Received: 3 July 2022 | Accepted: 6 August 2022

DOI: 10.1002/int.23001

RESEARCH ARTICLE

WILEY

Understanding adaptive gradient clipping in DP-SGD, empirically

Guanbiao Lin[1] | **Hongyang Yan**[1] | **Guang Kou**[2] | **Teng Huang**[1] | **Shiyu Peng**[1] | **Yingying Zhang**[1] | **Changyu Dong**[1]

[1]Institute of Artificial Intelligence and Blockchain, Guangzhou University, Guangdong, China

[2]Artificial Intelligence Research Center, Defense Innovation Institute, Beijing, China

Correspondence
Hongyang Yan, Institute of Artificial Intelligence and Blockchain, Guangzhou University, 510006 Guangdong, China.
Email: hyang_yan@gzhu.edu.cn and hyang.yan@foxmail.com

Funding information
National Natural Science Foundation of China, Grant/Award Numbers: 62102107, 62072132, 62002074

Abstract

Differentially Private Stochastic Gradient Descent (DP-SGD) is a prime method for training machine learning models with rigorous privacy guarantees. Since its birth, DP-SGD has gained popularity and has been widely adopted in both academic and industrial research. One well-known challenge when using DP-SGD is how to improve utility while maintaining privacy. To this end, recently we have seen several proposals that clip the gradients with adaptive thresholds rather than a fixed one. Although each proposal comes with some theoretical justification, the theories often rely on strong assumptions and are not compatible with each other. It is hard to know whether they are good in practice and how good they are. In this paper, we investigate adaptive clipping in DP-SGD from an empirical perspective. With extensive experiments, we were able to gain some fresh insights and proposed two new adaptive clipping strategies based on them. We cross-compared the existing methods and our new strategies experimentally. Results showed that our strategies did provide a substantial improvement in model accuracy, and outperformed the state-of-the-art adaptive clipping methods consistently.

KEYWORDS
adaptive gradient clipping, differentially private stochastic gradient descent, machine learning, privacy protection

1 | INTRODUCTION

Neural networks have been widely applied in various domains, such as image classification,[1,2] natural language processing,[3–5] speech recognition,[6–9] and recommendation systems.[10–12] However, recent researches[13–16] have shown that neural network models may leak or memorize their training data, which raises privacy concerns when training with sensitive data. In addition, the leakage of sensitive information such as model gradients caused by malicious adversarial attacks[17–19] on neural networks is also worrying. Differential privacy[20] has become the de facto standard for protecting an individual's privacy in machine learning, and it has been widely used in various scenarios.[21,22] When the neural network model satisfies the differential privacy in the training process, it can ensure that the model will not overdisclose any sensitive information. Following the seminal paper,[23] there has been significant work on the Differentially Private Stochastic Gradient Descent (DP-SGD), which can be used to train privacy-preserving neural networks.

DP-SGD can provide effective protection for sensitive data, but at a cost. The accuracy of differentially private machine learning models is always lower than that of nonprivate learning. The loss of utility seems inevitable, given the construction of DP-SGD. At a high level, DP-SGD extends SGD with gradient clipping and Gaussian perturbation. When given a minbatch, DP-SGD first calculates the per-sample gradient, $g(x_i)$, and then clips the ℓ_2-norm of the $g(x_i)$ through formula $g(x_i)/\max(1, \|g(x_i)\|_2/C)$ (Line 6 in Abadi et al.,[23] Algorithm 1). We can see that the norm will be large (proportional to the number of model parameters), especially for some complex neural networks with multiple layers, clipping will bring a great influence. The "smaller" clipped gradients are easily overwhelmed by the added noise, which leads to the decrease of model utility.

In this paper, we report interesting insights gained empirically, regarding adaptive gradient clipping in DP-SGD. We started from DP-SGD with a constant clipping threshold, to verify the folklore (but has never been proven theoretically or empirically) that a constant clipping threshold is not a good choice. Then we developed a greedy algorithm that searches for the optimal clipping thresholds in each training epoch. The clipping thresholds found by the greedy algorithm are compared with those found/calculated by existing adaptive clipping methods. The comparison highlights some limitations of existing methods, and also provides intuitions for better designs. We then proposed two simple strategies, utilizing the greedy algorithm and transferability, to adaptively adjust clipping thresholds. We conducted extensive experiments, and the results show that our strategies achieved the best performance consistently, despite their simplicity. The main contributions of this paper are summarized as follows:

- We report interesting insights gained empirically, regarding adaptive gradient clipping in DP-SGD. And verify the folklore that a constant gradient clipping threshold is not a good choice, which will affect the model convergence and reduce the model utility.
- We developed a greedy algorithm to find the optimal clipping threshold for each epoch, and compared the gradient clipping threshold found by it with the thresholds found by other existing adaptive methods, pointing out some limitations of these methods.
- We proposed two methods for adaptively selecting gradient clipping thresholds for DP-SGD: Transfer and Decay. The results show that our strategies achieved the best performance consistently.
- We conducted extensive experiments on a variety of data sets, in particular we conduct experiments on the Imagenet data set, which to our knowledge is the first to do so, previous

similar work has only been done on small data sets. We also experiment and compare various other adaptive clipping methods, and draw some conclusions to help understand adaptive gradient clipping in DP-SGD.

The rest of this paper is structured as follows. In Section 2, we summarize the existing methods for improving the utility of DP-SGD models. In Section 3, we give some definitions and overview. In Section 4, we analyze some problems in DP-SGD clipping through experiments and introduce the framework of our propose methods. In Section 5, we describe the experimental setup and results. Finally, we conclude in Section 6.

2 | RELATED WORK

Abadi et al.[23] first proposed DP-SGD, which has become a de facto training method for differentially private machine learning models. However, as pointed out by many papers,[24–28] it is difficult to balance the trade-off between privacy and utility when using DP-SGD. Recently, Papernot et al.[29] showed that using bounded activation functions (the tempered sigmoids) can lead to better utility than using unbounded ones in DP-SGD training. They use this method to achieve the best performance among the fixed clipping threshold schemes. This change is on the networks rather than the DP-SGD algorithm. There are two other types of ways to improve the utility of DP-SGD, as described below.

2.1 | Adaptive clipping in DP-SGD

A major algorithmic approach to improving utility when using DP-SGD is by setting the clipping threshold adaptively during the training process. The earliest method was proposed by van der Veen et al.,[30] which sets the threshold based on the differentially private mean ℓ_2-norm of the previous batch. However, it has been shown that this method does not perform well, and the additional privacy budget for DP-mean calculation could have a negative impact on utility. Another early work in this field is Yu et al.[31] They use a linear decay function $C_t = \frac{C}{\min(2, 1+\frac{t}{T})}$ to set the threshold C_t at the tth training round (T is the total round). The decay function is independent of the training data, thus does not consume any additional privacy budget. AdaClip[32] is a per-coordinate adaptive clipping method. It does not change the ℓ_2 clipping threshold, but adaptively scales the gradient coordinate-wisely. This effectively changes the per coordinate threshold. More recently, Andrew et al.[33] proposed a new method for federated learning, which can also be applied to nonfederated training. The method sets threshold by tracking a specified quantile of the gradient norm distribution. The quantile tracking is differentially private, but only assumes a small privacy budget. Another recent work is Du et al.[34] They use a near-linear decay function $C_t = (\rho_c)^{-\frac{t}{T}}$ to set the threshold. In addition, they also adjust the noise scaler adaptively.

2.2 | Reduce gradient dimension

Another approach for improving the utility when using DP-SGD is by reducing the dimensionality of gradients. The observation is that the intensity of the added noise scales

linearly with the gradients dimension. Hence the gradients will be overwhelmed by the added noise if the dimensionality is large. Gondara et al.[35] reduce the dimensionality of the gradient by looking for a subnetwork, based on the lottery ticket hypothesis,[36] and then use DP-SGD to train the subnetwork. Zhou et al.[37] proposed Projected DP-SGD that performs noise reduction by projecting the noisy gradients to a low-dimensional subspace, which is given by the top gradient eigenspace on a small public data set. Yu et al.[38] extend the work of Zhou et al.[37] They proposed a Gradient Embedding Perturbation algorithm to reduce the gradient dimension. Tramer and Boneh[39] used a public function (ScatterNet) for feature extraction, so that the model trained with DP-SGD using the features can be simplified and much smaller. These approaches are quite different from adaptive clipping, so we do not compare with them in the paper. We should mention that the adaptive threshold approach can actually be applied in training the dimension-reduced models produced by some of those methods of Gondara et al.[35] and Tramer and Boneh.[39]

3 | PRELIMINARIES

3.1 | Differential privacy

Differential privacy[20] provides a formal privacy definition, with the intuition that a randomized algorithm behaves similarly on "similar" input data sets.

Definition 1. A randomized mechanism $\mathcal{M}: \mathcal{X}^n \to \mathbb{R}^d$ satisfies (ϵ, δ)-differential privacy if for any two data sets $\mathcal{D}, \mathcal{D}' \in \mathcal{X}^n$ differing by a single element and for any set of possible output $\mathcal{O} \subseteq \text{Range}(\mathcal{M})$:

$$\Pr[\mathcal{M}(\mathcal{D}) \in \mathcal{O}] \leq e^{\epsilon}\Pr[\mathcal{M}(\mathcal{D}') \in \mathcal{O}] + \delta. \tag{1}$$

In the above, ϵ is the privacy budget, a parameter that controls the trade-off between the utility and privacy protection of the differential privacy algorithm. Smaller ϵ usually means better privacy but also worse utility. The additive term δ, is a small probability that the output distributions differ more than the ϵ bound, and the value of δ is typically chosen to be smaller than $1/|\mathcal{D}|$.

Definition 2. Given a query $f: \mathcal{X}^n \to \mathbb{R}^d$, the global sensitivity Δ_f is defined as

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

The global sensitivity measures the maximum possible change in $f(\mathcal{D})$ when one record in the data set changes.

Theorem 1. *For a query function $f: \mathcal{X}^n \to \mathbb{R}^d$ with sensitivity Δ_f, the Gaussian Mechanism that adds noise generated from the Gaussian distribution $\mathcal{N}(0, \sigma^2\mathbb{I})$ to the output of f satisfies (ϵ, δ)-differential privacy, where $\epsilon, \delta \in (0, 1)$ and $\sigma \geq \frac{\sqrt{2\ln(1.25/\delta)}\Delta_f}{\epsilon}$.*

3.2 | Gradient clipping in DP-SGD

DP-SGD[23] is a useful optimization technique for learning a model f under differential privacy constraints, the detailed procedure can be found in Appendix A. DP-SGD randomly draws a mini-batch of training examples, computes the gradients, clip them, then applies the Gaussian mechanism to the gradients. To bound the ℓ_2-sensitivity, a fixed-threshold $\mathcal{C}$ is used to clip the per-example gradient. More formally, given the gradient g on an example and a threshold $\mathcal{C}$, gradient clipping does the following:

$$\text{clip}(g) = g \cdot \min\left(1, \frac{\mathcal{C}}{\|g\|_2}\right). \tag{3}$$

On the basis of the clipped gradients, DP-SGD crafts a randomized gradient $\tilde{g}$ through computing the average over the clipped gradients and adding noise whose scale is defined by $\mathcal{C}$ and σ_ε, where σ_ε is noise scaler to satisfy (ϵ, δ)-DP.

$$\tilde{g} = \frac{1}{B}\left(\sum_{i\in[B]} \text{clip}(g_i) + \mathcal{N}(0, (\sigma_\varepsilon \mathcal{C})^2 \mathcal{I})\right). \tag{4}$$

However, the clipping operation can create a substantial bias in the update direction,[40] which will prevent the model from converging in the worst case. Therefore, the selection of clipping threshold is particularly important, as it determines the scale of gradient clipping and Gaussian noise addition. Song et al.[41] showed that if the clipping norm is set smaller than the optimal, even by a constant factor, the excess empirical risk for convex empirical risk minimization can increase from $\mathcal{O}(1/n)$ to $\Omega(1)$ in DP-SGD, where n is the number of data samples.

3.3 | Greedy algorithm

The greedy algorithm is a mathematical process that looks for simple, easy-to-implement solutions to complex, multistep problems by deciding which next step will provide the most obvious benefit. It builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. Therefore, the problems where choosing locally optimal also leads to global solution are the best fit for Greedy. There are many works[42–44] that use greedy algorithm to optimize the problem. In this paper, we also develop a greedy algorithm to iteratively find the optimal clipping threshold under each epoch in DP-SGD training. Our problem is expressed as

$$\text{Greedy}(f_{t-1}, C) \rightarrow \text{optimal}\, C_t. \tag{5}$$

In the tth epoch of training, we iterate the gradient clipping threshold of C until model f_{t-1} achieves the best performance on a certain C. Then this gradient clipping threshold C is optimal under the current epoch.

4 | ADAPTIVE CLIPPING THRESHOLD

4.1 | DP-SGD with constant clipping threshold

In the original paper and most implementations, the DP-SGD algorithm uses a fixed clipping threshold throughout the training process. Although intuitively an adaptive threshold is better than a fixed one, there is no convincing analysis or evidence so far to support that. In a nonprivate setting, the gradient norm almost always decreases in the number of training epochs. However, in DP-SGD the trend of gradient norm is less clear because of the clipping operation and the noises added. Hence it is hard to know what is the impact of using a fixed clipping threshold on training, especially in later epochs. To gain the intuition, we measured the ℓ_2 norm of the per-sample (CIFAR-10) gradient *before clipping* in each epoch, and took the median to plot Figure 1A. In the experiment, we used fixed thresholds 0.6, 1.0, and 1.5. We also measured the training loss accordingly, and show the result in Figure 1B. There are a few interesting observations: (1) there is a ramp-up in the gradient norm in early epochs; (2) the gradient norm then decreases after the ramp-up period, at a larger rate when using a smaller clipping threshold. (3) The training loss decreases sharply during the ramp-up period; (4) after the ramp-up period, the training loss decreases at a mild rate at first, then goes up eventually; (5) a larger threshold allows the training loss to decrease faster at early epochs, and a smaller clipping threshold allows the training loss to reach a lower point.

For the phenomenon that the gradient norm would gradually decrease in DP-SGD training observed in the above experiment, we further provide a theoretical analysis, as shown below:

Theorem 2. *Suppose model* $f : \mathbb{R}^m \mapsto \mathbb{R}$ *is a twice differentiable strictly convex function. Consider a function* $g = f(x_{k-1}) + \nabla f(x_{k-1})(x - x_{k-1}) + \frac{1}{2}(x - x_{k-1})^T H (x - x_{k-1})$ *that is the second-order Taylor approximation of* f *around* x_{k-1}. *Let* $\Omega = H^{\mathrm{T}}H$, *for* $x_{k-1}, x_k \in \mathbb{R}^m$, *if* $|\nabla f(x_{k-1})|_2^2 > 0$, *and* $x_k = x_{k-1} - \alpha \cdot (\nabla f(x_{k-1}) + N(\delta, n))$, *then there exists an* $\alpha > 0$ *such that with probability* $1 - \exp\left(-\left(\frac{\sqrt{\mathrm{tr}(\Omega^2) - 2\,|\,\Omega\,|_2\,(\mathrm{tr}(\Omega) - \lambda_{\min}\,|\,\nabla f(x_{k-1})\,|_2\,/\delta)^2} - \sqrt{\mathrm{tr}(\Omega^2)}}{|\Omega|_2}\right)^2\right)$, *the inequality* $|\nabla g(x_k)|_2^2 \leq |\nabla g(x_{k-1})|_2^2$ *holds.*

Proof. Since f is twice differentiable, we have

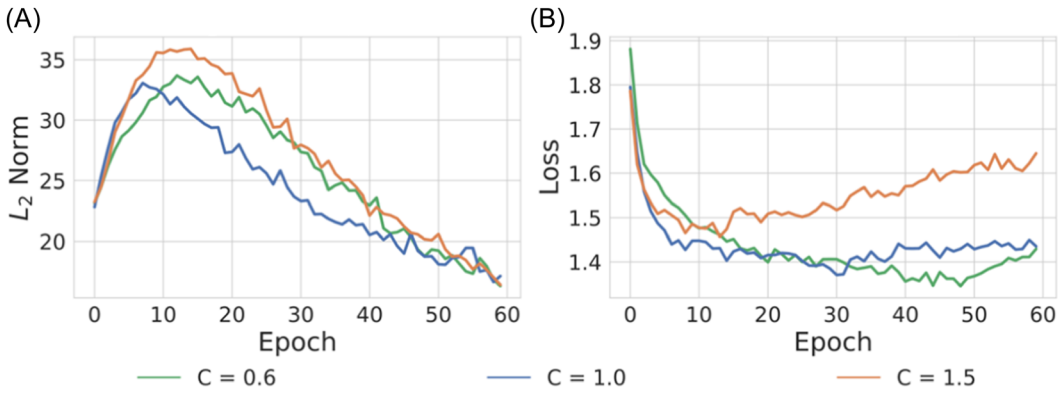

FIGURE 1 Trend of (A) L_2 norm of the gradient and (B) training loss, fixed $C = 0.5$, 1.0, and 1.5, and $\epsilon = 6.0$ (data set CIFAR-10) [Color figure can be viewed at wileyonlinelibrary.com]

$$f(x) = f(x_{k-1}) + \nabla f(x_k - 1)(x - x_{k-1}) + \frac{1}{2}(x - x_{k-1})^{\top} \cdot H(x - x_{k-1}) + o\left(|x - x_{k-1}|_2^2\right) \tag{6}$$

holds for sufficiently small $|x - x_{k-1}|_2$. Because f is strictly convex, then the Hessian matrix H is symmetric and positive definite.

With noisy gradient descent

$$x_k = x_{k-1} - \alpha \cdot (\nabla f(x_{k-1}) + N(d, m), \tag{7}$$

then the gradient at x_k is

$$\begin{aligned} \nabla f(x_k) &= \nabla f(x_{k-1}) - \alpha H(\nabla f(x_{k-1}) + \mathcal{N}(\delta, m)) \\ &= (I_m - \alpha H)\nabla f(x_{k-1}) - \alpha H \cdot \mathcal{N}(\delta, m). \end{aligned} \tag{8}$$

Hence we have $|\nabla f(x_k)|_2 \le |I_m - \alpha H|_2 \cdot |\nabla f(x_{k-1})|_2 + |\alpha H \cdot \mathcal{N}(\delta, m)|_2$. Let $\lambda_{\min}, \lambda_{\max} > 0$ denote the minimum and maximum eigenvalue of H, respectively, when $\alpha \le \frac{1}{\lambda_{\max}}$, the l_2 matrix norm $|I_m - \alpha H|_2$ is equal to $1 - \alpha\lambda_{\min}$. It implies that

$$|\nabla f(x_k)|_2 - |\nabla((x_k - 1)|_2 \geqslant \alpha(\lambda_{\min}|\nabla f(x_{k-1})|_2 - |H \cdot N(\delta, m)|_2). \tag{9}$$

Now consider the (probabilistic) bound of $|H \cdot \mathcal{N}(\delta, m)|_2$, which is equivalent to $\delta \cdot |H \cdot \mathcal{N}(1.0, m)|_2$. Since $\Omega = H^{\mathrm{T}}H$, according to the tail bounds on ℓ_2-norm of multivariate Gaussian variables,[1] we actually have

$$\mathbb{P}\left[|H \cdot \mathcal{N}(1.0, m)|_2^2 \ge tr(\Omega) + 2\sqrt{tr(\Omega^2)t} + 2|\Omega|_2 t\right] \le \exp(-t). \tag{10}$$

Therefore, when $\lambda_{\min}|\nabla f(x_{k-1})|_2 \ge \mathrm{tr}(\Omega)$, the inequality $\lambda_{\min}|\nabla f(x_{k-1})|_2 - |H \cdot \mathcal{N}(\delta, m)|_2 \ge 0$ holds, with the probability at least $1 - \exp\left(-\left(\frac{\sqrt{\mathrm{tr}(\Omega^2) - 2 \mid \Omega \mid_2 (\mathrm{tr}(\Omega) - \lambda_{\min} \mid \nabla f(x_{k-1}) \mid_2 / \delta)^2} - \sqrt{\mathrm{tr}(\Omega^2)}}{|\Omega|_2}\right)^2\right)$.

Combining the results in the previous two paragraphs, we have the conclusion (when $\alpha \le \frac{1}{\lambda_{\max}}$ and $\lambda_{\min}|\nabla f(x_{k-1})|_2 \ge \mathrm{tr}(\Omega)$). □

As Theorem 2 implied, when the scaled gradients norm $\lambda_{\min}|\nabla f(x_{k-1})|_2$ is relatively large or the Gaussian deviation $\delta \to 0$, the norm of gradients $|\nabla g(x_k)|_2^2$ in the next step decreases with certain.

From the above, we can see a fixed clipping threshold is not ideal, the norm of the gradient is changing all the time during training. Heuristically, we can deduce the following: (1) at the early stage, a larger clipping threshold allows faster convergence since it allows more changes to be incorporated into the model updates, and (2) at the later stage, a smaller clipping threshold allows the model to converge to a better one since the noise magnitude is scaled based on the threshold, and a smaller threshold prevents overshadowing noise.

4.2 | Optimal clipping threshold

In Section 4.1, we have shown that a fixed threshold is not good, and the rough ideas of setting thresholds at different training stages. Existing adaptive clipping methods in the literature[31,33,34] are proposed based on similar motivations, but the ways they choose the clipping threshold and the threshold chosen in practice differ significantly. Since the adaptive threshold could affect the model utility greatly, a question arises naturally: How good are the thresholds chosen by those methods?

Due to the lack of theories, currently there is no theoretically sound framework for analyzing, evaluating, and comparing those methods. Hence, we do so empirically by designing a greedy algorithm to find the optimal clipping thresholds. The algorithm is shown in Algorithm 1. In each epoch t, to find the threshold that offers the best utility, we start from a very small initial threshold C, then increase it to search for the one that gives the best test accuracy. In each iteration of the while loop, we train the model with DP-SGD for 1 epoch (starting from θ_{t-1}). If the test accuracy is better than the current best, we store the current test accuracy, threshold, and model parameters, then increase C by d for the next iteration. Because of the stochastic nature of DP-SGD, we allow the search to continue even if increasing C causes a decrease in the test accuracy, as long as the decrease is no more than a predefined threshold S. The loop ends if the test accuracy decreases sharply. The algorithm then outputs the model parameter with the all-time-best test accuracy (to be used to find $\mathcal{C}_{t+1}$ and θ_{t+1}), and the corresponding best threshold $\mathcal{C}_t$.

We ran the greedy algorithm on four different data sets: MNIST, Fashion-MNIST, CIFAR-10, and Imagenette. We also ran the adaptive clipping methods proposed in Yu et al.,[31] Andrew et al.,[33] and Du et al.[34]* We recorded the optimal thresholds found by the greedy algorithm, and the thresholds found by the other methods. We show three sets of results in Figures 2–4 (the result of Imagenette data set can be found in Appendix E). In these figures, the orange line plots the optimal thresholds found by the greedy algorithm, and the green line is the trendline fitted using the discrete optimal threshold points. The thresholds chosen by Andrew et al.[33] are significantly larger than the other two, so it is plotted separately in the

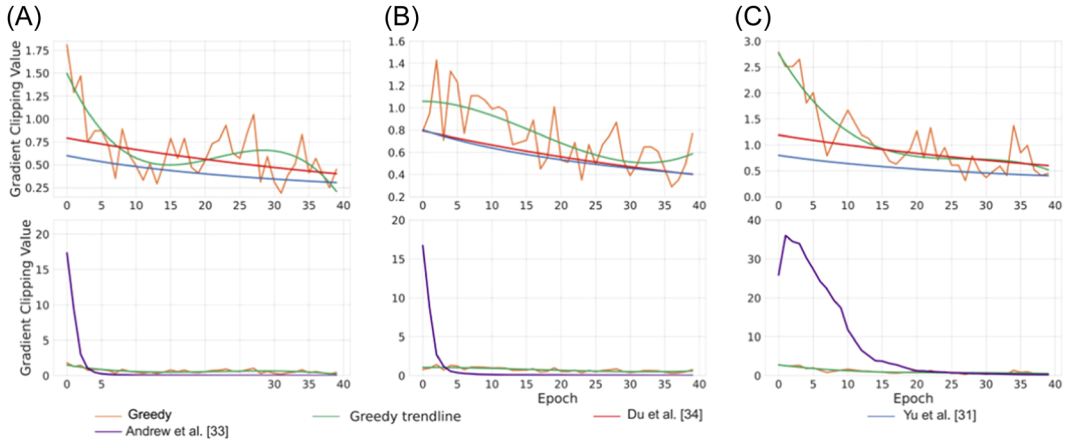

FIGURE 2 Comparison of gradient clipping thresholds with different adaptive methods for fixed privacy budget ϵ = (A) 1.2, (B) 2.93, and (C) 4.5 on MNIST data set [Color figure can be viewed at wileyonlinelibrary.com]

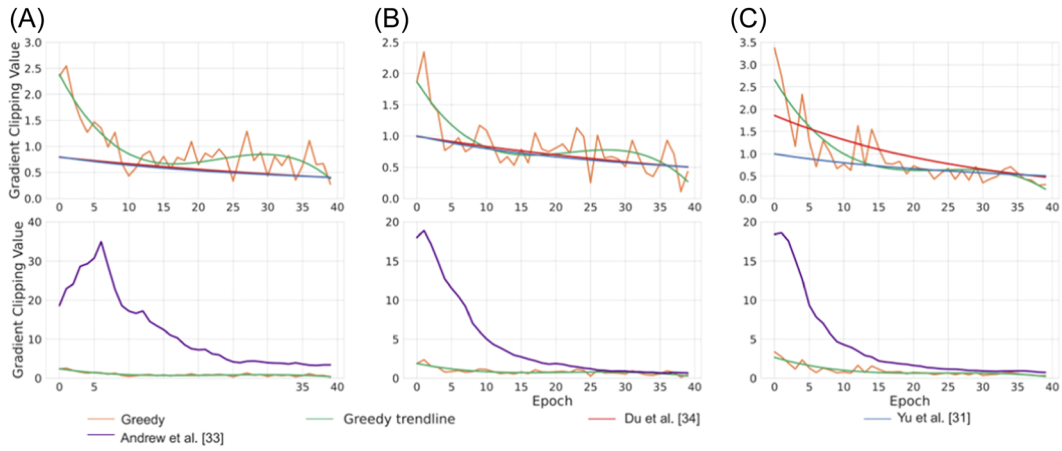

FIGURE 3 Comparison of gradient clipping thresholds with different adaptive methods for fixed privacy budget ϵ = (A) 1.5, (B) 2.7, and (C) 4.5 on Fashion-MNIST data set [Color figure can be viewed at wileyonlinelibrary.com]

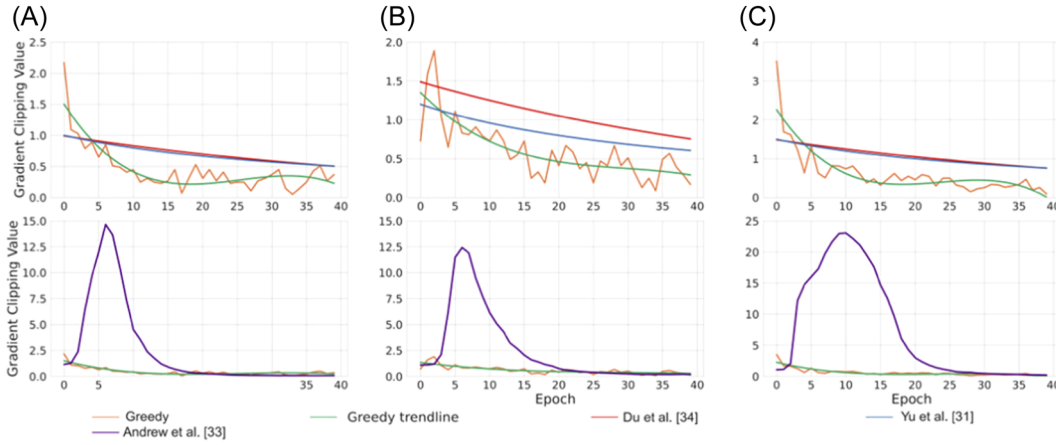

FIGURE 4 Comparison of gradient clipping thresholds with different adaptive methods for fixed privacy budget ϵ = (A) 3.0, (B) 6.78, and (C) 7.53 on CIFAR-10 data set [Color figure can be viewed at wileyonlinelibrary.com]

three subfigures in the second row of each figure. Note the thresholds chosen by all the previous three methods depend much on hyperparameters. The thresholds we report for each method in Figures 2 and 4 across different data sets are those that achieved the best test accuracy but with different hyperparameters.

As we can see in these figures, the magnitude of the optimal thresholds in general is decreasing, albeit oscillating (orange line). The methods in Yu et al.[31] and Du et al.[34] choose thresholds by a linear or a near-linear decay function with an initial value. The thresholds chosen by these two methods with the best performing hyperparameters are very close in value, and the decreasing trend is similar to the optimal ones found by the greedy algorithm. Also we can see that the trendline of the optimal thresholds decreases fast in the early epochs, then becomes plateaued. This however is not captured by the decay functions used in Yu et al.[31] and

Du et al.[34] Looking ahead, this may be the main reason these two methods perform worse than our decay function suggested in Section 4.3. On the other hand, the shape of the thresholds curve by Andrew et al.[33] is very dissimilar to the green line, and the values deviate significantly from the optimal ones, especially in early epochs. This is understandable because this method hooks the thresholds with a particular quantile of the gradient's ℓ_2 norm. If we look again at Figure 1A, and compare it with the lower plot in Figures 2C–4C, we can see the correlation clearly. This actually reflects a common misunderstanding when using DP-SGD, that is we should set the clipping threshold close to the gradient's norm to preserve information as much as possible. This intuition works when clipping gradient in plain SGD. However in DP-SGD, noise is added after clipping and the standard deviation of the noise scales with the clipping threshold. Hence a large threshold means adding too much noise, and that outweighs the benefit of the extra amount of information.

4.3 | Strategies for setting adaptive threshold

On the basis of the results in Section 4.2, we now propose two simple strategies for setting adaptive thresholds in DP-SGD. As we mentioned in Section 1, one difficulty of adaptive clipping in DP-SGD comes from the privacy requirement. To ensure the thresholds are privacy-preserving, in Andrew et al.,[33] the threshold selection algorithm is differentially private (but needs to consume an additional privacy budget); in Yu et al.[31] and Du et al.,[34] the thresholds are calculated by a decay function independent of the training data (but the hyperparameters can affect the utility greatly). While both of our proposed strategies could avoid these shortcomings.

4.3.1 | Transfer gradient clipping threshold

The first strategy we propose is based on the observation that the threshold values can be similar across data sets when using the same network architecture. In Figure 5, the same

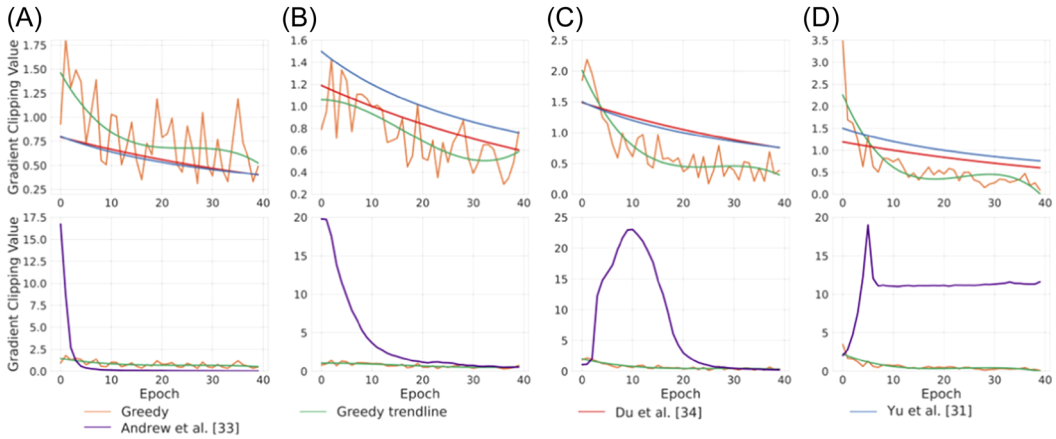

FIGURE 5 Comparison of gradient clipping thresholds found by different adaptive methods. (A) MNIST, (B) Fashion-MNIST, (C) CIFAR-10, and (D) Imagenette. [Color figure can be viewed at wileyonlinelibrary.com]

architecture is used when training with MNIST and Fashion-MNIST, also the same architecture for CIFAR-10 and CIFAR-100. We can see that in data sets using the same model architecture, the optimal gradient clipping thresholds found by the greedy algorithm are very similar in terms of the range of starting thresholds and the downward trend. This similarity suggests that we can utilize transferability: use a public data set, find the optimal thresholds using the greedy algorithm (Algorithm 1), then transfer those when training using a private data set. Since the thresholds are estimated using public data, this method does not consume an additional privacy budget. Also, the threshold decisions are better informed than calculated with blindly guessed hyperparameters as in Yu et al.[31] and Du et al.[34]

Algorithm 1 Greedy threshold finding

Input: Training data set D_r, test data set D_t, model parameters θ_{t-1} at the end of epoch t, batch size m, learning rate η_t, noise scale σ, accuracy threshold S, step size d.

Output: Optimal clipping threshold C_t for epoch t, and model parameters θ_t.

Set $C = 0.01, A = 0, C_t = C, \mathcal{A} = A, \theta = \mathbf{0}$
while $\mathcal{A} - A \leq S$ **do**
  Take a random batch B_t with the size m
  Compute Gradients
  for each $i \in B_t$, compute $\mathbf{g_t}(x_i) \leftarrow \nabla_{\theta_{t-1}} L(\theta_{t-1}, \mathbf{x}_i)$
  Clip Gradients
  $\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\| \mathbf{g}_t(x_i) \|_2}{C}\right)$
  Add Noise
  $\tilde{g}_t \leftarrow \frac{1}{m}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$
  Descent
  $\theta \leftarrow \theta_{t-1} - \eta_t \tilde{g}_t$
  Test θ to obtain test accuracy A
  if $A > \mathcal{A}$ **then**
    $\mathcal{A} = A, C_t = C, \theta_t = \theta$
  end if
  $C = C + d$
end while
Output C_t and θ_t

4.3.2 | Decay gradient clipping threshold

The above strategy, due to the multiple training iterations in the greedy algorithm, is computationally expensive. Therefore, we design a training strategy with an adaptive decay gradient threshold. This strategy is more efficient. It is similar to Yu et al.[31] and Du et al.,[34] by using a predefined decay function. However, from our experiments, the decay functions they propose are not the best fit. We choose to use a simple nonlinear function:

$$C_t = \frac{C_0}{t^a}, \quad \text{where } 0 < a \leq 1, C_0, t > 0. \tag{11}$$

In the above, C_t is the threshold for the tth epoch, C_0 is the initial threshold, and a is a parameter controlling the curvature. There are two hyperparameters that need to be set properly. The first is the initial threshold C_0. From the results in Figure 5, the optimal starting threshold varies across data sets/network architectures, and it is unlikely that there exists a universally good value for all settings. Moreover, from Figure 6, we can see that the initial threshold has a significant impact on model utility for all decay strategies. However, in Yu et al.[31] and Du et al.,[34] there is no indication of how to choose the initial threshold properly. To decide the initial threshold, we can utilize transferability again. This time, we only run the greedy algorithm for the first epoch on a public data set, then use the result as the initial threshold for training with the private data set. The second hyperparameter we need to decide is a. The intuition is that a good a should make the threshold curve match roughly the green trendline in Figure 5. In our experiments, we use $a = 0.5$ based on our experience. The value of a is robust to variation (to some extent), as we can see in Table 1.

5 | EVALUATION OF OUR STRATEGIES

In this section, we report performance evaluation results of our adaptive clipping strategies. To establish the baseline, we measured the performance of nonprivate SGD, and the fixed-threshold DP-SGD. For the fixed-threshold DP-SGD, we tested two variants: the original DP-SGD[23] and DP-SGD with the tempered sigmoid activation function.[29] The latter is the state-of-the-art for DP-SGD training with a fixed threshold. We compared our strategies against four state-of-the-art adaptive clipping methods.[31–34]

5.1 | Experimental setup

5.1.1 | Data sets

The performance evaluation was performed on four widely used data sets: MNIST, Fashion-MNIST, CIFAR-10, and Imagenette. The default training/testing split was used. More details of the data sets can be found below. To obtain clipping thresholds through transferability, we use

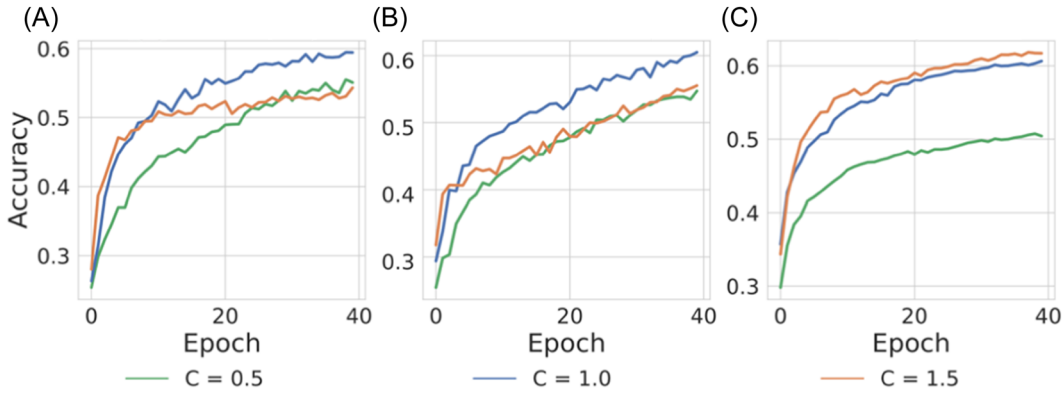

FIGURE 6 Model accuracy of the three methods with different initial clipping thresholds C ($\varepsilon = 3.0$, data set CIFAR-10). (A) Yu et al.,[31] (B) Du et al.,[34] and (C) Decay (ours). [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 1 Model accuracy of the decay strategy with different a, privacy budget $\epsilon = 2.93$, 2.7, 7.53, and 130.5 for the four data sets

	$a = 0.3$	$a = 0.4$	$a = 0.5$	$a = 0.6$
MNIST	96.9 ± 0.0	97.3 ± 0.1	97.7 ± 0.1	97.4 ± 0.0
Fashion-MNIST	86.4 ± 0.3	86.2 ± 0.2	87.1 ± 0.1	86.2 ± 0.2
CIFAR-10	66.9 ± 0.2	67.3 ± 0.2	68.2 ± 0.0	67.4 ± 0.4
Imagenette	62.6 ± 0.3	62.3 ± 0.4	64.0 ± 0.1	63.7 ± 0.5

MNIST for Fashion-MNIST and vice versa, we also use CIFAR-100 (transfer to CIFAR-10) and images of other 10 classes from ImageNet (transfer to Imagenette—see Appendix C).

- *MNIST*: This is a data set of 60,000 28×28 grayscale images of the 10 digits, along with a test set of 10,000 images.
- *Fashion-MNIST*: This is a data set of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28×28 grayscale image, associated with a label from 10 classes.
- *CIFAR-10*: This is a subset of the Tiny Images data set and consists of 60,000 32×32 color images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images.
- *Imagenette*: This is a subset of 10 easily classified classes from Imagenet (tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, and parachute). For details see Appendix C.

5.1.2 | Hardware and software

All experiments were conducted on a personal workstation with an Intel Xeon Gold 6248R 3.00 GHz CPU, an NVIDIA Tesla V100S-PCIE-32GB GPU and 128 GB memory. We implemented all algorithms on top of Opacus[2].

5.1.3 | Network and training details

For each of MNIST, Fashion-MNIST, and CIFAR-10 data sets, we used the same CNNs as in Papernot et al.[29] For Imagenette, we used Alexnet. The detailed setting of the network structures can be found in Appendix B. As for training details, for the greedy algorithm (Algorithm 1), the hyperparameters C (starting threshold), d (step size), and S (accuracy threshold) were 0.05, 0.01, and 0.02, respectively, for the first three data sets. For Imagenette, the hyperparameters of the greedy algorithm are $C = 2.0, d = 0.5, S = 0.06$. For our decay strategy, we run the greedy algorithm for the first epoch five times and take the average of the optimal thresholds as the initial threshold. For other adaptive clipping methods that require hyperparameter tuning, we ran the experiments either with hyperparameters recommended by the authors (if available), or with different hyperparameter combinations then report the best results.

5.1.4 | Hyperparameter settings

In this paper, we set the batch size of all experiments of the MNIST, Fashion-MNIST, and CIFAR-10 data sets to 1024 and the learning rate to 1, and for the Imagenette data set, we set these two hyperparameters to 64 and 0.01.

Other hyperparameters include (due to space limit, Tables D1–D4 can be found in Appendix D):

- The fixed clipping thresholds (used by Abadi et al.[23] and Papernot et al.[29]) for the experiments that produced the results in Table 2. They are listed in Table D1. Those used for the experiments that produced the results in Table 3 are listed in Table D3.
- The initial clipping thresholds (used by Yu et al.[31] and Du et al.[34]) for the experiments that produced the results in Table 2. They are also listed in Table D1. Those used for the experiments that produced the results in Table 3 are listed in Table D3.
- For Pichapati et al.,[32] we set two hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$ as recommended by the authors. For the other two hyperparameters h_1, h_2, the values that gave the best results for Tables 2 and 3 are listed in Tables D2 and D4, respectively.
- For Andrew et al.,[33] a hyperparameter η_c was set to 0.2 as recommended by the authors. For another hyperparameter γ, the values that gave the best results for Tables 2 and 3 are listed in Tables D2 and D4, respectively.

TABLE 2 Performance comparison under different privacy budgets ϵ

ϵ	Abadi et al.[23]	Papernot et al.[29]	Yu et al.[31]	Pichapati et al.[32]	Andrew et al.[33]	Du et al.[34]	Transfer (ours)	Decay (ours)
MNIST								
1.2	95.0 ± 0.1	95.5 ± 0.2	95.1 ± 0.1	75.8 ± 0.4	93.4 ± 0.3	94.4 ± 0.2	96.0 ± 0.1	**96.2 ± 0.1**
2.93	97.2 ± 0.1	97.6 ± 0.1	96.7 ± 0.0	83.9 ± 0.4	94.0 ± 0.1	96.6 ± 0.1	**98.1 ± 0.1**	97.7 ± 0.1
4.5	97.2 ± 0.2	97.8 ± 0.1	96.9 ± 0.1	87.0 ± 0.4	96.8 ± 0.1	97.2 ± 0.1	**98.2 ± 0.0**	98.1 ± 0.0
Fashion-MNIST								
1.5	82.7 ± 0.3	83.9 ± 0.2	84.0 ± 0.2	69.6 ± 0.2	80.9 ± 0.5	84.4 ± 0.1	84.3 ± 0.1	**84.7 ± 0.3**
2.7	85.2 ± 0.2	86.2 ± 0.1	86.0 ± 0.1	71.8 ± 0.4	85.6 ± 0.3	86.1 ± 0.1	86.6 ± 0.0	**87.7 ± 0.1**
4.5	86.0 ± 0.3	87.4 ± 0.1	86.3 ± 0.2	75.5 ± 0.3	86.2 ± 0.1	87.1 ± 0.2	**87.7 ± 0.1**	87.5 ± 0.1
CIFAR-10								
3.0	55.0 ± 0.2	60.5 ± 0.3	59.1 ± 0.3	27.6 ± 0.5	54.7 ± 0.3	60.1 ± 0.3	**63.1 ± 0.3**	62.1 ± 0.1
6.78	61.2 ± 0.3	64.6 ± 0.2	63.0 ± 0.2	32.3 ± 0.3	52.3 ± 0.5	64.5 ± 0.3	**67.2 ± 0.2**	67.1 ± 0.3
7.53	62.9 ± 0.3	66.1 ± 0.1	64.7 ± 0.4	32.6 ± 0.1	63.4 ± 0.4	65.4 ± 0.2	68.1 ± 0.1	**68.2 ± 0.0**
Imagenette								
11.0	39.3 ± 0.5	49.2 ± 0.4	45.3 ± 0.4	12.9 ± 0.0	45.7 ± 0.3	49.5 ± 0.2	**55.8 ± 0.1**	55.6 ± 0.3
46.6	46.2 ± 0.5	56.0 ± 0.4	50.3 ± 0.4	15.3 ± 0.2	50.1 ± 0.2	51.5 ± 0.5	**60.8 ± 0.1**	60.2 ± 0.3
130.5	48.0 ± 0.4	58.4 ± 0.1	55.1 ± 0.5	21.2 ± 0.2	50.9 ± 0.4	52.9 ± 0.4	63.8 ± 0.4	**64.0 ± 0.1**

Note: The bold data indicates the best performance.

TABLE 3 Performance on alternative network structures, privacy budgets $\epsilon = 1.0$, 1.0, 1.5, and 11.0, respectively

	ϵ	Abadi et al.[23]	Papernot et al.[29]	Yu et al.[31]	Pichapati et al.[32]	Andrew et al.[33]	Du et al.[34]	Transfer (ours)	Decay (ours)
MNIST	1.0	93.0 ± 0.2	95.1 ± 0.2	94.5 ± 0.2	67.5 ± 0.4	93.4 ± 0.3	95.2 ± 0.2	96.1 ± 0.0	$\mathbf{96.3 \pm 0.1}$
Fashion-MNIST	1.0	81.2 ± 0.2	82.4 ± 0.2	81.4 ± 0.1	63.5 ± 0.2	82.5 ± 0.2	83.5 ± 0.2	84.1 ± 0.1	$\mathbf{84.2 \pm 0.1}$
CIFAR-10	1.5	52.1 ± 0.1	53.4 ± 0.2	50.5 ± 0.2	20.2 ± 0.2	53.2 ± 0.3	54.0 ± 0.1	$\mathbf{56.2 \pm 0.3}$	55.5 ± 0.2
Imagenette	11.0	34.9 ± 0.6	42.9 ± 0.2	35.6 ± 0.2	–	37.6 ± 0.2	40.5 ± 0.1	$\mathbf{46.2 \pm 0.1}$	46.0 ± 0.1

Note: The bold data indicates the best performance.

5.2 | Experimental results

We now report the results. All data reported are based on the measurements from five runs.

5.2.1 | Performance analysis

We first compared the performance of different methods under the same privacy constraints (the same DP budget ϵ). We used $\epsilon = 2.93, 2.7,$ and 7.53 for MNIST, Fashion-MNIST, and CIFAR-10, as suggested by Papernot et al.[29] For Imagenette, we used $\epsilon = 130.5$. Another DP parameter δ was set to $1/|\mathcal{D}|$, where $|\mathcal{D}|$ is the size of the training set. The other parameters were set accordingly with respect to (ϵ, δ). The fixed clip thresholds for Abadi et al.[23] are 0.3, 0.3, 1.0, and 10.0, respectively, for the four data sets, and for Papernot et al.[29] are 0.4, 0.5, 0.6, and 10.0. And under the same network, the test accuracies without privacy in these four data sets are 98.9%, 89.8%, 76.8%, and 70.7%, respectively. As we can see in Figure 7, the performance of all DP methods is worse than nonprivate SGD, and the gap is more significant in more complex data sets. Our two strategies have similar performance. Both of them performed better than the fixed-threshold baseline.[29] The performance of Yu et al.,[31] Andrew et al.,[33] and Du et al.[34] is generally worse than Papernot et al.,[29] but is better than the original DP-SGD[23] with a fixed threshold. The performance of Andrew et al.[33] is close to Abadi et al.[23] The performance of Pichapati et al.[32] is much worse than all other methods. The main reason could be that the algorithm adds too much noise. The noise in Pichapati et al.[32] is added to each sample (see Algorithm 1, lines 9–12 in the original paper), rather than each mini-batch as in other methods. We can also see from Figure 7 that our strategies converge faster than other methods with DP. This is likely a consequence of setting large (but not overly so) clip thresholds at the early stage of training. Fast convergence is beneficial to privacy as well: We can stop training earlier hence the overall ϵ, which is the sum of the budget for each epoch, can be reduced. Therefore, in some scenarios[45,46] where the number of training epochs needs to be limited to save computing resources, the advantages of our two strategies will be more obvious.

5.2.2 | Effect of privacy budget

In Section 5.2.1, we only analyze the performance of each data set on one privacy budget. To better demonstrate the superiority of our two strategies compared with other methods.

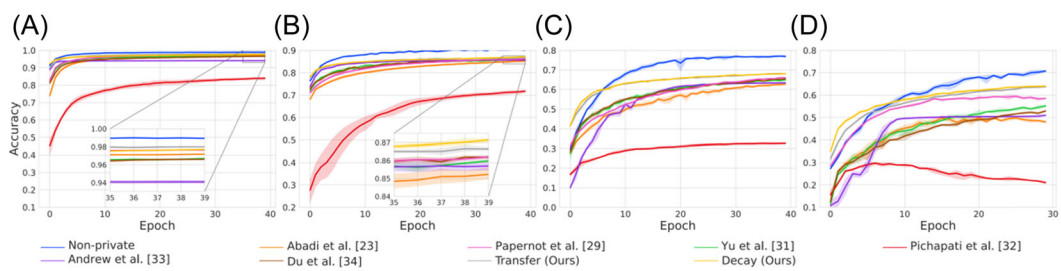

FIGURE 7 Comparison of the accuracy performance of different methods, where differential privacy training consumes a privacy budget ϵ of 2.93, 2.7, 7.53, and 130.5 in these four data sets, respectively. (A) MNIST, (B) Fashion-MNIST, (C) CIFAR-10, and (D) Imagenette. [Color figure can be viewed at wileyonlinelibrary.com]

We further show how the methods perform with different privacy levels. The results are displayed in Table 2 and Figures 8 and 9. For each data set, we varied the privacy budget ϵ, and measured the test accuracy. As we can see, our two strategies consistently outperformed the baselines and the other adaptive clipping methods. The advantage is more significant when training with complex data sets (CIFAR-10 and Imagenette). Again, our two strategies will make the model converge more quickly and the performance of our transfer strategy and that of our decay strategy are quite close. Given that the transfer strategy is computationally intensive, the decay strategy could be a better choice in many cases.

5.2.3 | Effect of model structure

Lastly, we show the performance of our strategies when using other network structures. For MNIST, Fashion-MNIST and CIFAR-10, we used a smaller network with two convolutional, two pooling, and two fully connected layers, and for the Imagenette data set, we used a larger ResNet18 (removed the BN layer as the BN layer affects privacy). The specific network parameters can be found in Appendix B. We run the greedy algorithm on those networks to obtain transferable thresholds for our strategies. We compared the performance against the fixed-threshold baselines as well as the adaptive clipping methods. We were not able to obtain the result for Pichapati et al.[32] on Imagenette due to an out-of-memory error. The results are shown in Table 3 and Figure 10. As we can see, our strategies still perform better than all the

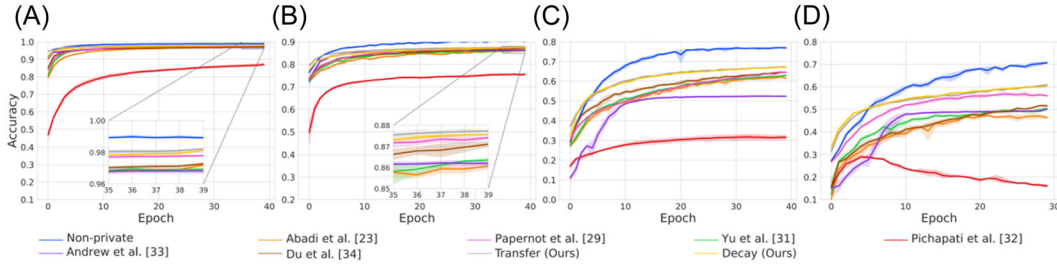

FIGURE 8 A comparison of the accuracy performance of different methods, where differential privacy training consumes a privacy budget ϵ of 4.5, 4.5, 6.78, and 46.6 in these four data sets, respectively. (A) MNIST, (B) Fashion-MNIST, (C) CIFAR-10, and (D) Imagenette. [Color figure can be viewed at wileyonlinelibrary.com]

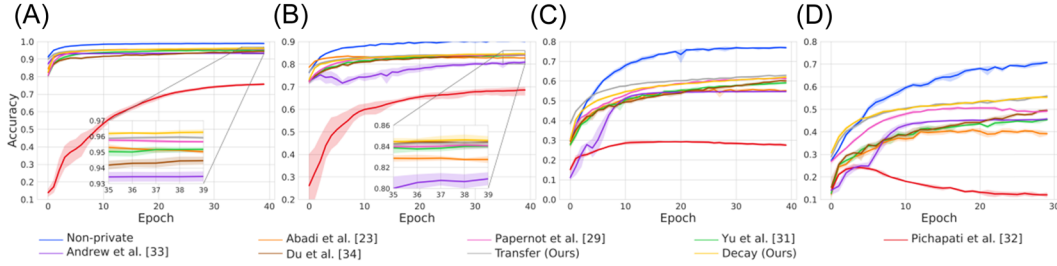

FIGURE 9 A comparison of the accuracy performance of different methods, where differential privacy training consumes a privacy budget ϵ of 1.2, 1.5, 3.0, and 11.0 in these four data sets, respectively. (A) MNIST, (B) Fashion-MNIST, (C) CIFAR-10, and (D) Imagenette. [Color figure can be viewed at wileyonlinelibrary.com]

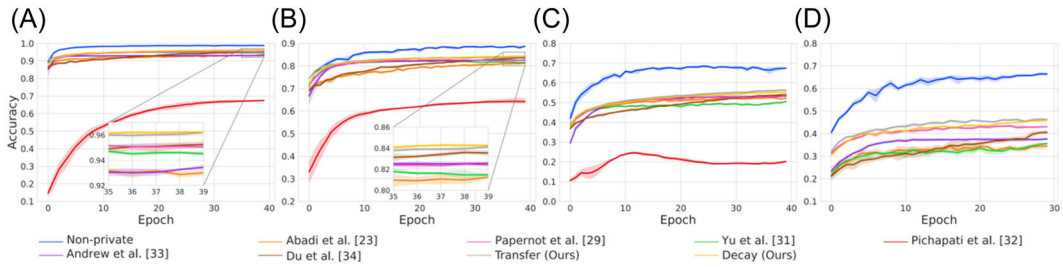

FIGURE 10 A comparison of the accuracy performance of different methods, where differential privacy training consumes a privacy budget ϵ of 1.0, 1.0, 1.5, and 11.0 in these four data sets, respectively. (A) MNIST, (B) Fashion-MNIST, (C) CIFAR-10, and (D) Imagenette. [Color figure can be viewed at wileyonlinelibrary.com]

others, and the other experimental performances are also similar to those described in Section 5.2.2.

6 | CONCLUSION

In this paper, we conducted an empirical study on adaptive clipping in DP-SGD. Through experiments, we confirmed the intuition that adaptive clipping thresholds could improve the utility when training differentially private models. We also uncovered new design insights that were not captured by existing literature. These novel findings enabled us to propose two simple yet effective adaptive clipping strategies, that consistently outperform the state-of-the-art. We hope our study could shed a light on future theoretical analysis of gradient clipping in private settings as well.

ORCID

Guanbiao Lin http://orcid.org/0000-0002-7372-7345
Hongyang Yan http://orcid.org/0000-0002-1493-9671
Teng Huang http://orcid.org/0000-0001-7261-6398

ENDNOTES

* Pichapati et al.[32] not included here because it uses a fixed overall threshold but adaptive clips the per-coordinate gradient.

† Opacus is an open source library provided by Facebook that implements DP-SGD in the Pytorch framework.

‡ The Imagenette data set. https://github.com/fastai/imagenette.

REFERENCES

1. He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE International Conference on Computer Vision*; 2015: 1026-1034.
2. Zheng W, Yan L, Gou C, Wang FY. Fighting fire with fire: a spatial-frequency ensemble relation network with generative adversarial learning for adversarial image classification. *Int J Intell Syst*. 2021;36(5): 2081-2121.
3. Mikolov T, Karafiát M, Burget L, Cernockỳ J, Khudanpur S. Recurrent neural network based language model. In: *Interspeech*. Vol 2. Makuhari; 2010:1045-1048.

4. Ai S, Hong S, Zheng X, Wang Y, Liu X. CSRT rumor spreading model based on complex network. *Int J Intell Syst*. 2021;36(5):1903-1913.
5. Hu L, Yan H, Li L, Pan Z, Liu X, Zhang Z. MHAT: an efficient model-heterogeneous aggregation training scheme for federated learning. *Inf Sci*. 2021;560:493-503.
6. Deng L, Hinton G, Kingsbury B. New types of deep neural network learning for speech recognition and related applications: an overview. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE; 2013:8599-8603.
7. Qin X, Tan S, Tang W, Li B, Huang J. Image steganography based on iterative adversarial perturbations onto a synchronized-directions sub-image. In: *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2021)*. IEEE; 2021:2705-2709.
8. Jiang N, Jie W, Li J, Liu X, Jin D. GATrust: a multi-aspect graph attention network model for trust assessment in OSNs. *IEEE Trans Knowl Data Eng*; 2022.
9. Tianqing Z, Zhou W, Ye D, Cheng Z, Li J. Resource allocation in IoT edge computing via concurrent federated reinforcement learning. *IEEE Internet Things J*. 2021;9(2):1414-1426.
10. Vartak M, Huang S, Siddiqui T, Madden S, Parameswaran A. Towards visualization recommendation systems. *ACM SIGMOD Record*. 2017;45(4):34-39.
11. Zhu T, Li J, Hu X, Xiong P, Zhou W. The dynamic privacy-preserving mechanisms for online dynamic social networks. *IEEE Trans Knowl Data Eng*. 2020;34(6):2962-2974.
12. Li J, Ye H, Li T, et al. Efficient and secure outsourcing of differentially private data publishing with multiple evaluators. *IEEE Trans Dependable Secure Comput*; 2020.
13. Shokri R, Stronati M, Song C, Shmatikov V. Membership inference attacks against machine learning models. In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE; 2017:3-18.
14. Hitaj B, Ateniese G, Perez-Cruz F. Deep models under the GAN: information leakage from collaborative deep learning. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*; 2017:603-618.
15. Salem A, Zhang Y, Humbert M, Berrang P, Fritz M, Backes M. ML-leaks: model and data independent membership inference attacks and defenses on machine learning models. In: *Network and Distributed Systems Security (NDSS) Symposium*; 2019.
16. Huang T, Zhang Q, Liu J, Hou R, Wang X, Li Y. Adversarial attacks on deep-learning-based SAR image target recognition. *J Network Comput Appl*. 2020;162:102632.
17. Zhu L, Liu Z, Han S. Deep leakage from gradients. Adv Neural Inf Process Syst. 2019;32.
18. Mo K, Liu X, Huang T, Yan A. Querying little is enough: model inversion attack via latent information. *Int J Intell Syst*. 2021;36(2):681-690.
19. Mo K, Tang W, Li J, Yuan X. Attacking deep reinforcement learning with decoupled adversarial policy. *IEEE Trans Dependable Secure Comput*; 2022.
20. Dwork C, McSherry F, Nissim K, Smith A. Calibrating noise to sensitivity in private data analysis. In: *Theory of Cryptography Conference*. Springer; 2006:265-284.
21. Xin B, Geng Y, Hu T, et al. Federated synthetic data generation with differential privacy. *Neurocomputing*. 2022;468:1-10.
22. Hu C, Li J, Liu Z, et al. How to make private distributed cardinality estimation practical, and get differential privacy for free. In: *30th USENIX Security Symposium (USENIX Security 21)*; 2021:965-982.
23. Abadi M, Chu A, Goodfellow I, et al. Deep learning with differential privacy. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*; 2016:308-318.
24. Leino K, Fredrikson M. Stolen memories: leveraging model memorization for calibrated white-box membership inference. In: *29th USENIX Security Symposium (USENIX Security 20)*; 2020:1605-1622.
25. Jayaraman B, Evans D. Evaluating differentially private machine learning in practice. In: *28th USENIX Security Symposium (USENIX Security 19)*; 2019:1895-1912.
26. Pan Z, Hu L, Tang W, Li J, He Y, Liu Z. Privacy-preserving multi-granular federated neural architecture search a general framework. *IEEE Trans Knowl Data Eng*; 2021.
27. Tang W, Li B, Barni M, Li J, Huang J. An automatic cost learning framework for image steganography using deep reinforcement learning. *IEEE Trans Inf Forensics Secur*. 2020;16:952-967.
28. Li T, Li J, Chen X, Liu Z, Lou W, Hou YT. NPMML: a framework for non-interactive privacy-preserving multi-party machine learning. *IEEE Trans Dependable Secure Comput*. 2020;18(6):2969-2982.

29. Papernot N, Thakurta A, Song S, Chien S, Erlingsson Ú. Tempered sigmoid activations for deep learning with differential privacy. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol 35; 2021: 9312-9321.
30. van der Veen KL, Seggers R, Bloem P, Patrini G. Three tools for practical differential privacy. In: *NeurIPS 2018 Workshop*; 2018.
31. Yu D, Zhang H, Chen W. Improve the gradient perturbation approach for differentially private optimization. In: *NeurIPS 2018 Workshop*; 2018.
32. Pichapati V, Suresh AT, Yu FX, Reddi SJ, Kumar S. AdaCliP: adaptive clipping for private SGD. *arXiv preprint arXiv:1908.07643*. 2019.
33. Andrew G, Thakkar O, McMahan HB, Ramaswamy S. Differentially private learning with adaptive clipping. In: *NeurIPS*; 2021.
34. Du J, Li S, Feng M, Chen S. Dynamic differential-privacy preserving SGD. *arXiv preprint arXiv:2111.00173*. 2021.
35. Gondara L, Carvalho RS, Wang K. Training differentially private neural networks with lottery tickets. In: *European Symposium on Research in Computer Security*. Springer; 2021: 543-562.
36. Frankle J, Carbin M. The lottery ticket hypothesis: finding sparse, trainable neural networks. In: *International Conference on Learning Representations*; 2018.
37. Zhou Y, Wu S, Banerjee A. Bypassing the ambient dimension: private SGD with gradient subspace identification. In: *International Conference on Learning Representations*; 2020.
38. Yu D, Zhang H, Chen W, Liu TY. Do not let privacy overbill utility: gradient embedding perturbation for private learning. In: *International Conference on Learning Representations*; 2020.
39. Tramer F, Boneh D. Differentially private learning needs better features (or much more data). In: *International Conference on Learning Representations*; 2021.
40. Chen X, Wu SZ, Hong M. Understanding gradient clipping in private SGD: a geometric perspective. In: *NeurIPS*; 2020.
41. Song S, Thakkar O, Thakurta A. Characterizing private clipped gradient descent on convex generalized linear problems. *arXiv e-prints*. 2020: arXiv-2006.
42. Li JQ, Du Y, Gao KZ, et al. A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem. *IEEE Trans Autom Sci Eng*; 2021.
43. Kumar R, Moseley B, Vassilvitskii S, Vattani A. Fast greedy algorithms in mapreduce and streaming. *ACM Trans Parallel Comput (TOPC)*. 2015;2(3):1-22.
44. Wang X, Kuang X, Li J, Li J, Chen X, Liu Z. Oblivious transfer for privacy-preserving in VANET's feature matching. *IEEE Trans Intell Transp Syst*. 2020;22(7):4359-4366.
45. Li M, Soltanolkotabi M, Oymak S. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In: *International Conference on Artificial Intelligence and Statistics*. PMLR; 2020:4313-4324.
46. Li M, Yan C, Liu W, Liu X. An early warning model for customer churn prediction in telecommunication sector based on improved bat algorithm to optimize ELM. *Int J Intell Syst*. 2021;36(7):3401-3428.

How to cite this article: Lin G, Yan H, Kou G, et al. Understanding adaptive gradient clipping in DP-SGD, empirically. *Int J Intell Syst*. 2022;1-27. doi:10.1002/int.23001

APPENDIX A: DP-SGD ALGORITHM

The DP-SGD algorithm[23] is shown in Algorithm 2.

Algorithm 2 Differentially private SGD

Input: Examples D, loss function L, parameters θ, batch size m, learning rate η_t, noise scale σ, gradient norm bound C

Output: θ_T and the overall privacy cost (ε, δ) using a privacy accounting method.

Initial θ_0 randomly

for $t \in [T]$ **do**

  Take a random batch B_t with the size m

  Compute Gradients

  for each $i \in B_t$, compute $\mathbf{g_t}(x_i) \leftarrow \nabla_{\theta_t} L(\theta_t, \mathbf{x}_i)$

  Clip Gradients

  $\overline{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\| \mathbf{g}_t(x_i) \|_2}{C}\right)$

  Add Noise

  $\tilde{g}_t \leftarrow \frac{1}{m}\left(\sum_i \overline{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$

  Descent

  $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{g}_t$

end for

APPENDIX B: NETWORK STRUCTURE PARAMETER

For all our experiments except those for Table 3, we use the same CNN structures suggested in Papernot et al.[29] For MNIST and Fashion-MNIST, the network structure is shown in Table B1. For CIFAR-10 and CIFAR-100, the network structure is shown in Table B2. For Table 3 experiments, the network structures are shown in Tables B3 and B4.

TABLE B1 CNN model for MNIST and Fashion-MNIST

Layer	Parameters
Convolution	16 filters of 3×3, stride 2, padding 1
Max-Pooling	2×2, stride 1
Convolution	32 filters of 3×3, stride 1, padding 1
Max-Pooling	2×2, stride 1
Fully connected	32 units
Fully connected	32 units

Abbreviation: CNN, convolutional neural network.

TABLE B2 CNN model for CIFAR-10 and CIFAR-100

Layer	Parameters
Convolution × 2	32 filters of 3 × 3, stride 1, padding 1
Max-Pooling	2 × 2, stride 2
Convolution × 2	64 filters of 3 × 3, stride 1, padding 1
Max-Pooling	2 × 2, stride 2
Convolution × 2	128 filters of 3 × 3, stride 1, padding 1
Max-Pooling	2 × 2, stride 2
Fully connected	128 units
Fully connected	10 units

Abbreviation: CNN, convolutional neural network.

TABLE B3 Smaller CNN model for MNIST and Fashion-MNIST

Layer	Parameters
Convolution	16 filters of 5 × 5, stride 1, padding 0
Avg-Pooling	2 × 2, stride 2
Convolution	32 filters of 5 × 5, stride 1, padding 0
Avg-Pooling	2 × 2, stride 2
Fully connected	512 units
Fully connected	10 units

Abbreviation: CNN, convolutional neural network.

TABLE B4 Smaller CNN model for CIFAR-10 and CIFAR-100

Layer	Parameters
Convolution	32 filters of 5 × 5, stride 1, padding 2
Max-Pooling	2 × 2, stride 2
Convolution	32 filters of 5 × 5, stride 1, padding 0
Max-Pooling	2 × 2, stride 2
Fully connected	128 units
Fully connected	10 units

Abbreviation: CNN, convolutional neural network.

APPENDIX C: IMAGENETTE AND ImageNet SUBSET

Imagenette[‡] is a subset of 10 easily classified classes from Imagenet (tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, and parachute). For details see Table C1.

TABLE C1 Imagenette data set

Category ID	#Name	#Sample	#Train sample	#Test sample
n01440764	Tench	1350	963	387
n02102040	English springer	1350	955	395
n02979186	Cassette player	1350	993	357
n03000684	Chain saw	1244	858	386
n03028079	Church	1350	941	409
n03394916	French horn	1350	956	394
n03417042	Garbage truck	1350	961	389
n03425413	Gas pump	1350	931	419
n03445777	Golf ball	1350	951	399
n03888257	Parachute	1350	960	390

TABLE C2 The 10 classes in ImageNet used for transferability

Category ID	#Name	#Sample	#Train sample	#Test sample
n01443537	Goldfish	1300	1040	260
n02102177	Spaniel	1300	1040	260
n02123045	Tabby	1299	1040	259
n02129604	Tiger	1300	1040	260
n02690373	Airliner	1300	1040	260
n02951358	Canoe	1270	1016	254
n03197337	Digital watch	876	701	175
n03393912	Freight car	1275	1020	255
n03478589	Half track	1300	1040	260
n04461696	Tow truck	1277	1022	255

When conducting the transfer experiments, we selected 10 other classes in ImageNet for the experiments, namely (goldfish, spaniel, tabby, tiger, airliner, canoe, digital watch, freight car, half track, and tow truck). For details see Table C2.

APPENDIX D: HYPERPARAMETER SETTINGS

We described the setting of hyperparameters in Section 5.1.4. In this appendix, we show the detailed setting of these parameters in Tables D1–D4.

TABLE D1 Fixed clipping gradient threshold and initial gradient clipping threshold for the experiment of Table 2

	Data set											
	MNIST			Fashion-MNIST			CIFAR-10			Imagenette		
Methods	$\epsilon = 1.2$	$\epsilon = 2.93$	$\epsilon = 4.5$	$\epsilon = 1.5$	$\epsilon = 2.7$	$\epsilon = 4.5$	$\epsilon = 3.0$	$\epsilon = 6.78$	$\epsilon = 7.53$	$\epsilon = 11.0$	$\epsilon = 46.6$	$\epsilon = 130.5$
Abadi et al.[23]	0.6	0.3	0.5	0.3	0.3	0.8	1.0	1.0	1.0	5.0	7.0	10.0
Papernot et al.[29]	0.5	0.5	0.6	0.7	0.4	0.8	0.6	0.5	0.6	5.0	7.0	10.0
Yu et al.[31]	0.6	0.8	0.8	0.8	1.0	1.0	1.0	1.2	1.5	6.0	10.0	10.0
Du et al.[34]	0.8	0.8	1.2	0.8	1.0	1.2	1.0	1.5	1.5	5.0	6.0	6.0

TABLE D2 Settings of hyperparameters h_1 and h_2 and the settings of hyperparameter γ are in the experiments of Table 2

	Data set										
	MNIST			**Fashion-MNIST**			**CIFAR-10**			**Imagenette**		
Parameters	$\epsilon = 1.2$	$\epsilon = 2.93$	$\epsilon = 4.5$	$\epsilon = 1.5$	$\epsilon = 2.7$	$\epsilon = 4.5$	$\epsilon = 3.0$	$\epsilon = 6.78$	$\epsilon = 7.53$	$\epsilon = 11.0$	$\epsilon = 46.6$	$\epsilon = 130.5$
h_1	1e − 2	1e − 2	1e − 2	1e − 3	1e − 3	1e − 2	1e − 3	1e − 3	1e − 3	1e − 3	1e − 3	1e − 3
h_2	1e − 4	1e − 4	1e − 4	1e − 5	1e − 5	1e − 4	1e − 5	1e − 5	1e − 5	1e − 5	1e − 5	1e − 5
γ	0.7	0.7	0.9	0.7	0.7	0.7	0.1	0.1	0.3	0.025	0.04	0.05

TABLE D3 Fixed clipping gradient threshold and initial gradient clipping threshold for the experiment of Table 3

Methods	Data set			
	MNIST	Fashion-MNIST	CIFAR-10	Imagenette
Abadi et al.[23]	0.7	0.7	0.7	4.0
Papernot et al.[29]	0.7	0.7	0.6	5.0
Yu et al.[31]	0.8	0.8	1.0	5.0
Du et al.[34]	0.8	0.8	1.0	4.0

TABLE D4 Settings of hyperparameters h_1 and h_2 and the settings of hyperparameter γ are in the experiments Table 3

Parameters	Data set			
	MNIST	Fashion-MNIST	CIFAR-10	Imagenette
h_1	1e − 4	1e − 3	1e − 3	–
h_2	1e − 5	1e − 5	1e − 5	–
γ	0.7	0.6	0.1	0.02

APPENDIX E: ADDITIONAL EXPERIMENTAL RESULTS

In this appendix, we show a comparison of the optimal clipping thresholds found by Greedy algorithms and other adaptive algorithms on the Imagenette data set in Figure E1 and the optimal clipping thresholds found by these methods on four data sets after the network structure is changed in Figure E2.

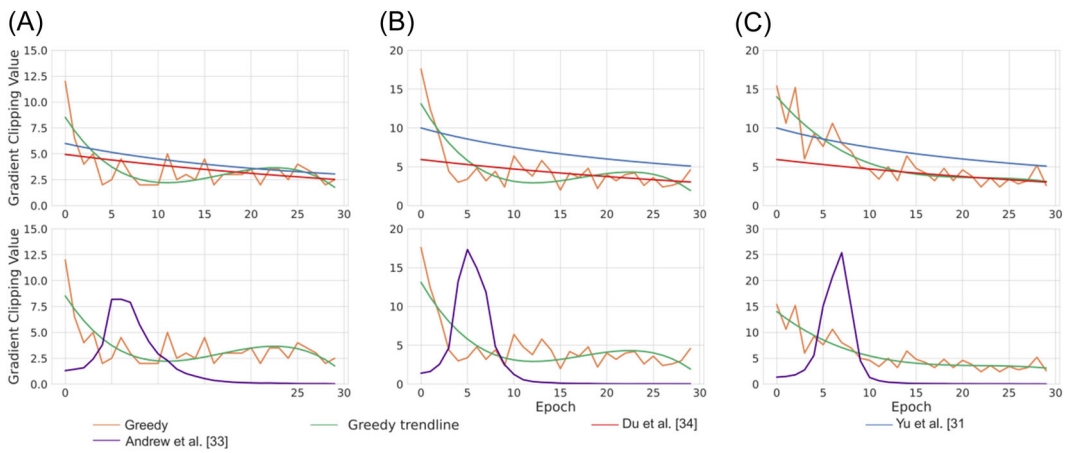

FIGURE E1 Comparison of gradient clipping thresholds with different adaptive methods for fixed privacy budget ε = (A) 11.0, (B) 46.6, and (C) 130.5 on Imagenette data set [Color figure can be viewed at wileyonlinelibrary.com]

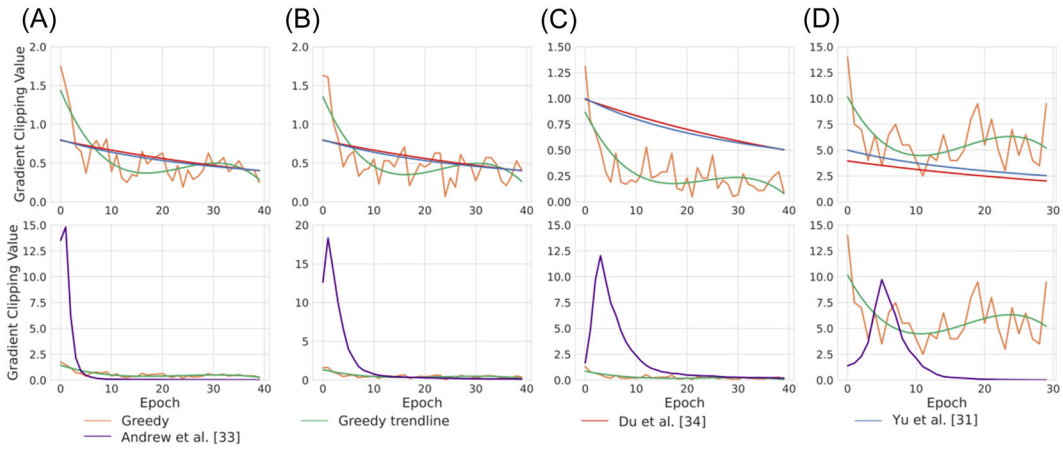

FIGURE E2 Comparison of gradient clipping thresholds with different adaptive methods for fixed privacy budget $\epsilon = 1.0$, 1.0, 1.5, and 11.0 on MNIST, Fashion-MNIST, CIFAR-10, and Imagenette data sets, respectively. (A) MNIST, (B) Fashion-MNIST, (C) CIFAR-10, and (D) Imagenette. [Color figure can be viewed at wileyonlinelibrary.com]