12-2020

Differential privacy protection over deep learning: An investigation of its impacted factors

Ying LIN
Yunnan University

Ling-Yan BAO
Yunnan University

Ze-Minghui LI
Yunnan University

Shu-Sheng SI
Yunnan University

Chao-Hsien CHU
Singapore Management University, chchu@smu.edu.sg

Citation

LIN, Ying; BAO, Ling-Yan; LI, Ze-Minghui; SI, Shu-Sheng; and CHU, Chao-Hsien. Differential privacy protection over deep learning: An investigation of its impacted factors. (2020). *Computers & Security*. 99, 1-16.

Available at: https://ink.library.smu.edu.sg/sis_research/5402

Differential privacy protection over deep learning: An investigation of its impacted factors

Ying Lin[a], Ling-Yan Bao[a], Ze-Minghui Li[a], Shu-Zheng Si[a], Chao-Hsien Chu[b]

[a] School of Software, Yunnan University, China

[b] School of Information Systems, Singapore Management University, Singapore

Published in Computers & Security, December 2021, 99, Article No. 102061, pp 1-16.
https://doi.org/10.1016/j.cose.2020.102061

Abstract

Deep learning (DL) has been widely applied to achieve promising results in many fields, but it still exists various privacy concerns and issues. Applying differential privacy (DP) to DL models is an effective way to ensure privacy-preserving training and classification. In this paper, we revisit the DP stochastic gradient descent (DP-SGD) method, which has been used by several algorithms and systems and achieved good privacy protection. However, several factors, such as the sequence of adding noise, the models used etc., may impact its performance with various degrees. We empirically show that adding noise first and clipping second will not only significantly achieve high accuracy, but also accelerate convergence. Rigorous experiments have been conducted on three different datasets to train two popular DL models, Convolutional Neural Network (CNN) and Long and Short-Term Memory (LSTM). For the CNN, the accuracy rate can be increased by 3%, 8% and 10% on average for the respective datasets, and the loss value is reduced by 18%, 14% and 22% on average. For the LSTM, the accuracy rate can be increased by 18%, 13% and 12% on average, and the loss value can be reduced by 55%, 25% and 23% on average. Meanwhile, we have compared the performance of our proposed method with a state-of-the-art SGD-based technique. The results show that under the premise of a reasonable clipping threshold, the proposed method not only has better performance, but also achieve ideal privacy protection effects. The proposed alternative can be applied to many existing privacy preserving solutions.

Keywords: Differential privacy, Privacy preserving, Deep learning, Stochastic gradient descent (SGD)

1. Introduction

With recent major advances in Artificial Intelligence (AI) and the computational capability of computers, DL has achieved promising performance in many fields, such as data analytics, image classification, pattern recognition and health care, but its wide application also makes it a natural target for attackers (Papernot et al., 2016). In addition, its capability to capture and memorize elements of training data provides a convenient pathway for an attacker to determine whether a particular data record is in the training dataset through membership inference attack (Shokri et al., 2017). Hence, privacy assurance in DL has become a hot topic and many researchers have proposed various privacy-aware DL mechanisms over the past few decades (Liu et al., 2018).

☆ Corresponding author: On leave (2019-2020) from The Pennsylvania State University, University Park, PA 16802, USA. E-mail address: chu@ist.psu.edu (C.-H. Chu).

Privacy assurance in ML or DL commonly needs to address the following issues (Shokri and Shmatikov, 2015; Senavirathne and Torra, 2019): (1) privacy of the input data used to test the model or get a prediction; (2) privacy of the model itself; (3) privacy of the training data used to train the model; and (4) privacy of the model's output. Protecting data means preventing sensitive information from exposure, whereas, protecting the model usually means protecting the model's architecture and various parameters from illegal access and modification. There are two main avenues to protect privacy in ML or DL. One is encryption, represented by multi-party secure computing (Du and Atallah, 2001) and homomorphic encryption (Rivest et al., 1978), to protect sensitive information from exposure. Almost all of the encryption-based solutions require a fair amount of communications between the participating parties, so they are communication-bound (Badawi et al., 2018) and their ability to protect the neural networks themselves still remains an open problem. The other solution is perturbation. Various perturbation methods have been proposed for privacy protection. Input perturbation adds noise to the training data, and subsequent calculations are based on the noise-added data. Objective perturbation (Zhang et al., 2012; Phan et al., 2016; Phan et al., 2017) perturbs the objective function of the training model, which usually requires first deriving the approximate polynomial representation, so it is nontrivial. Output perturbation adds noise to the output of the model, such as in the PATE approach (Papernot et al., 2016). Compared with the above three methods, gradient perturbation receives a lot of attention as it can achieve DP guarantee even for nonconvex objectives. It is also the focus of this study.

Usually, gradient perturbation perturbs the computed gradients in each training step, and the whole training procedure can by guaranteed by applying the *composition property* (Dwork et al., 2006). In previous differentially private gradient computing mechanisms, the procedure of clipping the gradient was commonly performed before the procedure of adding noise. In this paper, we empirically found that changing the sequence of gradient clipping and adding noise can not only achieve high accuracy, but also accelerate convergence. Moreover, through our exploration and testing, we found that several other factors such as the types of learning models used, the gradient descent optimization methods, and model parameters etc., may impact the model's final performance with various degrees. As there has been no thorough analysis on whether those factors would make a difference, we examine these issues as well. Considering the trade-off between privacy protection and model utility, we use three metrics, convergence, classification accuracy and level of privacy protection (a new proposed metric) to evaluate the model's performance.

The purposes of this study and our main contributions are:

(1) We propose a generic *noisy gradient* (NG) method to handle the problem of privacy preserving DL. In NG, not only the conventional stochastic gradient descent algorithm, but also other adaptive gradient descent algorithms are taken into consideration. In addition, the relative importance of various parameters was studied. The efficiency and some merits of the proposed method was verified by rigorous experiments.
(2) By detailed empirical comparison experiments, we found that in the category of perturbing gradients to protect privacy of DL model, the sequence of adding noise first and clipping gradient second can achieve higher accuracy and faster convergence speed than when clipping gradient first and adding noise second. We give a detailed analysis on why changing the process sequence can achieve such effects.
(3) We implement two popular DL models, CNN and LSTM, with two different gradient descent optimization methods, using three datasets to show the applicability of our proposed method.
(4) We apply detailed statistical analysis methods to determine the impacts of various parameters, specifically the impact of *dataset, model, optimizer, noise, clipping*, and *sequence*, on the privacy preserving DL model's performance.
(5) In order to better study whether changing the process sequence would affect the model's privacy protection effects, we propose a new privacy protection metric called Total Parameters Value Difference (TPVD) and investigate the trade-off between privacy and utility under different parameter settings. Experimental results show that our proposed metric can achieve a better privacy guarantee while still obtaining higher utility by carefully calibrating input parameters.
(6) We compare our proposed method with a state-of-the-art SGD-based technique, which adaptively selects the clipping threshold during the training process. The results show that under the premise of a reasonable clipping threshold, the method proposed in this paper not only has better performance, but also achieve ideal privacy protection effects.

The rest of the paper is organized as follows: In Section 2, we present existing work related to gradient perturbation methods. In Section 3, we present the preliminaries and technical background associated with this paper. In Section 4, we introduce our proposed algorithm and present experimental setting. We present the experimental results, analyses and compare our work with a state-of-the-art gradient perturbation technique in Section 5. In Section 6, we use statistical analysis methods to analyze the impacts of various factors and parameters. In Section 7, we provide further improvements on privacy protection and discuss the implications. In Section 8, we conclude the paper and discuss future work.

2. Related work

The purpose of privacy preserving DL is to prevent adversaries from inferring accurate personal information through inversion attacks (Fredrikson et al., 2014; Fredrikson et al., 2015), membership attacks (Shokri et al., 2017; Hayes et al., 2019) or model extraction attacks (Tramèr et al., 2016; Wang and Gong, 2018). Gradient perturbation (Zhao et al., 2019) is one of the commonly used methods in privacy preserving DL.

Many early research focused on using encryption to prevent sensitive information from exposure. Dowlin et al.

(Gilad-Bachrach et al., 2016) presented a method to convert learned neural networks to *CryptoNets*, which can be applied to encrypted data to maintain data privacy and security. Hesamifard et al., (2018) presented a framework, named *CryptoDL*, for running deep neural network (DNN) algorithms over encrypted data. Using this, data owners can send their encrypted data to a cloud and then get an encrypted prediction in return.

There have been several efforts in gradient perturbation (Shokri and Shmatikov, 2015; Bassily et al., 2014; Abadi et al., 2016; Song et al., 2013; Lee, 2017). Early research on SGD-based gradient perturbation focused on adding noise to the output of the standard Empirical Risk Minimization (ERM) algorithm (Chaudhuri et al., 2011), which can produce privacy preserving approximations of classifiers. Song et al., (2013) proposed differentially private versions of single-point SGD and mini-batch SGD, but this paper did not examine how to track the privacy consumption of the entire training process. Shokri and Shmatikov, (2015) implemented a distributed selective stochastic gradient descent (DSSGD) method for collaborative DL. One key idea of DSSGD is that every participant shares a fraction of the parameters with other participants, which allows participants to benefit from other parties without sharing their own data. To minimize parameters leakage, they applied DP when updating parameters. However, if the numbers of training epoch, sharing parameters and participants are large, this method may consume a large portion of the privacy budget.

Abadi et al., (2016) improved the computational efficiency of differential privacy SGD (DP-SGD) and proposed a privacy accounting method called *moments accountant* to track cumulative privacy loss. The DP-SGD algorithm clips the gradients of the sample first to limit the sensitivity of each sample, and then adds noise to the gradients in batches before applying descent. Since this technique was proposed, it has been extended to a variety of situations. Our research work is also based on this technique to compare the influence of several factors that may impact the model's performance such as the sequence of adding noise, the types of models used, the gradient descent optimization methods, and the parameters of the model.

Xie et al., (2018) proposed a differentially private Generative Adversarial Network (DP-GAN) by adding noise to the gradients of the discriminator and then training a generator with the differentially private discriminator. At the end of the training, both discriminator and generator are DP. Acs et al., (2019) proposed a differentially private generative model (DP-GM), which is a composition of private kernel k-means and DP-SGD. In both private methods, the clipping threshold C was adapted to the gradient update of every batch to ensure fast convergence with small error. Yu et al., (2019) is also based on DP-SGD, but the authors proposed some new techniques, which effectively optimizes both model accuracy and privacy loss analysis.

From the recent work on applying gradient perturbation to achieve privacy, we can see that the work of DP-SGD (Abadi et al., 2016) method is the cornerstone. To achieve a trade-off between particular privacy vs. utility, some follow up work was done by adaptively choosing parameters, such as adaptively choosing clipping threshold C (Acs et al., 2019; Mcmahan et al., 2018) or by using different DP mechanism, such as f-differential privacy (Dong et al., 2019).

3. Preliminaries/Technical background

In this section, we briefly revisit some technical backgrounds of DP and two popular DL models, CNN and LSTM, which lay the foundation for our theoretical development.

3.1. *Differential privacy (DP)*

DP is defined in terms of the concept of adjacent databases, that is, two datasets D and $D'$are adjacent if they differ in at most one record. DP establishes a guarantee that a randomized algorithm behaves similarly on the two adjacent databases. The ε-DP was proposed by Dwork in 2006 (Dwork et al., 2006) to protect or preserve privacy at different levels of probability. This mechanism ensures that inserting or deleting a record in a data set does not affect any calculated output. In addition, ε-DP is robust against hackers who have auxiliary information. The ε-differential protection mechanism relies on incorporating random noise into the data (Dwork et al., 2006). The injected noise should be carefully calibrated and it can be generated by different mechanisms such as the Laplace mechanism (Dwork et al., 2006), the Exponential mechanism (McSherry and Talwar, 2007) and the Gaussian mechanism (Dwork and Roth, 2014). The Gaussian mechanism is usually used in the gradient perturbation approach.

Definition 1 (ε-*differential privacy* (Dwork et al., 2006)) **.** A random mechanism M provides ε-DP if for two adjacent databases D and D', and for all $O \in \text{range}(M)$, the following inequality holds:

$$Pr(M(D) \in O) \leq e^{\varepsilon} Pr(M(D') \in O) \tag{1}$$

Where, the parameter ε is defined as the privacy budget, which controls the privacy guarantee level. A lower ε means stronger privacy guarantee and more perturbation. Sensitivity determines how much perturbation is required for a random mechanism M. Global sensitivity and local sensitivity are two types of sensitivity that are employed in the DP.

Definition 2 (Global sensitivity (Dwork et al., 2006)). Given a function f: $D \rightarrow R^d$, for any two adjacent datasets D and D', the global sensitivity of f is defined as:

$$GS_f = \max_{D,\, D'} \| f(D) - f(D') \| \tag{2}$$

Definition 3 (*Local sensitivity* (Dwork et al., 2006)) **.** Given a function f: $D \rightarrow R^d$, for dataset D and its adjacent dataset D', the local sensitivity of f on dataset D is defined as:

$$LS_f = \max_{D'} \| f(D) - f(D') \| \tag{3}$$

Compared with the global sensitivity, local sensitivity is related to a specific database D, so it may result in information

disclosure. In this paper, sensitivity refers to l_2-global sensitivity.

Definition 4 (l_2-*global sensitivity*) **.** The l_2-global sensitivity of a function f is the maximum L_2 norm of the difference between $f(D)$ and $f(D')$, i.e.:

$$L2_f = \max_{D,\, D'} \| f(D) - f(D') \|_2 \tag{4}$$

In some situations, it is possible that the ε-*DP* is broken with probability δ, which is called the (ε, δ)-differential privacy.

Definition 5 ((ε, δ)-*differential privacy* (Dwork and Roth, 2014)) **.** A privacy mechanism M guarantees (ε, δ)-*DP* if for any two adjacent databases D and D', and for all $O \in range(M)$, the following inequality holds:

$$\Pr(M(D) \in O) \leq e^{\varepsilon} \Pr(M(D') \in O) + \delta \tag{5}$$

The (ε, δ)-DP allows that the privacy loss does not exceed ε with probability at most $1 - \delta$.

Definition 6 (*Gaussian mechanism* (Dwork and Roth, 2014)) **.** The Gaussian mechanism with parameter σ is to add independent and identically distributed (*i.i.d.*) Gaussian noise to a true output of a function f whose value is a k-dimensional vector, defined bellow:

$$M(D) = f(D) + \langle x_1, x_2, \ldots x_k \rangle \tag{6}$$

Where $x_1, x_2, \ldots x_k$ are *i.i.d.* random variables drawn from $N(0, \sigma^2 I)$.

Theorem 1 (*Dwork and Roth, 2014*) **.** *Let* $\varepsilon \in (0, 1)$, *the Gaussian mechanism with parameter* $\sigma > \sqrt{2ln(1.25/\delta}L2_f/\varepsilon$ *is* (ε, δ)*-DP.*

According to Theorem 1, the Gaussian mechanism can achieve (ε, δ)-DP guarantee as long as those parameters ε, δ and σ meet the above inequality conditions.

3.2. *Deep learning (DL)*

DL is a subset of ML based on artificial neural networks. Its learning methods can be either supervised, semi-supervised or unsupervised, which allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification (LeCun et al., 2015). Many applications of DL use feedforward neural networks, which are composed of several layers of transformation of the form $F^i(x) = g(W^i \bullet F^{i-1}(x))$, where the i_{th}-layer F^i takes the results of previous layers F^{i-1} as inputs, W^i is a matrix of the parameters and g represents the non-linear activation function. Let $Z(x)$ denote the output of the last layer (usually before the softmax), i.e., $Z(x) = F^n(x)$, then the final output of the network is $F(x) = soft\max(Z(x))$. In this paper, we will use two popular DLs, the CNN and LSTM, for exploration.

3.2.1. *Convolutional neural network (CNN)*

The CNN is a variation of a feedforward neural network, which has shown excellent performance in many ML problems (Kandi et al., 2017). One processing step in a CNN is usually called a layer. The convolution layer, pooling layer, and fully connected layer are three main types of layers in CNN architectures.

(1) *Convolutional layer*: The convolutional layer is the major building block used in a CNN. In a convolutional layer, multiple convolution kernels (filters) are used to extract features from the raw input data. Convolution is a mathematical operation that computes the dot product between the input and the entries of the filter, which produces an activation map of that filter. Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer.
(2) *Pooling layer*: The pooling layer is used for down sampling feature maps to reduce the number of parameters and computations in the network. The pooling layer operates independently on every depth slice of the input and resizes it spatially by using two common pooling methods: *average pooling* and max *pooling*. The output of the pooling layer is a summarized version of the features detected in the input.
(3) *Fully connected layer*: In the fully connected layer, each neuron receives input from every element of the previous layer. A fully connected layer outputs a vector of length equal to the number of neurons in the layer and is usually built in the last layer of a CNN. This part is in principle the same as a regular neural network.

3.2.2. *Long short-term memory (LSTM)*

Recurrent Neural Networks (RNN) have been widely used to deal with variable-length sequence inputs with the output being dependent on the previous computations. LSTM, proposed by Hochreiter and Schmidhuber in 1997 (Hochreiter and Schmidhuber, 1997), is a special kind of RNN. The key element of LSTM is the cell state, which is controlled by three kinds of gates: input, output and forget gates. The following equations give the step-by-step update for the current cell, where σ stands for the sigmoid function and W_f, W_i, W_c, W_o, b_f, b_i, b_c, b_o are parameters of the network.

The first step in LSTM is that the forget gate reads the input of h_{t-1} (the output of the previous unit) and x_t (the input of the current unit), and then outputs a value between *0* and *1* as shown in (7). The next step includes two parts to decide what new information needs to be stored in the cell state. Firstly, as shown in (8), the input gate decides which values need to be updated. Then as shown in (9), a *tanh* function generates $\tilde{C}_t$. Based on these previous steps, the old cell state C_{t-1} can be updated to a new state C_t as shown in (10). Finally, it is time to decide the output value. As shown in (11), an output gate gets o_t to decide what parts of the cell state needs to be output, then multiplies o_t with a *tanh* function of the new state C_t as shown in (12), so that it only outputs the needed parts.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{7}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{8}$$

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{9}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{10}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{11}$$

$$h_t = o_t * \tanh(C_t) \tag{12}$$

3.3. *Gradient descent optimization*

In most DL models, the loss function $L(W,X)$ is calculated as the difference between the actual output and the predict output. The gradient is the partial derivative of the loss function with respect to weights. Gradient descent is one of the popular algorithms used to perform optimization by updating the weights of the learning models. The weights are updated in the opposite direction of the calculated gradient and this cycle is repeated until reaching the minima of the loss function as shown in (13).

$$W^{(k+1)} = W^{(k)} - \frac{\partial L(W, X)}{\partial W^{(k)}} \tag{13}$$

Stochastic gradient descent (SGD) belongs to a type of constant learning rate algorithms, which are the most widely used to optimize function.

$$W^{(k+1)} = W^{(k)} - \eta * \frac{\partial L(W, X)}{\partial W^{(k)}} \tag{14}$$

Where η is called the learning rate which is a hyper parameter that needs to be properly tuned. The challenge of using gradient descent is that their hyper parameters must be defined in advance and they depend heavily on the type of model and problem considered. Another problem is that the same learning rate is applied to all parameter updates. Adaptive learning algorithms provide an alternative to the classical SGD. They have per-parameter learning rate methods, which provide heuristic approach without requiring expensive work in manually tuning hyper parameters for the learning rate schedule. Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012) and Adam (Kingma and Ba, 2014) are the most widely used adaptive gradient descent algorithms.

4. Materials and methods

4.1. *The proposed noisy gradient (NG) algorithm*

In this section, we formally present our mechanism; the pseudo-code of Algorithm 1 outlines four basic steps in our mechanism. The main process of implementing the NG algorithm includes the following steps:

First, we compute the gradient. This is achieved by selecting a random subset of examples and computing the gradient for each sample in this subset at every training step t. Second, we add noise. This is achieved by adding Gaussian noise to each gradient of a batch to obtain noised gradient. Thirdly, we perform gradient clipping. If the L_2 norm for the noised gradient exceeds the threshold value C, then the values in the vector will be rescaled, so that the L_2 norm of the noised gradient equals C; otherwise, keep the original noised gradient unchanged. The final step is gradient optimization. This is achieved by taking SGD, Adam, or other optimization methods.

Algorithm 1 – Noisy Gradient (NG) Algorithm.

Input: examples $\{x_1, x_2, \ldots x_N\}$, Loss function, parameters: σ, L, C, T
1: Initialize w_0 randomly
2: For $t \in T$ do
3: Take a random sample L_t with sampling probability L/N
4: For each $i \in L_t$, compute $g_i \leftarrow \nabla_{w_t} L(w_t, x_i)$
5: $\tilde{g}_i \leftarrow g_i + N(0, \sigma^2 C^2 I)$
6: $g_i \leftarrow \tilde{g}_i / max(1, \frac{\|\tilde{g}_i\|_2}{C})$
7: $\tilde{g}_t \leftarrow \frac{1}{L} \sum_i g_i$
8: Optimizer.Apply_gradients($\tilde{g}_t$)
9: End

4.2. *Theoretical rationality*

Our proposed algorithm is based on the work in (Abadi et al., 2016), which proposed a differentially private SGD (DP- SGD) algorithm. The main difference between our proposed algorithm and the DP-SGD is in the sequence of clipping gradient and adding noise. The DP-SGD algorithm first clips every computed gradient in a batch to ensure the L_2 norm of the gradient is within a threshold C and then adds Gaussian noises to the total sum of gradients in the batch. In our proposed algorithm, we add Gaussian noises to each computed gradient in a batch first and then clip the noised gradient to ensure the L_2 norm of the noised gradient is within a threshold C. In our explorations, we observed that although the L_2 norm of a gradient is forced to a threshold C after the step of clipping in the DP-SGD, the subsequent step of adding noise breaks this limit as long as the random noise is large enough. Although noisy updates help in finding new and better local minima, larger fluctuations will cause the optimization algorithm to go beyond the better local minima and continue overshooting close to the desired exact minima. The modification of adding noise before clipping can ensure the L_2 norm of the final gradient stays within the threshold C even if adding a larger random noise first.

Although our improved method only changes the sequence of clipping and perturbation, compared with the DP-SGD, it differs in that the noisy gradients updates is guaranteed to be within a certain threshold. **Adding noise into gradients before clipping results in higher accuracy and faster convergence than adding noise after gradient clipping.** Moreover, **adding bounded noise improves learning for DL models.** This finding constitutes the main contribution of this paper. In fact, adding random noise to the weights, gradient, and the hidden units has also been used when training neural networks (Graves, 2011; Blundell et al., 2015). In the work of (Neelakantan et al., 2015), the authors empirically illustrated that injecting noise into gradients guarantees stochasticity, and can actually achieve lower training loss. They showed that adding noise encourages active exploration of the parameter space and gives the model more chances to escape local minima or to traverse quickly through the stationary stage of early learning. These features can result in lower training loss, which helps to achieve a higher accuracy.

After adding random noise to the gradient, we apply gradient clipping. Gradient clipping was introduced in (Bengio et al.,

2013) to avoid the gradient explosion problem. There are many ways to compute gradient clipping. Two commonly used approaches are gradient norm clipping and gradient value clipping. The proposed NG algorithm uses the first method. In our approach, clipping noisy gradients to a certain threshold not only avoids the problem of gradient explosion, but also improves model's convergence. Although the theory behind it is still largely unknown, it is a fact that gradient clipping helps in convergence. Many studies such as (Bottou, 1998; Carmon et al., 2017) are striving to provide a theoretical explanation for the effectiveness of gradient clipping in training DNNs.

4.3. *The construction of CNN and LSTM*

4.3.1. *CNN model*

The CNN model we build for our experiments contains one input layer, two convolutional layers, two pooling layers, one fully connected layer, one dropout layer and an output layer.

The input layer of the CNN is set to $28\times28\times1$, which is consistent with the picture format of the MNIST dataset. Convolution layer is basically filtering the image with a smaller pixel filter. We use 32 5×5 filters for the first convolution layer and 64 5×5 filters for the second convolution layer. For the images in the training set, filters step over the entire image. At each step the window is moved by 1 stride. When constructing CNNs, it is common to insert pooling layers after each convolution layer. We select a pooling size to reduce the amount of parameters by using the max-pooling method with strides of 2 and kernel size of 2. Finally, we construct two fully connected layers at the end to classify our images. The first fully connected layer uses 1024 neurons. It receives input from the previous layer and those neurons are randomly dropped out during training. The final fully connected layer uses 10 neurons to represent which category the image belongs to.

As MNIST data input is a *1-dimensional* vector of 784 features (28×28 pixels), it can easily be reshaped in the input layer to match the $28\times28\times1$ requirement. However, for the CIFAR10 and SVHN datasets, because the image format of both data sets is $32\times32\times3$, we first need an image format pre-processing step to change it from a $32\times32\times3$ color image to a $28\times28\times1$ grayscale image in order to train these three different datasets on a uniform CNN architecture.

4.3.2. *LSTM model*

For processing images, the input of the LSTM network model constructed in this paper is a 28×28 gray image, which can be viewed as a matrix with 28 rows and 28 columns. We expand the network in 28 time-steps so that in each time step, we can enter a row of 28 pixels, thus inputting the image after 28 time-steps. In the hidden layer, a LSTM cell replaces the common hidden unit in the RNN. We set a LSTM cell with the number of units as 128. The number of units in a LSTM cell can be interpreted as analogous to a hidden layers from the RNN. We set a batch size of 128, which means that every time step will be supplied with a respective row of 128 images. At every time-step, each LSTM cell will generate a tensor of shape [128, 128], but we are only concerned with the output of the final time-step. Thus, in the last time-step, we will convert the final output of shape [128,128] to [128, 10] so that the correct class can be predicted.

To allow the image formats of CIFAR10 and SVHN to satisfy the input requirement of the LSTM, we also need to do image pre-processing work before training.

4.4. *Datasets*

To verify the effectiveness of our proposed technique, we use three datasets, MNIST (LeCun, 1998), CIFAR10 (Krizhevsky et al., 2014) and SVHN (Netzer et al., 2011) for experiments and testing. The MNIST has been widely used as a standard ML benchmark in pattern recognition for over two decades (Baldominos et al., 2019; Das, 2017). The MNIST dataset includes 70,000 28×28 grayscale images of handwritten digits from 0 to 9, in which 60,000 are for training and 10,000 for testing. The CIFAR-10 dataset includes 60,000 32×32 color images of ten different categories for airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships and trucks, and each category has 6000 images. The SVHN is a real-world image dataset, which includes over 600,000 digital images of street view numbers, from which we use 100,000 for training and 10,000 as test examples.

4.5. *Performance metrics*

Three metrics, convergence, classification accuracy and level of privacy protection, are used to evaluate the model's performance.

4.5.1. *Convergence*

The loss function reflects the degree to which the model fits the data. In general, the smaller the loss value, the better the model fits the data. To better illustrate the difference between the predicted value and the true value when we run different optimization methods under privacy-preserving guarantee, we use convergence as one of the performance metrics.

Definition 7 (*Convergence*). Suppose $L(W, X)$ is the loss function of a DL model with parameters W and input. For any $\varepsilon > 0$, for each $t \in 0, 1, ...T-1$, $W^{t+1} = W^t - \eta \frac{1}{|S_t|} \sum_{i \in S_t} L(W^t, x_i)$, if it satisfies with probability over the randomness of $S_1, S_2, ..., S_T$: $L(W^t, X) \leq \varepsilon$, then we say the loss function is convergent.

4.5.2. *Classification accuracy*

Although a few other metrics such as sensitivity, specificity, accuracy, precision and F1 score have been used in assessing the relative performance of ML/DL algorithms, we use accuracy to measure classification accuracy in this paper. It is the ratio of number of correct predictions to the total number of input samples. The higher the classification accuracy, the higher the model's utility.

$$accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions\ made} \tag{15}$$

4.5.3. *Privacy protection*

DP generally uses (ε, δ) as the metrics for privacy protection. In this paper, we propose another metric, *Total Parameters Value Difference (TPVD)*, to measure the privacy protection capability

of our proposed algorithm, because we want to examine from another perspective how the impact of adding noise in the training process will affect the model itself. It is well known that when the training process is over, the released weights can be used to represent the DL model in some sense. Because of this, measuring the difference between the weights obtained from a training process that has no privacy protection and the weights obtained from training with noise perturbation can better reflect the protection of the model itself.

Let $W = (W^1, W^2, \ldots W^n)$ and $\bar{W} = (\bar{W}^1, \bar{W}^2, \ldots \bar{W}^n)$ be the weights of a DL model whose training process has no privacy protection and has privacy protection respectively. Each component W^i or $\bar{W}^i$ is a matrix of some weights. Considering that the value difference (VD) (Wang et al., 2006) is commonly used to measure the level of data distortion between two matrices, we use it to evaluate the level of data distortion between the original matrix W^i and its distorted counterpart $\bar{W}^i$ as shown in (16).

$$VD^i = \frac{\left\| W^i - \bar{W}^i \right\|_F}{\left\| W^i \right\|_F} \tag{16}$$

Where the $\|W\|_F$ is the Frobenius norm of a n^*m-dimensional matrix W. It is the sum of the absolute value squared of each element in W as shown in (17):

$$\|W\|_F = \sqrt{\sum_{j=1}^{n} \sum_{i=1}^{m} \left| v_i^j \right|^2} \tag{17}$$

Based on the VD calculation of each component W^i and its counterpart $\bar{W}^i$, we can get the TPVD between Wand $\bar{W}$ as shown in (18):

$$TPVD = \frac{\sum_{i=1}^{n} VD^i}{n} \tag{18}$$

4.6. *Experimental design*

4.6.1. *Experimental parameter settings*

We design several experiments to compare and verify the performance impacted by different factors. First, we consider the sequence of adding noise, before or after gradient clipping, as it can influence the convergence performance and prediction accuracy of DL models. Second, to better demonstrate the versatility of our proposed method, we use two DL models, CNN and LSTM, and adopt two gradient descent optimization methods, SGD and Adam, for evaluation. Parameters considered in the experiments include Gaussian noise scale σ and clipping threshold C. Table 1 lists the key experimental factors and their levels considered in the experimental design.

4.6.2. *Hardware and software environment*

Table 2 lists the hardware and software environment we used in these experiments.

Table 1 – Experimental factors and their levels.

Experimental Factors	Levels	Values
Datasets	3	MNIST, CIFAR10, SVHN
DL models (M)	2	CNN, LSTM
Gradient descent optimization method (O)	2	SGD, Adam
Sequence of adding noise (S)	2	Adding noise then Clipping (AC), Clipping then Adding noise (CA)
Parameter σ (N)	2	0.1, 0.5
Parameter C (C)	2	0.8, 1.4

Table 2 – Hardware and software environment.

OS	Windows 10
RAM	16 G/8G
Graphics card	GTX1070/GTX1060/GTX950
Graphics memory	8 G/4 G/2G
Python	Python 3.6
IDE	Pycharm 2018
Anaconda	Anaconda 3.6
Tensorflow	Tensorflow-1.11.0 GPU
Keras	Version 2.2.4

5. Performance evaluation, results and analyses

5.1. *Experimental results with different parameter settings*

By setting different parameters, we conducted model training under different parameter combinations. Tables 3–5 show the experimental results by training on MNIST, CIFAR10 and SVHN datasets respectively. The best result of each column is highlighted in boldface and the worst result is shown in red color or italic format.

From the experimental results, we can see that both CNN and LSTM achieved a high accuracy and a fast convergence effect on the MNIST dataset. The training accuracy can be as high as *97%* even under perturbation. However, in contrast, the performance on CIFAR10 and SVHN is not so good. The reason is mainly because we performed image cropping and grayscale processing to meet the input format requirements of the CNN model and LSTM model constructed for this study. However, we can still see that adding noise first and then clipping second (AC) performs better than clipping first and then adding noise (CA) no matter which dataset is used.

In general, for the CNN, the accuracy rate can be increased by *3%, 8%* and *10%* on average for the respective datasets, and the loss value is reduced by *18%, 14%* and *22%* on average. For the LSTM, the accuracy rate can be increased by *18%, 13%* and

Table 3 – Training results on MNIST dataset.

	N1 (0.1)		N2 (0.5)	
Factors	Accuracy	Convergence	Accuracy	Convergence
M1O1C1S1	0.9662	0.1407	0.8684	0.9671
M1O1C1S2	0.9334	0.4919	0.8569	1.1841
M1O1C2S1	**0.9714**	**0.1002**	0.9119	0.6684
M1O1C2S2	0.9492	0.4387	0.9049	0.9732
Baseline (No Noise)	0.9839	0.0267		
M1O2C1S1	0.9467	0.1481	0.9136	0.1899
M1O2C1S2	0.9213	0.2019	0.8729	0.2620
M1O2C2S1	0.9472	0.1353	**0.9432**	**0.1881**
M1O2C2S2	0.9373	0.2031	0.8595	0.2149
Baseline (No Noise)	0.9794	0.036		
M2O1C1S1	0.9297	0.2280	0.8203	0.6098
M2O1C1S2	0.7734	0.7170	0.6016	1.2670
M2O1C2S1	0.9063	0.3620	0.7891	0.7453
M2O1C2S2	0.7500	0.7319	0.6328	1.2508
Baseline (No Noise)	0.9921	0.0282		
M2O2C1S1	0.9375	0.1918	0.8125	0.6638
M2O2C1S2	0.8203	0.7099	0.5625	1.3434
M2O2C2S1	0.8438	0.3687	0.8125	0.7325
M2O2C2S2	0.8437	0.5022	0.6328	1.2798
Baseline (No Noise)	0.9843	0.03		

Table 4 – Training results on CIFAR10 dataset.

	N1 (0.1)		N2 (0.5)	
Factors	Accuracy	Convergence	Accuracy	Convergence
M1O1C1S1	0.2942	2.1561	0.2858	2.2064
M1O1C1S2	0.2042	2.3759	0.1443	2.4367
M1O1C2S1	0.3181	2.0627	0.2794	2.1856
M1O1C2S2	0.2155	2.2955	0.1705	2.4608
Baseline (No Noise)	0.313	2.0764		
M1O2C1S1	**0.4272**	1.8739	0.2627	2.0944
M1O2C1S2	0.4034	**1.6298**	0.2850	2.1836
M1O2C2S1	0.4147	1.8396	**0.2864**	2.1372
M1O2C2S2	0.2585	2.1574	0.2585	2.1574
Baseline (No Noise)	0.6757	0.3431		
M2O1C1S1	0.33	1.999	0.38	**1.9848**
M2O1C1S2	0.2	2.198	0.15	2.2612
M2O1C2S1	0.24	2.0408	0.19	2.2414
M2O1C2S2	0.2	2.1963	0.1	2.5285
Baseline (No Noise)	0.41	1.6709		
M2O2C1S1	0.35	1.8632	0.28	2.1646
M2O2C1S2	0.16	2.1291	0.16	2.4350
M2O2C2S1	0.27	1.9748	0.28	2.1646
M2O2C2S2	0.16	2.2182	0.16	2.4350
Baseline (No Noise)	0.45	1.6502		

12% on average, and the loss value can be reduced by 55%, 25% and 23% on average.

In order to compare the impact on the model's privacy protection capability under different parameter settings, we compute the TPVD values, which can visually reflect the perturbation effects on a trained model. The following calculation results are based on the data collected from training on the MNIST dataset. The calculations based on the other two training datasets are similar.

Table 6 shows the values of a set of TPVD results and accuracy under different clipping size C and sequence. It can be seen that adding noise before gradient clipping (AC) really improves the accuracy of the model when the other parameter settings are the same. However, the TPVD values of AC are lower than those of CA. For example, the accuracy and TPVD value are 91.36% and 2.1489 respectively when the *noise* is 0.5, *clipping* size is 0.8, *model* is CNN, *optimizer* is Adam, and *sequence* is AC. However, the accuracy and TPVD value are 87.29% and 2.3202 respectively when the parameters are the same except for the *sequence* being CA. It is obvious that the TPVD value increases as the clipping size increases. The larger the clipping size, the larger the TPVD value. According to the definition of

Table 5 – Training results on SVHN dataset.

	N1 (0.1)		N2 (0.5)	
Factors	Accuracy	Convergence	Accuracy	Convergence
M1O1C1S1	0.2565	2.2249	0.2987	2.1233
M1O1C1S2	0.2221	2.2271	0.1893	2.4867
M1O1C2S1	0.2094	2.1836	0.2161	2.3241
M1O1C2S2	0.1	2.4208	0.1941	2.4219
Baseline (No Noise)	0.3213	2.1124		
M1O2C1S1	0.4219	**1.7289**	0.2578	2.0687
M1O2C1S2	0.2422	2.0998	0.1641	2.3081
M1O2C2S1	0.3281	1.9632	0.2813	2.0384
M1O2C2S2	0.1875	2.1340	0.1953	2.2913
Baseline (No Noise)	0.6172	0.8306		
M2O1C1S1	**0.4141**	1.7556	0.2188	2.2073
M2O1C1S2	0.1719	2.2120	0.2266	2.2381
M2O1C2S1	0.3281	1.9177	**0.3047**	**2.0124**
M2O1C2S2	0.2578	2.0276	0.1953	2.2098
Baseline (No Noise)	0.7031	0.8331		
M2O2C1S1	0.4219	**1.7289**	0.2578	2.0696
M2O2C1S2	0.2422	2.0998	0.1641	2.3081
M2O2C2S1	0.3281	1.9632	0.2578	2.0686
M2O2C2S2	0.1875	2.1340	0.1641	2.3081
Baseline (No Noise)	0.7344	0.8391		

Table 6 – TPVD and accuracy results under different parameters setting with MNIST.

Evaluation	Model	CNN (Adam)		LSTM (Adam)	
	(Noise, Clipping)	(0.5,0.8)	(0.5,1.4)	(0.5,0.8)	(0.5,1.4)
Accuracy	AC	91.36%	94.32%	81.25%	81.25%
	CA	87.29%	85.95%	56.25%	63.28%
TPVD	AC	2.1489	2.2032	1.1411	1.1512
	CA	2.3202	2.3716	1.1627	1.1347

TPVD, the larger the TPVD value, the better the perturbation effect on the weights.

5.2. *Visualization of selected experiments results*

We plot and depict various results in the figures below. We only show a portion of the results for illustration. Fig. 1 depicts accuracy and loss values under different sequences with the MNIST *dataset*,σ of 0.5, C of 1.4, CNN *model*, and Adam *optimizer*. Fig. 2 depicts accuracy and loss values under different sequences with the CIFAR10 *dataset*,σ of 0.1, C of 0.8, CNN *model*, and Adam *optimizer*. Fig. 3 depicts accuracy and loss values under different sequences with the SVHN *dataset*,σ of 0.1, C of 1.4, CNN *model*, and Adam *optimizer*.

In Figs. 1-3, (a) depicts the accuracy results and (b) depicts the loss values. No matter which parameters are used, the

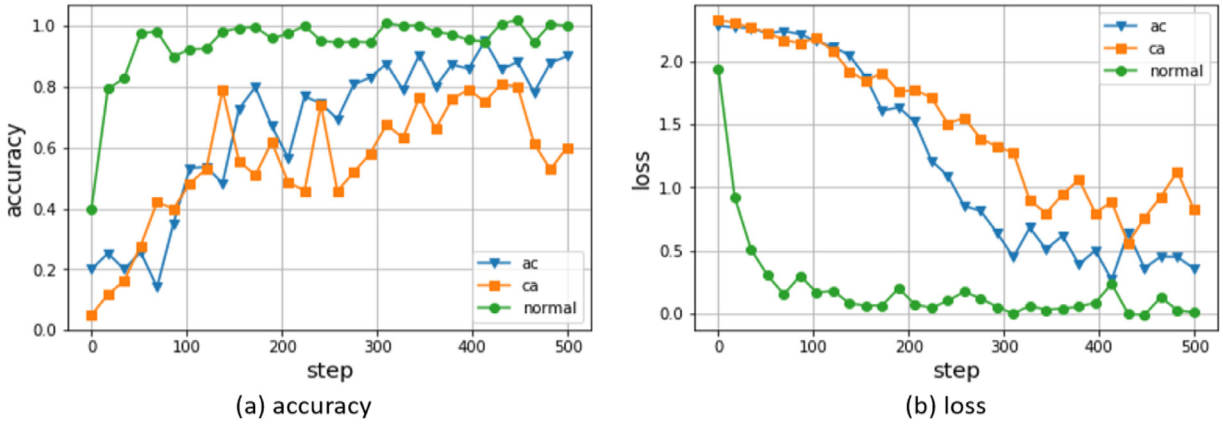


Fig. 1 – Accuracy and loss values of training MNIST dataset under different sequences.

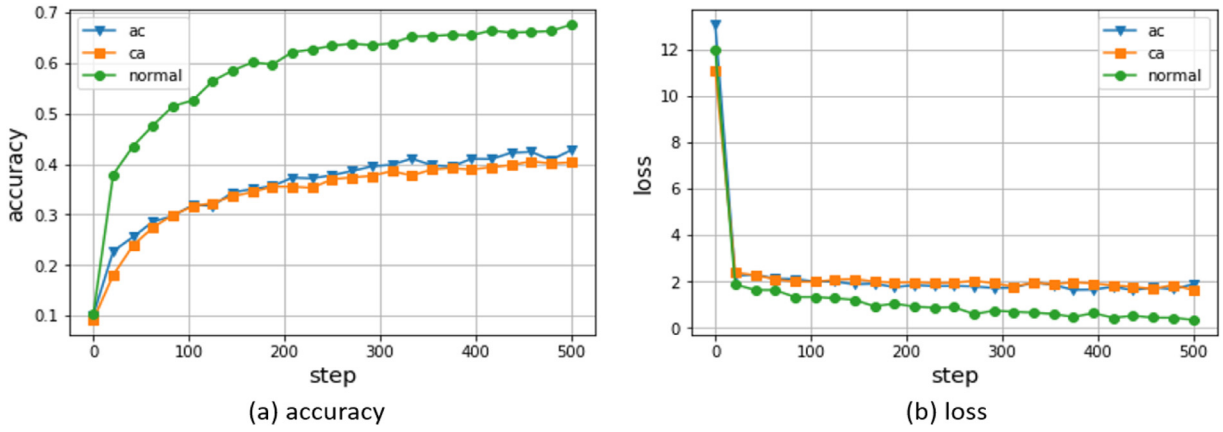


Fig. 2 – Accuracy and loss values of training CIFAR10 dataset under different sequences.

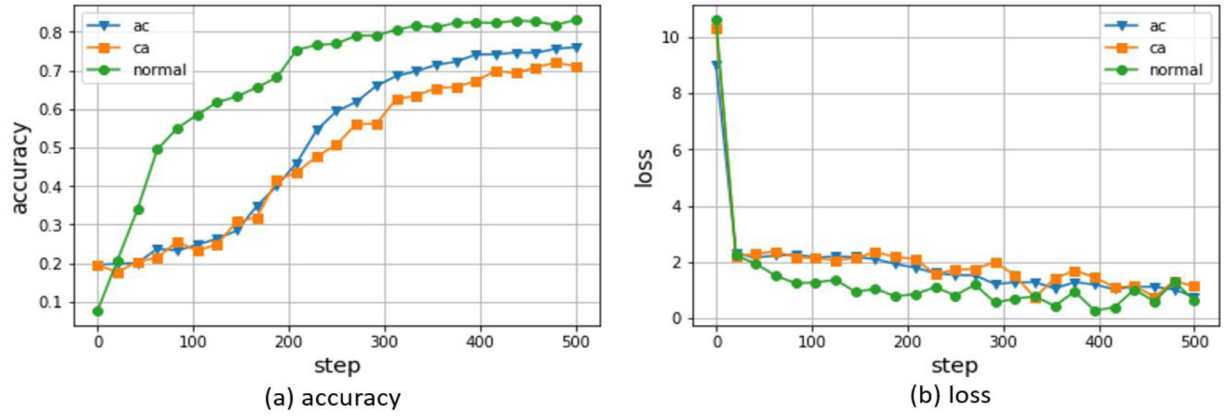


Fig. 3 – Accuracy and loss values of training SVHN dataset under different sequences.

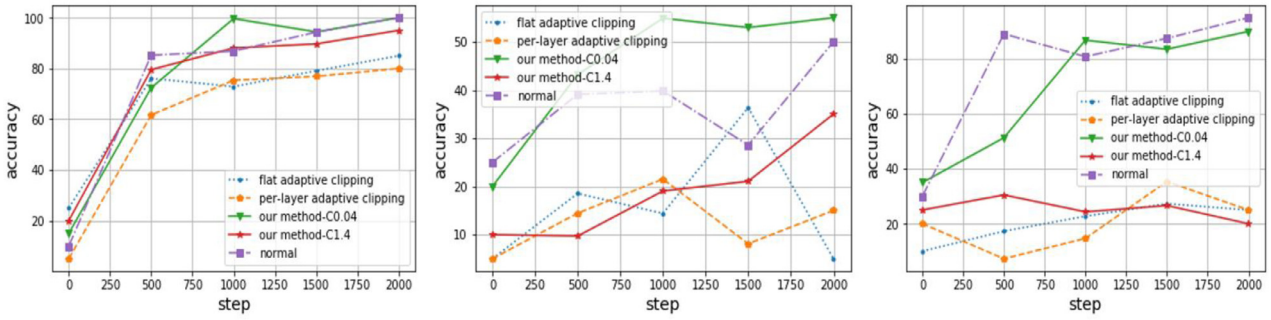
model with no privacy protection (normal) has the higher accuracy and the faster convergence speed than the model with privacy protection. The reason is obvious, as the gradient descent is a reliable process when there is no perturbation on the gradient. From Figs. 1-3(a), we can see that the accuracy values under the sequence of AC are higher than the accuracy values under the sequence of CA, which means the sequence of AC can bring higher accuracy. It is the same for the loss values under the sequence of AC, loss values are smaller than the loss values under the sequence of CA, which means the sequence of AC can bring faster convergence speed.

5.3. *Comparison with a state-of-the-art SGD-based approach*

As discussed in Section 2, there has been a lot of work in designing differentially private gradient perturbation algorithms for training ML models. SGD and its variations such as Adam, Adagrad and Adadelta are modified to provide privacy protection. But many of those approaches, including ours, set a fixed clipping threshold throughout the learning process. At present, some work such as (Thakkar et al., 2019; Pichapati et al., 2019) are trying to adaptively set hyperparameters for obtaining good utility. In this section, we compare our work with a state-of-the-art adaptive gradient clipping technology.

To adaptively adjust the value of the clipping threshold, (Thakkar et al., 2019) proposed a quantile clipping strategy, which obtains the clipping threshold for the next round by computing the quantile of the current round's gradients. (Thakkar et al., 2019) applied this strategy separately to two methods as *flat clipping* and *per-layer clipping*. Flat clipping adaptively calculates an overall clipping threshold C and clips the concatenation of all layers. Per-layer clipping calculates a per-layer clipping threshold C_j for each layer j and clips each layer separately. As (Thakkar et al., 2019) proposed, this strategy mainly considers the two settings of federated SGD and federated averaging, which are different from our setting of centralized learning. For comparison, we implemented differentially private learning with an adaptive clipping algorithm according to the quantile clipping strategy, which is shown in Algorithm 2. The main process of Algorithm 2 includes four steps: firstly, computing gradient; secondly, adaptively computing quantile clip threshold; thirdly, gradient clipping in flat clipping or per-layer clipping; finally, gradient optimization.

Three datasets MNIST, CIFAR10 and SVHN are used to train the CNN model. The CNN model has the same architecture in the previously discussed experiments. For comparison, we set our clipping threshold to 0.04 and 1.4 (via pilot tests). Further-



(a) MNIST dataset (b) CIFAR10 dataset (c) SVHN dataset

Fig. 4 – Accuracy of training on three datasets.

more, we also train the CNN model with no privacy protection as a reference. In these experiments, the quantile is set to 0.5 and the Adam optimizer is used. The experimental results show that by setting a reasonable constant C, our method is superior to the adaptive quantile clipping method mentioned in (Thakkar et al., 2019) in terms of loss convergence and accuracy. As shown in Figs. 4 and 5, our proposed method achieves good performance when C is set 0.04, as the green lines show in the figures. However, we also find that as the threshold C gradually increases, the performance of our proposed method gradually approaches that of the benchmark method.

We also compared the TPVD values of our proposed method and the benchmark methods, see Table 7. As can see that although the TPVD value of our proposed method is smaller than that of the benchmark adaptive clipping method, its performance is far better. For example, when C is set to 0.04, although the TPVD value is *1.5095*, which is smaller than adaptive clipping's *1.9679* and *2.2126*, the accuracy reaches *84.43%*, which is much higher than adaptive clipping's *20.57%* and *19.31%*.

Algorithm 2 – Differentially Private Learning with Adaptive Clipping.

Input: examples $\{x_1, x_2, \ldots x_N\}$, Loss function, parameters:σ, L, T, quantile
1: Begin
2: Initialize w_0 randomly
3: For $t \in T$ do
4: Take a random sample L_t with sampling probability L/N
5: For each $i \in L_t$, compute $g_i \leftarrow \nabla_{w_t} L(w_t, x_i)$
6: Quantile_C = get_quantile_norm (quantile)
7: $\bar{g}_i$= ClippingFn(g_i, Quantile_C) // FlatClip or PerlayerClip
8: $\tilde{g}_t \leftarrow \frac{1}{L}(\sum_i \bar{g}_i + N(0, \sigma^2 C^2 I)$
9: Optimizer.apply_gradients($\tilde{g}_t$)
10: End
11: Function FlatClip(Δ, C)
12: $\Delta' = \Delta / \max(1, \frac{\|\Delta\|_2}{C})$
13: Return (Δ')
14: Function PerlayerClip(Δ, C)
15: $C(j) = \frac{\|\Delta\|_2}{m}$
16: For each layer $j \in |m|$ do
17: $\Delta'(j) = \Delta(j) / \max(1, \frac{\|\Delta(j)\|_2}{C(j)})$
18: Return(Δ')

6. Analyses of impacted factors - Statistical Analysis

To further evaluate the relationship of the accuracy and convergence results with other different factors, we performed the following statistical analysis with SPSS 24.0 (Cor, 2016).

6.1. *Normality test*

When working with a sample of data, the normality test can be used to decide whether to use parametric or nonparametric statistical methods. Parametric statistical methods assume that the data has a specific distribution, commonly a normal distribution. If a data sample is not normally distributed, then nonparametric statistical methods must be used. Therefore, we first need to test whether our experimental data are in a normal distribution or not. There are many normality test methods available, such as the Shapiro-Wilk test, the Anderson-Darling test and the Kolmogorov-Smirnov test, etc. Considering that the Shapiro-Wilk test is a powerful test in most situation and it is appropriate for smaller samples of data, we decided to use the Shapiro-Wilk test as our numerical means of assessing normality.

6.1.1. *Accuracy*

First, the Shapiro-Wilk normality test was performed on dependent variable *accuracy* with other independent variables *dataset, model, optimizer, noise, clipping* and *sequence*. The test statistics are shown in Table 8. The null hypothesis for this test of normality is that *the data are normally distributed*, and the null hypothesis is rejected if the p-value (Sig.) is below 0.05. We can see that all the p-values are below 0.05 except for when the *dataset* is CIFAR10, so we can reject the hypothesis and conclude that the variable *accuracy* is not normally distributed except for the situation when the *dataset* is CIFAR10.

6.1.2. *Convergence*

Second, the Shapiro-Wilk normality test was performed on dependent variable *convergence* with other independent variables *dataset, model, optimizer, noise, clipping* and *sequence*. The test results are shown in Table 9. We can see that the p-value 0.334 and 0.335 when the *dataset* is CIFAR10 and SVHN. In other cases, the p-values are all less than 0.05. Therefore, we

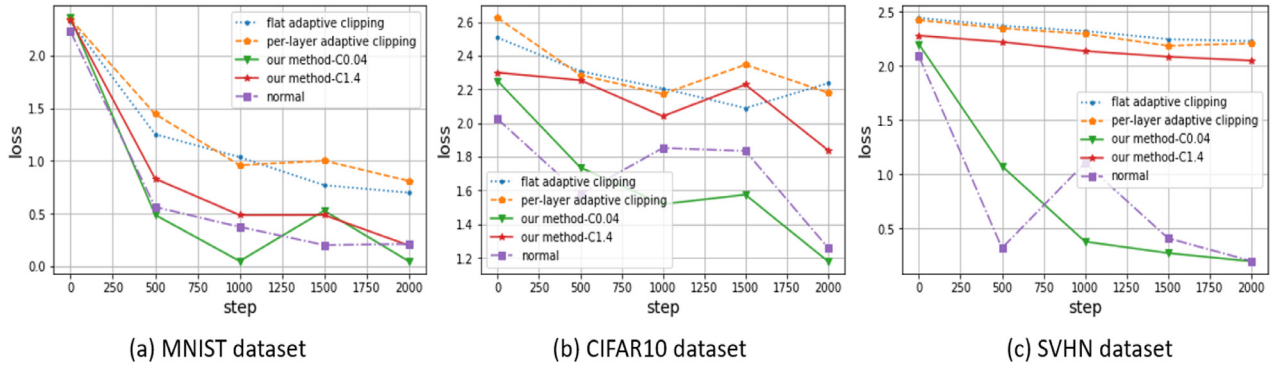


Fig. 5 – Loss values of training on three datasets.

Table 7 – TPVD and accuracy results trained on different datasets.

Datasets	Performance Metric	Per-layer adaptive clipping	Flat adaptive clipping	Our method C = 0.04	Our method C = 1.4	Normal
MNIST	Accuracy	72.85%	86.64%	97.10%	94.32%	97.74%
	TPVD	2.6466	2.4850	1.7926	2.2032	–
CIFAR10	Accuracy	17.95%	21.26%	48.84%	31.98%	49.05%
	TPVD	1.7629	1.7164	1.4575	1.4718	–
SVHN	Accuracy	19.31%	20.57%	84.43%	25%	84.12%
	TPVD	2.2126	1.9679	1.5095	1.6518	–

Table 8 – Normality test for accuracy.

Factor	Level	Statistics	df	Sig.
Dataset	MNIST	.852	32	.000
	CIFAR10	.960	32	**.268**
	SVHN	.929	32	.036
Model	CNN	. 767	48	.000
	LSTM	. 834	48	.000
Optimizer	SGD	.798	48	.000
	Adam	.806	48	.000
Noise	0.1	.807	48	.000
	0.5	.791	48	.000
Clipping	0.8	.818	48	.000
	1.4	.792	48	.000
Sequence	AC	.772	48	.000
	CA	.765	48	.000

Table 9 – Normality test for convergence.

Factor	Level	Statistics	df	Sig.
Dataset	MNIST	.890	32	.003
	CIFAR10	.963	32	**.334**
	SVHN	.963	32	**.335**
Model	CNN	. 781	48	.000
	LSTM	. 834	48	.000
Optimizer	SGD	.816	48	.000
	Adam	.786	48	.000
Noise	0.1	.784	48	.000
	0.5	.808	48	.000
Clipping	0.8	.818	48	.000
	1.4	.800	48	.000
Sequence	AC	.769	48	.000
	CA	.797	48	.000

can conclude that the variable *convergence* is not normally distributed except for the situation when the *dataset* is CIFAR10 and SVHN.

6.2. *Significance test*

After the normality test, we study the effect of different factors on the results of *accuracy* and *convergence*, so we carry out the Mann-Whitney U test and the Kruskal-Wallis H test. Both tests can determine whether two or more groups come from the same distribution under the assumption that the shapes of the underlying distributions are the same. We only report the results of the Mann-Whitney U Test here, as the results of Kruskal-Wallis H test led to the same conclusion.

The Mann-Whitney U test tests a null hypothesis that two samples come from the same population. If the p-value is less than 0.05, then the null hypothesis is rejected.

6.2.1. *Accuracy*

The test results are shown in Table 10. As can be seen, the p-value of different factors are all greater than 0.05 except for the factor *sequence*; thus, we can conclude that there is a significant difference between the *accuracy* for the AC sequence compared with the CA sequence.

6.2.2. *Convergence*

The test results are shown in Table 11. When we examine the factors of *model*, *optimizer*, *clipping* and *sequence*, the p-values are greater than 0.05, and when we examine the factor *noise*, the p-value is less than 0.05. Thus, we can conclude there is a significant difference between the *convergence* of the groups of different noise.

Table 10 – The Mann-Whitney U test for accuracy.

Factor	Level	Mann-Whitney U	Wilcoxon W	Z-value	P-value
Model	CNN LSTM	970	2146	−1.334	0.182
Optimizer	SGD Adam	1247	2423	.696	0.486
Noise	0.1 0.5	935	2111	−1.590	0.112
Clipping	0.8 1.4	1076.5	2252.5	−0.553	0.580
Sequence	AC CA	678	1854	−3.474	**0.001**

Table 11 – The Mann-Whitney U test for convergence.

Factor	Level	Mann-Whitney U	Wilcoxon W	Z-value	P-value
Model	CNN LSTM	1160	2336	0.059	0.953
Optimizer	SGD Adam	915.50	2091.50	−1.733	0.083
Noise	0.1 0.5	1497	2673	2.528	**0.011**
Clipping	0.8 1.4	1189.5	2365.5	0.275	0.580
Sequence	AC CA	678	1854	−3.474	0.783

Table 12 – The accuracy and TPVD with different clipping size C.

Model	Performance	Sequence	C = 0.001	C = 0.04	C = 1.4	C = 10	C = 100	Normal
CNN	Accuracy	AC	98.08%	97.10%	94.32%	88.38%	30.47%	97.74%
		CA	84.94%	85.29%	85.95%	88.04%	71.82%	-
	TPVD	AC	1.5440	1.7926	2.2032	2.3400	2.3550	–
		CA	2.3466	2.3546	2.3716	2.7681	2.1589	–
LSTM	Accuracy	AC	69.53%	89.06%	81.25%	48.43%	21.09	99.21%
		CA	62.5%	57.81%	63.28%	41.40%	14.0%	-
	TPVD	AC	0.9898	1.0212	1.1512	1.1596	1.1520	–
		CA	0.8761	1.1927	1.1347	1.0972	1.1288	–

7. Further improvements on privacy protection

According to the experimental results, although our proposed method does improve the convergence effect and the accuracy of the DL model, its parameters perturbation effects are not as good as clipping before adding noise. In this section, we further improve our approach to make it have a better perturbation effect while still maintaining high utility.

The biggest difference between AC and CA is that AC can limit the stochasticity of noised gradients by clipping, whereas CA can increase the stochasticity of gradients by adding noise. Thus, based on the idea that *bigger gradient norm clipping bound means bigger stochasticity of gradients*, we make a hypothesis: "*The TPVD values increases as the clipping size increases while still maintaining an acceptable accuracy*."

In order to verify this hypothesis, we gradually increase the clipping size under the same parameter settings where the *noise* is 0.5, the *optimizer* is Adam, and the *dataset* is MNIST. The experimental results are shown in Table 12. We can see that the TPVD value increases as the clipping size increases no matter under AC or CA. Under the same clipping threshold, the accuracy of AC is higher than that of CA. Compared with the CA sequence, with the increase of clipping size, we get a gradually increased TPVD value and finally achieve roughly the same perturbation effects while still maintain a high accuracy. To confirm this finding, we performed similar experiments under other parameter settings and still obtained the same results. We should also note that as C increases and after exceeding a certain threshold, the accuracy of the training model will also decrease. We can see that after the threshold exceeds 10, whether it is AC or CA, the accuracy decreases rapidly, so we cannot blindly pursue privacy protection and ignore utility.

8. Conclusions and discussions

DP has shown promise in protecting or preserving data and model privacy in ML applications. Applying DP to DL is a growing trend. Different methods have been proposed to address the privacy protection issues, focusing on objective function

or output or gradient. The basic blueprint for designing a differentially private additive-noise mechanism usually consists of several steps: identifying the functions that need to be perturbed; choosing parameters of additive noise; and analysing privacy. Although there have been many studies aiming to solve the privacy protection problem in DL, there have been no studies devoted to examining the effect of different factors.

Adding noise to gradient computation usually involves bounding gradients first and then adding noise. We propose a contrary method of adding noise first and then bounding gradients based on our theoretical analyses and empirical observations. We compare the performance between those two methods. To further explore the applicability of the proposed ideas, we also test several other factors which may impact a model's performance. To the best of our knowledge, this study is the first to attempt to explore and quantitatively identify their relative impacts.

Extensive experimental evaluations and statistical analyses validated the effectiveness of our proposed modification. Firstly, the detailed experimental results showed that changing the sequence of adding noise and clipping can really achieve higher accuracy and faster convergence than the original, even under different parameter settings. In principle, our proposed method can be applied to various optimization algorithms. Secondly, comparison with a state-of-the-art technique showed that our proposed method can achieve better performance by setting a reasonable clipping threshold. Thirdly, through extensive statistical analyses, the results indicate that: (1) the factors of *dataset* and *sequence* are significant to *accuracy*; (2) the factors of *dataset* and *noise* are significant to *convergence* and the results of our statistical analyses also indicate that changing sequence can achieve satisfactory accuracy; (3) the TPVD metric proposed in this paper as a privacy protection metric for DL models can better reflect the perturbation effects on learned weights; (4) under the same parameter settings, carefully increasing the clipping size can achieve roughly the same perturbation effect on learned weights while still maintaining an acceptable accuracy. This finding verifies that adding bounded noise can improve learning for DL models

Several opportunities exist for further research. For example, like other attempts, we still need to provide a rigorous theoretical proof as to why sequence plays such an important role on a model's performance. In the future, we also intend to explore the use of adaptive selection of parameters in our method.

Credit author statement

Ying LIN, the lead author, is responsible for conceptualization, funding acquisition, project administration, resources, supervision and writing - original draft of the paper.

LING-YAN BAO is responsible for data curation, formal analysis, investigation and implementation of the privacy-preserving algorithm.

ZE-MINGHUI LI is responsible for investigation and visualization of experimental data.

SHU-ZHENG SI is responsible for investigation, methodology, and software implementation of codes for privacy metrics and conducting the experiment.

Chao-Hsien Chu, the corresponding author, is responsible for conceptualization, structuring the paper, validation, writing - review & editing, and ensuring the overall accuracy and quality of the paper.

Declaration of Competing Interest

None.

Acknowledgments

This work is partially financed to the first author by (1) the Data Driven Software Engineering Research Innovation Team of Yunnan Province (No.2017HC012); (2) the project of Key Laboratory for Software Engineering of Yunnan Province (No. 2017SE102). Our deepest gratitude goes to the two anonymous reviewers and editor for their careful review and comments and thoughtful suggestions that have helped improve this paper substantially.

REFERENCES

Abadi M, et al. Deep learning with differential privacy. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016.

Acs G, et al. Differentially Private Mixture of Generative Neural Networks. IEEE Trans Knowl Data Eng 2019;31(6):1109–21.

Badawi, A.A., et al., The AlexNet moment for homomorphic encryption: HCNN, the first homomorphic CNN on encrypted data with GPUs. arXiv preprint arXiv:1811.00778, 2018.

Baldominos A, Saez Y, Isasi P. A survey of handwritten character recognition with mnist and emnist. Applied Sciences 2019;9(15):3169.

Bassily R, Smith A, Thakurta A. In: 2014 IEEE 55th Annual Symposium on Foundations of Computer Science. Private empirical risk minimization: efficient algorithms and tight error bounds. IEEE; 2014.

Bengio Y, Boulanger-Lewandowski N, Pascanu R. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. Advances in optimizing recurrent networks. IEEE; 2013.

Blundell, C., et al., Weight uncertainty in neural networks. arXiv preprint arXiv:1505.05424, 2015.

Bottou L. Online learning and stochastic approximations. On-line learning in neural networks 1998;17(9):142.

Carmon Y, et al. Lower bounds for finding stationary points i. Math Program 2017:1–50.

Chaudhuri K, Monteleoni C, Sarwate AD. Differentially Private Empirical Risk Minimization. J. Mach. Learning Res. 2011;12:1069–109.

Cor I. IBM SPSS Statistics for Windows, Version 24.0. Armonk, NY, USA: IBM Corp.; 2016.

Das T. Machine Learning algorithms for Image Classification of hand digits and face recognition dataset. Mach Learn 2017;4(12):640–9.

Dong, J., A. Roth, and W. Su, Gaussian Differential Privacy. arXiv: Learning, 2019.

Du W, Atallah MJ. Secure multi-party computation problems and their applications: a review and open problems. Proceedings of the 2001 workshop on New security paradigms. ACM, 2001.
Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 2011;12(Jul):2121–59.
Dwork C, Roth A. The algorithmic foundations of differential privacy, 9; 2014. p. 211–407.
Dwork C, et al. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Our data, ourselves: privacy via distributed noise generation. Springer; 2006a.
Dwork C, et al. In: Theory of cryptography conference. Calibrating noise to sensitivity in private data analysis. Springer; 2006b.
Fredrikson M, et al. In: USENIX Security Symposium. Privacy in Pharmacogenetics: an End-to-End Case Study of Personalized Warfarin Dosing; 2014.
Fredrikson M, Jha S, Ristenpart T. Model inversion attacks that exploit confidence information and basic countermeasures. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 2015.
Gilad-Bachrach R, et al. In: International Conference on Machine Learning. Cryptonets: applying neural networks to encrypted data with high throughput and accuracy; 2016.
Graves A. In: Advances in neural information processing systems. Practical variational inference for neural networks; 2011.
Hayes J, et al. LOGAN: membership inference attacks against generative models. Proceedings on Privacy Enhancing Technologies 2019;2019(1):133–52.
Hesamifard E, et al. Privacy-preserving machine learning as a service. Proceedings on Privacy Enhancing Technologies 2018;2018(3):123–42.
Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80.
Kandi H, Mishra D, Gorthi SRS. Exploring the learning capabilities of convolutional neural networks for robust image watermarking. Computers & Security 2017;65:247–68.
Kingma, D.P. and J. Ba, Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
Krizhevsky, A., V. Nair, and G. Hinton, The cifar-10 dataset. online: http://www.cs.toronto.edu/kriz/cifar.html, 2014.55.
LeCun, Y., Y. Bengio, and G. Hinton, Deep learning. nature, 2015.521(7553): p. 436.
LeCun, Y., The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998.
Lee J. In: 2017 International Conference on New Trends in Computing Sciences (ICTCS). Differentially Private Variance Reduced Stochastic Gradient Descent; 2017.
Liu Q, et al. A survey on security threats and defensive techniques of machine learning: a data driven view. IEEE access 2018;6:12103–17.
Mcmahan, B., et al., A General Approach to Adding Differential Privacy to Iterative Training Procedures. arXiv: Learning, 2018.
McSherry F, Talwar K. In: Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on. Mechanism design via differential privacy. IEEE; 2007.
Neelakantan, A., et al., Adding gradient noise improves learning for very deep networks. arXiv preprint arXiv:1511.06807, 2015.
Netzer, Y., et al., Reading digits in natural images with unsupervised feature learning. 2011.
Papernot, N., et al., Towards the science of security and privacy in machine learning. arXiv preprint arXiv:1611.03814, 2016.
Papernot, N., et al., Semi-supervised knowledge transfer for deep learning from private training data. arXiv preprint arXiv:1610.05755, 2016.
Phan N, et al. In: Thirtieth AAAI Conference on Artificial Intelligence. Differential privacy preservation for deep auto-encoders: an application of human behavior prediction; 2016.
Phan N, Wu X, Dou D. Preserving differential privacy in convolutional deep belief networks. Mach Learn 2017;106(9–10):1681–704.
Pichapati, V., et al., AdaCliP: adaptive clipping for private SGD. arXiv preprint arXiv:1908.07643, 2019.
Rivest RL, Adleman L, Dertouzos ML. On data banks and privacy homomorphisms. Foundations of secure computation 1978;4(11):169–80.
Senavirathne N, Torra V. Integrally private model selection for decision trees. Computers & Security 2019;83:167–81.
Shokri R, Shmatikov V. Privacy-preserving deep learning. Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. ACM, 2015.
Shokri R, et al. In: 2017 IEEE Symposium on Security and Privacy (SP). Membership inference attacks against machine learning models. IEEE; 2017.
Song S, Chaudhuri K, Sarwate AD. In: 2013 IEEE Global Conference on Signal and Information Processing. Stochastic gradient descent with differentially private updates. IEEE; 2013.
Thakkar, O., G. Andrew, and H.B. McMahan, Differentially private learning with adaptive clipping. arXiv preprint arXiv:1905.03871, 2019.
Tramèr F, et al. In: 25th {USENIX} Security Symposium ({USENIX} Security 16). Stealing machine learning models via prediction apis; 2016.
Wang B, Gong NZ. In: 2018 IEEE Symposium on Security and Privacy (SP). Stealing hyperparameters in machine learning. IEEE; 2018.
Wang J, et al. In: IKE. Selective Data Distortion via Structural Partition and SSVD for Privacy Preservation. Citeseer; 2006.
Xie, L., et al., Differentially Private Generative Adversarial Network. arXiv: Learning, 2018.
Yu L, et al. In: IEEE symposium on security and privacy. Differentially Private Model Publishing for Deep Learning; 2019.
Zeiler, M.D., ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701, 2012.
Zhang J, et al. Functional mechanism: regression analysis under differential privacy. Proceedings of the VLDB Endowment 2012;5(11):1364–75.
Zhao J, Chen Y, Zhang W. Differential Privacy Preservation in Deep Learning: challenges, Opportunities and Solutions. IEEE Access 2019;7:48901–11.

Ying Lin an associate professor in Yunnan University, China. She received the B.S. degree in Applied Mathematics from Shanghai Jiaotong University, China, in 1996, and the M.S. degree in Computer Software and Theory from Yunnan University, China in 2004. She received the Ph.D. degree in System Analysis and Integration from Yunnan University in 2013. Her research interests include network security, privacy protection, machine learning and artificial intelligent for security.

Ling-Yan Bao is an undergraduate student of department of Information Security in Yunnan University, China. His-research interests include machine Learning and privacy preserving deep learning. He has won the 5th "Internet +" provincial golden medal during his studying period.

Ze-Minghui Li is an undergraduate student of department of Information Security in Yunnan University, China. His-research interests include privacy preserving deep learning and network security.

Shu-Zheng Si is an undergraduate student of department of Information Security in Yunnan University, China. He has got Na-

tional Scholarship, Yunnan Provincial Government Scholarship, and Yunnan Provincial-Level Merit Student during his studying period. His-research interests include machine Learning, adversarial examples and differential privacy.

Chao-Hsien Chu is a professor of Information Sciences and Technology at the Pennsylvania State University, USA. His-research interests include (1) Cybersecurity and Privacy Assurance; (2) Internet of Things; and (3) Big Data analytics. He has published more than 190 papers; many of them are in top-ranking journals such as IEEE Transactions on Information Forensics & Security, IEEE Transactions on Dependable and Secure Computing, Computers & Security, INFORMS Journal on Computing; or proceeding including USENIX Security, ACM Conference on CCS, IEEE Conference on Computer Communications (INFOCOM) etc. Six of his papers have received the Best Paper Award from major societies, including IEEE, Decision Sciences Institute, European Operational Research Society, etc.