cases} \tag{8}$$

And the original pixel matrix A has

$$\prod_{i=1}^{W}\prod_{j=1}^{H}\prod_{k=1}^{C} G[i,j,k] \tag{9}$$

possible alternatives (Notice that $G[i,j,k]$ in Eq. (9) is calculated by Eq. (8)). For Rise+MMT, the adversary should know both the $RISE\_V$ and random matrix R, then he can recover the original data. Suppose $RISE\_V$ is a random integer in $[1,r]$, there are $r \times G[i,j,k]$ alternatives for the combination of $RISE\_V$ and random matrix R from the perspective of adversaries. For example, on CIFAR-10 dataset, each image has 3 channels and each channel has 1024 pixels in $[0,255]$. For each converted image A' processed with MAT, its original image has $G[1,1,1] \times \cdots \times G[32,32,3]$ possible alternatives($1 \leq G[i,j,k] \leq 256$). It's a huge number and it is difficult for adversary to recover those data only with the processed data. And DLMT can satisfy the requirement that the process must be hard to reverse.

E. Efficiency Analysis

According to Eq. (1), in MAT an original pixel matrix with dimension of $W \times H \times C$ will be added with a random matrix element by element. So, the computational complexity of MAT is $O(W \times H \times C)$. Similarly, the computational complexity of MMT is $O(W \times H \times C)$, too. In Rise+MMT, each pixel will be added with a certain value $RISE\_V$ and then multiplied with a random value. The computational complexity of Rise+MMT is $O(W \times H \times C)$. In MAT+MMT and MMT+MAT, the computational complexity of these two combinations is $O(W \times H \times C)$.

TABLE I
EXPERIMENT ENVIRONMENT

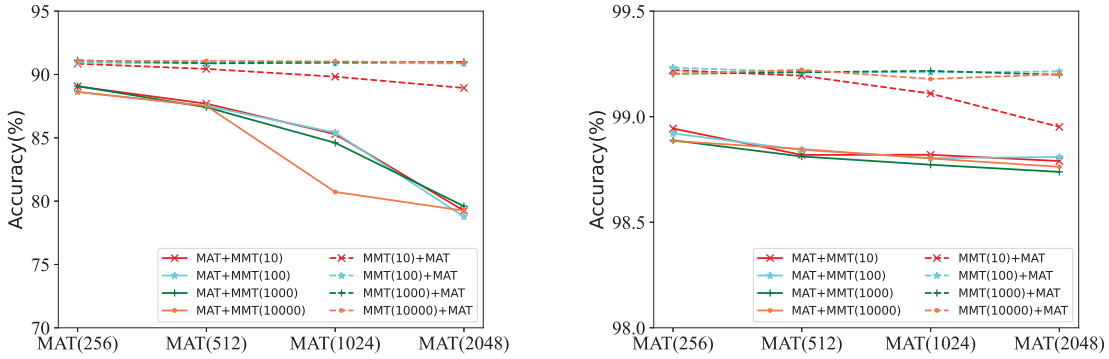
Operating System	CPU	GPU	Coding language
Ubuntu 20.04 LST	I5-8500 3.6GHz	3080Ti	Python3.8

IV. EXPERIMENTAL RESULTS

A. Model Accuracy Results

We implement DLMT using Pytorch and conduct several experiments on two benchmark datasets: CIFAR-10 and MNIST. For the two datasets, we utilize ResNet18 [3] and a simple network with five convolutional layers to classify images. In training, we use the SGDR (Stochastic Gradient Descent with warm restarts [26]) strategy with initial learning rate at 0.2 descending to 0 gradually and the maximum epoch is 200. The main information of our experimental environment can be found in TABLE I. Each experiment will be repeated at 10 times and the experiment results are stable. At last, the implementation of DLMT is based on the source codes of [27].

The results are shown in TABLE II and Fig. 3. On CIFAR-10 and MNIST datasets, DLMT could achieve a best accuracy of 91.361% and 99.211%, respectively. It is shown in TABLE II that the accuracy of DLMT decreases gradually as the MAX_V_ADD in MAT increases. The accuracy of DLMT tends to be stable when the MAX_V_MUL in MMT changes from 10 to 10000, and they might have no obvious relationship. When MAX_V_MUL in MMT is a fixed value, the accuracy of DLMT decreases about 0.81% on CIFAR-10 and 0.05% on MNIST as the $RISE\_V$ increases 100. In



(a) CIFAR-10 (b) MNIST

Fig. 3. Accuracy of DLMT using MAT+MMT and MMT+MAT

TABLE II
AVERAGE ACCURACY ON CIFAR-10 AND MNIST

	CIFAR-10	MNIST
Original Model	95.220%	99.440%
MAT(256)	**91.361%**	99.205%
MAT(512)	89.906%	99.026%
MAT(1024)	87.918%	98.906%
MAT(2048)	84.067%	98.879%
MMT(10)	91.222%	**99.211%**
MMT(100)	91.090%	99.202%
MMT(1000)	91.010%	99.201%
MMT(10000)	90.993%	99.211%
Rise(100)+MMT(1000)	90.166%	99.018%
Rise(200)+MMT(1000)	89.592%	98.923%
Rise(300)+MMT(1000)	88.586%	98.885%
Rise(400)+MMT(1000)	88.154%	98.870%

each image of MNIST dataset, most pixels' values are 0 and the main information inside the image is more obvious. After process, the loss of useful information on MNIST dataset is much smaller than CIFAR-10 dataset. That is the reason why the accuracy loss of DLMT on CIFAR-10 dataset is greater than MNIST dataset.

It is shown in Fig. 3 that, the accuracy of DLMT using MAT+MMT is significantly influenced by the MAX_V_ADD in MAT. When the MAX_V_MUL in MMT is relatively small, the accuracy of DLMT is mainly determined by the MAX_V_ADD in MAT. And the accuracy changes seem to be mainly determined by the MAX_V_MUL in MMT when its value reaches 100.

To summarize, we test accuracy of DLMT using different methods like MAT. The results show that our proposal can achieve high level accuracy. At the same time, DLMT can achieve different accuracy levels by utilizing different methods or parameter settings. For example, in MAT we can adjust the MAX_V_ADD to satisfy different privacy preserving requirements and get corresponding model accuracy. On CIFAR-10 dataset, we even test the accuracy of MAT(5) at 95.005% which is extremely close to the accuracy on original CIFAR-10 dataset. So, DLMT makes it possible that we can balance security and model accuracy in a more qualitative level.

B. Comparisons with state-of-the-art method

In this section, we compare DLMT with Sharma's method [16] on Model Accuracy (MA), Visual Privacy (VP) and Model Mis-usability (MM) [16]. Model Accuracy can show us how effectively our models can identify images. Visual Privacy can measure how many useful information remains in processed data. It is defined as (1- accuracy of the DNN examiner) [16]. The higher Visual Privacy is, the less useful information remains in processed data. Model Mis-usability is a measurement to show how possible our models will be misused. The comparison results on MNIST and CIFAR-10 datasets can be found in TABLE III. Some results are estimated from [16] since we could not reproduce the same results successfully. TABLE III shows that on MNIST dataset DLMT can achieve the same level of Model Accuracy, Visual Privacy and Model Mis-usability with Sharma's method in [16]. This is because image classification on MNIST dataset is relatively simple and both the two methods can get approximately best results. But compared with Sharma's method, DLMT can get more diverse Model Accuracy, Visual Privacy and Model Mis-usability results by fine-grained parameters control. In TABLE III, we can know that on CIFAR-10 dataset, with same level of Model Accuracy and Visual Privacy, DLMT can achieve better Model Mis-usability. For example, compared with Sharma's method (N=0 w/o perm. Block size=2×2), DLMT using MAT(256) has a lower Model Mis-usability at 20.161%.

V. CONCLUSION

In this paper, we propose a privacy-preserving deep learning model called DLMT based on matrix transformation. To hide the sensitive information like background and texture in original images from adversaries, data owners can use DLMT to process their image data before uploading it to cloud servers. Our security and efficiency analyses show that DLMT can satisfy the privacy preserving requirements with high efficiency. Our experiments demonstrate that the models

TABLE III
MA, VP, MM OF DLMT AND SHARMA'S METHOD ON CIFAR-10 AND MNIST

Mechanism		CIFAR-10			MNIST		
		MA	VP	MM	MA	VP	MM
DLMT	MAT(256)	**91.361%**	**90.256%**	20.161%	**99.205%**	90.259%	33.843%
	MMT(100)	91.090%	90.028%	10.232%	99.202%	49.000%	67.755%
	Rise(100)+MMT(1000)	90.166%	90.000%	10.000%	99.018%	87.006%	10.381%
	MAT(256)+MMT(10)	89.062%	90.000%	10.238%	98.944%	90.152%	10.481%
	MMT(10)+MAT(2048)	88.933%	89.789%	**9.627%**	98.952%	90.254%	10.036%
Sharma's method	N=0 w/o perm. Block size=2×2	91.243%	90.501%	≈42%	99.218%	85.512%	≈14%
	N=50 w/o perm. Block size=2×2	88.952%	90.065%	≈34%	99.196%	**93.708%**	**≈9%**
	N=0 w. perm. Block size=32×32 (28×28 on MNIST)	77.035%	88.782%	≈21%	98.800%	88.398%	≈13%
	N=50 w. perm. Block size=32×32 (28×28 on MNIST)	73.800%	89.675%	≈14%	98.725%	89.774%	≈12%

trained with processed data can achieve high accuracy and data owner even can adjust model accuracy for different privacy preserving requirements by utilizing proper DLMT strategies and parameter settings. In future work, we would like to pursue some other data preprocessing methods that can achieve higher accuracy and security. We will also try to apply DLMT to some real-world scenarios with concrete privacy protection requirements.

ACKNOWLEDGMENT

This work was partially supported by the National Natural Science Foundation of China (Grant No. 61806151, 61672398) and the Natural Science Foundation of Chongqing City (Grant No. CSTC2021JCYJ-MSXMX0002).

REFERENCES

[1] Joseph Redmon and Ali Farhadi, "Yolov3: An incremental improvement", arXiv preprint arXiv:1804.02767, 2018.
[2] Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao, "Yolov4: Optimal speed and accuracy of object detection", arXiv preprint arXiv:2004.10934, 2020.
[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, "Deep residual learning for image recognition", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770-778, 2016.
[4] Jeffrey Pennington, Richard Socher and Christopher D. Manning, "Glove: Global vectors for word representation", Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532-1543, 2014.
[5] Reza Shokri, Marco Stronati, Congzheng Song and Vitaly Shmatikov, "Membership inference attacks against machine learning models", 2017 IEEE Symposium on Security and Privacy (SP), IEEE, pp. 3-18, 2017.
[6] Trishul Chilimbi, Yutaka Suzue, Johnson Apacible and Karthik Kalyanaraman, "Project adam: Building an efficient and scalable deep learning training system", 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), pp. 571-582, 2014.
[7] Le Trieu Phong and Tran Thi Phuong, "Privacy-preserving deep learning via weight transmission", IEEE Transactions on Information Forensics and Security, vol. 14(11), pp. 3003-3015, 2019.
[8] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge and Michael Moeller, "Inverting gradients- how easy is it to break privacy in federated learning?", Advances in Neural Information Processing Systems, vol. 33, pp. 16937-16947, 2020.
[9] Ligeng Zhu, Zhijian Liu and Song Han, "Deep leakage from gradients", Advances in Neural Information Processing Systems, vol. 32, pp. 14774–14784, 2019.
[10] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig and John Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy", International Conference on Machine Learning, PMLR, pp. 201-210, 2016.
[11] Hervé Chabanne, Amaury De Wargny, Jonathan Milgram, Constance Morel and Emmanuel Prouff, "Privacy-preserving classification on deep neural network", Cryptology ePrint Archive, 2017.
[12] Yoshinori Aono, Takuya Hayashi, Lihua Wang and Shiho Moriai, "Privacy-preserving deep learning via additively homomorphic encryption", IEEE Transactions on Information Forensics and Security, IEEE, vol. 13(5), pp. 1333-1345, 2017.
[13] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar and Li Zhang, "Deep learning with differential privacy", Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 308-318, 2016.
[14] Robin C. Geyer, Tassilo Klein and Moin Nabi, "Differentially private federated learning: A client level perspective", arXiv preprint arXiv:1712.07557, 2017.
[15] Eugene Bagdasaryan, Omid Poursaeed and Vitaly Shmatikov, "Differential privacy has disparate impact on model accuracy", Advances in Neural Information Processing Systems, vol. 32, 2019.
[16] Sagar Sharma, AKM Mubashwir Alam and Keke Chen, "Image Disguising for Protecting Data and Model Confidentiality in Outsourced Deep Learning", In 2021 IEEE 14th International Conference on Cloud Computing (CLOUD), pp. 71–77, 2021.
[17] Warit Sirichotedumrong, Takahiro Maekawa, Yuma Kinoshita and Hitoshi Kiya, "Privacy-Preserving Deep Neural Networks with Pixel-Based Image Encryption Considering Data Augmentation in the Encrypted Domain", 2019 IEEE International Conference on Image Processing (ICIP), pp. 674-678, 2019.
[18] Haokun Fang and Quan Qian, "Privacy preserving machine learning with homomorphic encryption and federated learning", Future Internet, vol. 13(4), pp. 94, 2021.
[19] Ehsan Hesamifard, Hassan Takabi and Mehdi Ghasemi, "Cryptodl: Deep neural networks over encrypted data", arXiv preprint arXiv:1711.05189, 2017.
[20] Qiang Zhu and Xixiang Lv, "2P-DNN: Privacy-preserving deep neural networks based on Homomorphic cryptosystem", arXiv preprint arXiv:1807.08459, 2018.
[21] Alberto Ibarrondo and Melek Önen, "Fhe-compatible batch normalization for privacy preserving deep learning", Data Privacy Management, Cryptocurrencies and Blockchain Technology, Springer, pp. 389-404, 2018.
[22] Ping Li, Jin Li, Zhengan Huang, Tong Li, Chong-Zhi Gao, Siu-Ming Yiu and Kai Chen, "Multi-key privacy-preserving deep learning in cloud computing", Future Generation Computer Systems, Elsevier, vol. 74, pp. 76-85, 2017.
[23] Liyue Fan, "Image pixelization with differential privacy", IFIP Annual Conference on Data and Applications Security and Privacy, Springer, pp. 148-162, 2018.
[24] Mahawaga Arachchige Pathum Chamikara, Peter Bertók, Ibrahim Khalil, Dongxi Liu and Seyit Camtepe, "Privacy preserving face recognition utilizing differential privacy", Computers & Security, Elsevier, vol. 97, pp. 101951, 2020.
[25] Warit Sirichotedumrong and Hitoshi Kiya, "A gan-based image transformation scheme for privacy-preserving deep neural networks", In 2020 28th European Signal Processing Conference (EUSIPCO), pp. 745-749, 2021.
[26] Ilya Loshchilov and Frank Hutter, "Sgdr: Stochastic gradient descent with warm restarts", arXiv preprint arXiv:1608.03983, 2016.
[27] Kuang Liu, Wei Yang, Peiwen Yang and Ducau Felipe, "Pytorch CIFAR-10 example codes", https://github.com/kuangliu/pytorch-cifar, last accessed: March 16, 2022.