A Theory to Instruct Differentially-Private Learning via Clipping Bias Reduction

Hanshen Xiao *§, Zihang Xiang †§, Di Wang † and Srinivas Devadas *
* {hsxiao, devadas}@mit.edu, MIT
† {zihang.xiang, di.wang}@kaust.edu.sa, KAUST

***Abstract*—We study the bias introduced in Differentially-Private Stochastic Gradient Descent (DP-SGD) with clipped or normalized per-sample gradient. As one of the most popular but artificial operations to ensure bounded sensitivity, gradient clipping enables composite privacy analysis of many iterative optimization methods without additional assumptions on either learning models or input data. Despite its wide applicability, gradient clipping also presents theoretical challenges in systematically instructing improvement of privacy or utility. In general, without an assumption on globally-bounded gradient, classic convergence analyses do not apply to clipped gradient descent. Further, given limited understanding of the utility loss, many existing improvements to DP-SGD are heuristic, especially in the applications of private deep learning.**

In this paper, we provide meaningful theoretical analysis validated by thorough empirical results of DP-SGD. We point out that the bias caused by gradient clipping is underestimated in previous works. For generic non-convex optimization via DP-SGD, we show one key factor contributing to the bias is the *sampling noise* of stochastic gradient to be clipped. Accordingly, we use the developed theory to build a series of improvements for sampling noise reduction from various perspectives. From an optimization angle, we study variance reduction techniques and propose *inner-outer momentum*. At the learning model (neural network) level, we propose several tricks to enhance network internal normalization and *BatchClipping* to carefully clip the gradient of a batch of samples. For data preprocessing, we provide theoretical justification of recently proposed improvements via data normalization and (self-)augmentation.

Putting these systematic improvements together, private deep learning via DP-SGD can be significantly strengthened in many tasks. For example, in computer vision applications, with an $(\epsilon = 8, \delta = 10^{-5})$ DP guarantee, we successfully train ResNet20 on CIFAR10 and SVHN with test accuracy 76.0% and 90.1%, respectively; for natural language processing, with $(\epsilon = 4, \delta = 10^{-5})$, we successfully train a recurrent neural network on IMDb data with test accuracy 77.5%.

1. Introduction

Given access to representative data with proper preprocessing, deep learning has witnessed remarkable success. In certain image recognition tasks, well-trained neural networks can already achieve human-level performance [1], [2]. Meanwhile, the trend of scaling up machine learning systems is accelerating. From 2018-2022, the size of notable models for natural language processing increased by five orders of magnitude [3]. Accompanied by emerging and practical applications, privacy leakage from models trained over users' data has also received increasing attention. Without proper privacy preservation, many practical attacks are known to efficiently identify or even recover the data used for training [4], [5], [6], [7]. To formalize the privacy leakage when releasing sensitive information, Differential Privacy (DP) [8] provides a rigorous metric. However, as larger models are employed, the gap between the boosted performance in the non-private regime and that achievable with meaningful DP guarantees is widening [9], [10]. It still largely remains open whether such utility loss is fundamental to current private learning frameworks. To answer this question, we first need to understand the key challenges in DP privatization, especially for high-dimensional statistical learning.

At a high level, DP sets out to ensure that, from the output of a mechanism, one cannot distinguish the participation of an arbitrary individual in the input dataset. Rooted in cryptography, DP formalizes such distinguishing advantage by measuring the worst-case difference between the output distributions of two arbitrary adjacent datasets. Here, we call two datasets adjacent if they only differ in one datapoint. Based on different metrics, DP and its variants were widely studied over last two decades. For example, the classic ϵ-DP [8] applies maximal divergence as the measurement, while (α, ϵ) Rényi-DP [11] is developed on Rényi divergence to produce tighter composition bounds. To concretely control DP security parameters, one key quantity that needs to be determined is *sensitivity*. Roughly speaking, sensitivity captures the maximal change to a function's output after one arbitrarily changes a single datapoint in the input set. Given sensitivity, one popular privatization method is perturbation, such as adding Laplace or Gaussian noise proportionally [8], [12]. Unfortunately, the computation of sensitivity is in general NP-hard [13]. Indeed, for many commonly-used algorithms, a tight sensitivity bound remains open. Therefore, an alternative *decompose-then-compose* framework is widely applied in practice.

Though tight sensitivity is intractable in general, the sensitivity of aggregation (or average) does enjoy a closed form. In a sum of multiple terms, if each of them is determined by an individual sample, the sensitivity is simply the maximal norm of each item being aggregated. There-

§. Hanshen Xiao and Zihang Xiang are co-first authors.

fore, one natural idea is to (approximately) decompose a complicated algorithm into multiple (possibly adaptive) aggregations. To provide concrete privacy analysis, a more powerful *white-box* adversary is assumed who can observe all intermediate outputs of the algorithm rather than the final one only. Thus, it suffices to privately release each aggregation, and the total privacy leakage is then upper bounded using composition. More complicated algorithms with more such artificial decompositions usually result in looser composite privacy analysis. To ensure bounded sensitivity of each step, a popular approach is to truncate or clip each individual item to be aggregated up to some threshold c in l_2 or l_1 norm. Accordingly, the aggregation outputs are perturbed following Gaussian or Laplace mechanisms [8], [12]. For a d-dimensional release, given constant l_2/l_1-norm sensitivity, the scale of required perturbation is $\Theta(\sqrt{d})$, known as the curse of dimensionality [14].

DP Stochastic Gradient Descent (DP-SGD) [15], [16] and many other variants of gradient methods such as the private Frank–Wolfe algorithm [17] all follow the above-described framework. As the most popular method for private learning, DP-SGD enjoys wide applicability without any additional assumptions on either the objective loss function or training data. Essentially, DP-SGD can be viewed as an iterative aggregation of gradients evaluated by subsampled samples. The mechanism works by repeating the following two operations. First, DP-SGD computes and clips each subsampled per-sample gradient evaluated on the latest model state. Then, clipped gradients are privately aggregated and the model state is updated via a noisy gradient descent. Following a compositional reasoning, to ensure an (ϵ, δ)-DP guarantee for a DP-SGD with T iterations, one needs a perturbation in a scale of $\tilde{\Theta}(\sqrt{Td\log\delta}/(n\epsilon))$ [18] in each iteration. This suggests that in the high-dimensional scenario, for example, training an overparameterized ($d \gg n$) neural network, DP noise could be much larger than the original gradient given a limited number of samples.

For convex optimization with a Lipschitz assumption, where the l_2 norm of gradient is globally bounded, DP-SGD is known to produce the optimal privacy-utility tradeoff asymptotically [18], [19]. The generalization of DP-SGD in many other setups such as for heavy-tailed data [20], [21] or distributed optimization [22] (with local DP) are extensively studied. However, for generic non-convex optimization without a Lipschitz assumption, the understanding of practical implementation of DP-SGD is limited. The empirical performance of DP-SGD has been reported in a vast number of learning models and tasks, accompanied by heavy utility loss. For example, the classification model trained for Imagenet via DP-SGD from scratch can only achieve 47.9% accuracy with $(\epsilon = 10, \delta = 10^{-6})$ on ResNet18 [23], while non-privately one can easily achieve above 75% on the same model.

It was believed that such performance gap is mainly because of massive composition [1] and large model dimension, which result in prohibitive noise injection [10], [24]. A previous perception of DP-SGD is that DP noise required in large models will offset the enhanced learning capacity. For example, Tramer and Boneh in [9] argued that within the current framework, for image datasets with less than 500K samples, simple linear models would outperform the Convolutional Neural Network (CNN). To this end, for a given dataset, many existing works are devoted to searching for a small enough model that produces optimal tradeoff [10]. If the above is the case, to close the gap between private and non-private learning, either there is some efficient shallow learning mechanism with competitive performance, or we need a completely different private optimization method from DP-SGD. Unfortunately, both problems are extremely challenging given state-of-the-art in machine learning and DP privatization.

1. Though larger models could possibly generalize better, they also usually require a longer convergence/training time.

1.1. Our Contribution

In this paper, we try to understand the utility loss of DP-SGD and develop a theoretical foundation for systematic improvement. One key argument we want to make is that both the applicable model dimension and the bias caused by gradient clipping in DP-SGD are underestimated in previous works. We show that given only second-moment-bounded stochastic gradient, clipped/normalized gradient descent in general does *not* converge and clipping bias widely occurs in practical learning tasks. On the other hand, if we are given access to less biased clipped gradient, even under current conservative (white-box adversary) privacy analysis, practical deep learning can indeed tolerate much larger noise than expected.

The remaining paper is organized to answer the following two questions in sequence: *why* DP-SGD or clipped-SGD endures utility loss *even without noise perturbation*, and *what and how to clip* to close the gap between non-private and private learning with sensitivity restriction. We first present generic theoretical convergence analysis of DP-SGD, where we show the bias caused by gradient clipping is upper bounded by the *sampling noise of the stochastic gradient* to be clipped. Our theory supports the previous empirical observation [9], [25] that the selection of small clipping threshold with normalization produces good utility-privacy tradeoff. More importantly, the developed theory validated by extensive experiments shows that sampling noise reduction of stochastic gradient is promising: we transform the abstract improvement over DP-SGD to a more concrete variance reduction problem for per-sample gradient. Within the classic framework subject to per-sample gradient clipping, we propose and study various reduction methods from different angles, summarized as follows,

1) From the optimization angle, we propose *inner-outer momentum*: per-sample gradients over past iterates are first exponentially averaged (inner momentum) before clipping, and afterwards, aggregate noisy clipped gradients over past iterates are averaged (outer momentum) before gradient descent.

2) From the perspective of the learning model, we propose several useful tricks to optimize network architecture subject to per-sample gradient computation. In most existing works on deep learning via DP-SGD, to enable per-sample gradient computation, network normalization involving participation of multiple samples, such as BatchNorm layer [26], is simply replaced by GroupNorm layer [9], [27], [28]. However, we show such straightforward replacement may not perfectly adapt the original network architecture and lead to sub-optimal performance. Several tricks to enhance network normalization are proposed, which provide significant sampling noise reduction and accelerate convergence rate. Moreover, with additional access to (a small amount of) public data, we propose *BatchClipping*, a novel method to carefully clip the gradient of a batch of samples while ensuring the same sensitivity guarantee and good generalization, simultaneously.
3) From the data preprocessing angle, we use our theory to examine two recently-proposed improvements, namely, data normalization through scattering network [9] and self augmentation [28].

The rest of the paper is organized as follows. In Section 2.1, we introduce the background and in Section 2.2 we include the empirical measurements on the clipping effect in practical deep learning. The generic convergence analysis of DP-SGD with either gradient normalization or clipping is presented in Section 3, and we study the implications of the theory in Section 3.3. Concrete improvements from three perspectives, optimization, network normalization and data preprocessing are presented in Sections 4, 5 and 6, respectively. Additional experiments combining all those improvements are included in Section 7. Finally we conclude in Section 8.

1.2. Related Works

Theoretical (Clipped) DP-SGD Analysis: In most existing theoretical studies on DP-SGD [15], [18], [19], the objective loss function is assumed to be L-Lipschitz continuous. The L-Lipschitz assumption implies that the l_2 norm of the gradient evaluated on any point is uniformly bounded by L. Thus, in such a setup, DP-SGD is essentially the original SGD with additional perturbation across iterations: gradient clipping is not necessary or the clipping threshold is virtually set to be L. With a Lipschitz assumption, for convex optimization on a dataset of n samples, a training loss of DP-SGD $\tilde{\Theta}(L\sqrt{d\log(1/\delta)}/(\epsilon n))$ is known to be tight under an (ϵ, δ) DP guarantee [18]. However, in practical learning problems, a (tight) Lipschitz bound is intractable or may not even exist. Gradient clipping is then introduced in [16] to artificially ensure a bounded sensitivity.

Compared to other private learning methods such as objective perturbation [29] or *Private Aggregation of Teacher Ensembles* (PATE) [30], [31], which either require strong convexity or massive unlabelled public data, gradient clipping enables DP-SGD to work for most learning problems without any additional assumptions. However, classic results [18], [19] assuming bounded gradients cannot be directly generalized to clipped DP-SGD. Some existing works try to narrow this gap with new analysis. For example, [32] presents a convergence analysis of smooth optimization but requires bounded sampling noise in stochastic gradient. Moreover, a somewhat impractical assumption is made where the clipping threshold c needs to be $\Omega(T)$ where T is the total number of iterations. [33] relaxed the requirement with an assumption that the sampling noise is symmetric. [34] studied the special case of generalized linear models. Recent papers show more advanced and complicated analysis [25], [35]. However, [35] requires assumptions on adaptively bounded sampling noise, while [25] characterizes the convergence rate using an implicit envelope function without a closed form. Though existing works have made many interesting advances with the assistance of various additional assumptions, few neat and intuitive theoretical results are known, which can be later used to systematically instruct improvements. The bottleneck of clipped DP-SGD is still not clear. Moreover, our motivation and results are also very different from those prior works. Instead of figuring out the theoretical and restrictive conditions such as symmetry [33] or generalized smooth assumption [35] that clipped or normalized SGD could converge, in this paper we show that clipping bias exists both in the worst case and in many practical learning tasks, and in general cannot be averaged out across the optimization iterations. Our focus is thus how to reduce such clipping bias in practice.

Heuristic Empirical Improvement: DP-SGD is notorious for its sensitivity to hyper-parameter selection including the learning rate, iteration number and clipping threshold. Recently, [25] proposes to apply normalized gradient to avoid the overhead to optimize the clipping threshold. [23], [28] report that given the same privacy budget, a larger iteration number (composition) with a bigger batchsize usually produces better performance [2]. [37] proposes to spend some privacy budget to adaptively adjust the clipping threshold during optimization. Another line of works view the model dimension as the main obstacle impeding DP-SGD, and set out to optimize the tradeoff between model size and utility loss [9], [10], and report that a carefully-selected small model can outperform a large model for many medium-size datasets. To overcome the curse of dimensionality, when additional public data is available, another idea is to project the private gradient into a low-rank/dimensional subspace approximated by public samples [27], [38], [39], [40], or implement transfer learning [9], [24], [41], where only a small fraction of weights in the network need to be privately fine-tuned.

To our knowledge, most existing works on DP-SGD report the best performance via grid searching on those hyper-parameters for specific datasets. Such adaptive searching and selection incur privacy loss themselves though some theoretical solutions for private parameter tuning [42], [43]

2. Similar phenomena are observed in our simulation and we are inclined to believe this is because more iterations may help average out the DP noise, and from the sampling amplification [14], a larger batchsize indeed produces better signal-to-noise ratio. But, too big a batchsize causes efficiency issues and influences the generalization of local minima found [36].

are known. Therefore, a theory for generic instruction is in demand. Besides, due to the lack of theory, most empirical improvements are still devoted to forcing a known learning algorithm to match the composite privacy analysis and studying assumptions that enable less DP noise.

2. DP-SGD and Deep Learning

2.1. Preliminaries and Notations

Empirical Risk Minimization: To be self-contained, we briefly introduce the background of statistical machine learning from an optimization perspective. In general, the model to be trained is represented by a parameterized function $f(w,x) : (\mathcal{W},\mathcal{X}) \to \mathbb{R}$, mapping feature x from input domain $\mathcal{X}$ into an output (prediction/classification) domain. In the following, we will always use d to represent the dimension of the parameter w, i.e., $w \in \mathbb{R}^d$. For example, one may consider $f(w,x)$ as a neural network with multiple linear layers connected by (non-linear) activation layers $\phi_{[1:L]}$, and $w_{[1:L]}$ represent the weights to be learned. In general, we can represent a neural network in the following function,

$$f(w,x) = \phi_L(...\phi_2(\phi_1(x \cdot w_1) \cdot w_2)...w_L),$$

where w_l is the weight at the l-th layer and ϕ_l represents some (non-linear) activation layer.

To find a model with appropriate weight w to fit the data, in supervised learning, the training process can be described as solving an optimization problem. Given a set $\mathcal{D}$ of n samples $\{(x_i,y_i), i = 1,2,...,n\}$, where x and y represent the feature and label, respectively, we define the problem of Empirical Risk Minimization (ERM) [44] for some loss function $l(\cdot,\cdot)$ as follows,

$$\min_w F(w) = \min_w \frac{1}{n} \cdot \sum_{i=1}^{n} l(f(w,x_i),y_i). \tag{1}$$

For convenience, we simply use $f(w,x_i,y_i)$ to denote the objective loss function $l(f(w,x_i),y_i)$ in the rest of the paper. Below, we formally introduce the definitions of *Lipschitz continuity* and *smoothness*, which are commonly used in optimization research.

Definition 2.1 (Lipschitz Continuity)**.** *A function f is L-Lipschitz if for all $w,w' \in \mathcal{W}$, $|f(w)-f(w')| \le L\|w-w'\|_2$.*

Definition 2.2 (Smoothness)**.** *A function f is β-smooth on $\mathcal{W}$ if $\nabla f(w)$ is β-Lipchitz such that for all $w,w' \in \mathcal{W}$, $\|\nabla f(w) - \nabla f(w')\|_2 \le \beta\|w'-w\|_2$.*

In the following, we will simply use $\|\cdot\|$ to denote the l_2 norm unless specified otherwise.

Differential Privacy (DP): We first formally define (ϵ,δ)-DP and (α,ϵ)-Rényi DP as follows.

Definition 2.3 (Differential Privacy [45])**.** *Given a data universe $\mathcal{X}^*$, we say that two datasets $\mathcal{D},\mathcal{D}' \subseteq \mathcal{X}^*$ are adjacent, denoted as $\mathcal{D} \sim \mathcal{D}'$, if $\mathcal{D} = \mathcal{D}' \cup s$ or $\mathcal{D}' = \mathcal{D} \cup s$ for some additional datapoint s. A randomized algorithm $\mathcal{A}$ is said to be (ϵ,δ)-differentially-private (DP) if for any pair of adjacent datasets $\mathcal{D},\mathcal{D}'$ and any event S in the output space of $\mathcal{A}$, it holds that*

$$\mathbb{P}(\mathcal{A}(\mathcal{D}) \in S) \le e^{\epsilon} \cdot \mathbb{P}(\mathcal{A}(\mathcal{D}') \in S) + \delta.$$

Definition 2.4 (Rényi Differential Privacy [11])**.** *A randomized algorithm $\mathcal{A}$ satisfies (α,ϵ)-Rényi Differential Privacy (RDP), $\alpha > 1$, if for any pair of adjacent datasets $\mathcal{D} \sim \mathcal{D}'$,*

$$\epsilon \ge \mathsf{D}_\alpha(\mathbb{P}_{\mathcal{M}(\mathcal{D})} \| \mathbb{P}_{\mathcal{M}(\mathcal{D}')}).$$

Here, $\mathbb{P}_{\mathcal{M}(\mathcal{D})}$ and $\mathbb{P}_{\mathcal{M}(\mathcal{D}')}$ represents the distributions of $\mathcal{M}(\mathcal{D})$ and $\mathcal{M}(\mathcal{D}')$, respectively, and

$$\mathsf{D}_\alpha(\mathsf{P}\|\mathsf{Q}) = \frac{1}{\alpha-1} \log \int \mathsf{q}(o)(\frac{\mathsf{p}(o)}{\mathsf{q}(o)})^\alpha \, do, \tag{2}$$

represents α-Rényi Divergence between two distributions P and Q whose density functions are p and q, respectively.

In Definition 2.3 and 2.4, if two adjacent datasets $\mathcal{D}$ and $\mathcal{D}'$ are defined in a form that $\mathcal{D}$ can be obtained by arbitrarily replacing an datapoint in $\mathcal{D}'$, then they become the definitions of bounded DP [8], [12] and RDP, respectivaly. In this paper, we adopt the unbounded DP version to match existing DP deep learning works [9], [16], [27] for a fair comparison.

In practice, to ensure meaningful privacy guarantees, ϵ is usually selected as some small one-digit constant and the failure probability δ is $o(1/|\mathcal{D}|) = o(1/n)$. To randomize an algorithm, the most common approaches in DP are Gaussian or Laplace Mechanisms [14], where a Gaussian or Laplace noise proportional to the sensitivity is added to perturb the algorithm's output. In many applications, for example, using the *decompose-then-compose* privatization framework that includes DP-SGD, we need to quantify the cumulative privacy loss across sequential queries of some differentially-private mechanism on one dataset. The following theorems provide an upper bound on the overall privacy leakage.

Theorem 2.1 (Advanced Composition [46])**.** *For any $\epsilon > 0$ and $\delta \in (0,1)$, the class of (ϵ,δ)-differentially-private mechanisms satisfies $(\tilde{\epsilon},T\delta+\tilde{\delta})$-differential privacy under T-fold adaptive composition for any $\tilde{\epsilon}$ and $\tilde{\delta}$ such that*

$$\tilde{\epsilon} = \sqrt{2T\log(1/\tilde{\delta})} \cdot \epsilon + T\epsilon(e^\epsilon - 1).$$

Theorem 2.2 (Advanced Composition via RDP [11])**.** *For any $\alpha > 1$ and $\epsilon > 0$, the class of (α,ϵ)-RDP mechanisms satisfies $(\tilde{\epsilon},\tilde{\delta})$-differential privacy under T-fold adaptive composition for any $\tilde{\epsilon}$ and $\tilde{\delta}$ such that*

$$\tilde{\epsilon} = T\epsilon - \log(\tilde{\delta})/(\alpha-1).$$

Theorem 2.1 shows a good characterization on how the privacy loss accumulates with composition. For small (ϵ,δ), asymptotically we have an $\tilde{O}(\sqrt{T}\epsilon, T\delta)$ DP guarantee after a T composition. In practice using RDP, Theorem 2.2 usually produces tighter constants in the privacy bound.

DP-SGD: (Stochastic) Gradient Descent ((S)GD) is a very popular optimization approach. Consider an ERM problem to minimize some function $F(w) = \frac{1}{n}\sum_{i=1}^n f(w,x_i,y_i)$. SGD can be described as the following iterative mechanism. In

the k-th iteration, a Poisson sampling is implemented where each datapoint is i.i.d. sampled by a constant rate q [3] , and a minibatch of B_k samples is produced from the dataset $\mathcal{D}$, denoted as S_k. In the following, we assume $q \geq 1/n$. We calculate the stochastic gradient as

$$G_k \leftarrow \sum_{(x_i,y_i)\in S_k} \nabla f(w_{k-1}, x_i, y_i). \tag{3}$$

Then, a gradient descent update is

$$w_k = w_{k-1} - \eta \cdot G_k, \tag{4}$$

for some stepsize η. Without subsampling (or $q = 1$) but applying the full batch of samples, (4) becomes GD. In the following, we formally define the sampling noise of stochastic gradient evaluated at some point w as follows.

Definition 2.5 (Sampling Noise of Stochastic Gradient)**.** *For the given dataset $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$ and the loss function $F(w) = \frac{1}{n} \cdot \sum_{i=1}^{n} f(w, x_i, y_i)$, let (x, y) be a sample randomly selected from $\mathcal{D}$, then the sampling noise of the stochastic gradient at w is defined as*

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}[\|\nabla F(w) - \nabla f(w, x, y)\|^2].$$

The modification from GD/SGD to the corresponding DP version is straightforward. When the loss function f is assumed to be L-Lipschitz [15], [18], i.e., $\|\nabla f(w, x_i, y_i)\| \leq L$ for any w, the worst-case sensitivity in Equation (4) is bounded by ηL in each iteration. Thus, SGD can be privatized via iterative perturbation by replacing Equation (4) with the following:

$$w_k = w_{k-1} - \eta \cdot (G_k + \Delta_k), \tag{5}$$

where Δ_k is the noise for the k-th iteration. For example, if we want to use the Gaussian Mechanism to ensure (ϵ, δ)-DP when running T iterations, then Δ_k can be selected to be i.i.d. generated from

$$\Delta_k \leftarrow \mathcal{N}(\mathbf{0}, O(\frac{L^2 T \log(1/\delta)}{\epsilon^2}) \cdot \boldsymbol{I}_d),$$

where $\boldsymbol{I}_d$ represents the $d \times d$ identity matrix.

However, when we do not have the Lipschitz assumption, an alternative is to force a bounded sensitivity through gradient clipping, or normalization as a special clipping. Following the same notations as before, we describe per-sample gradient clipping [16] as follows,

$$G_k \leftarrow \sum_{(x_i,y_i)\in S_k} \mathsf{CP}\big(\nabla f(w_{k-1}, x_i, y_i), c\big). \tag{6}$$

Here, $\mathsf{CP}(\cdot, c)$ represents a clipping function of threshold c,

$$\mathsf{CP}(\nabla f(w, x, y), c) = \nabla f(w, x, y) \cdot \min\{1, \frac{c}{\|\nabla f(w, x, y)\|}\}.$$

With clipping, the l_2 norm of each per-sample gradient is bounded by c. As a special case of clipping to ensure bounded sensitivity, one can also apply per-sample gradient normalization [25] (the gradient norm is scaled to 1) described as follows,

$$G_k \leftarrow \sum_{(x_i,y_i)\in S_k} \frac{\nabla f(w_{k-1}, x_i, y_i)}{\|\nabla f(w_{k-1}, x_i, y_i)\|}. \tag{7}$$

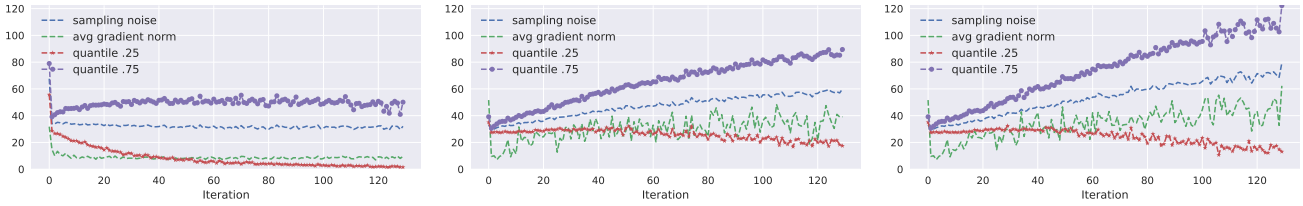
Thus, the clipping threshold c (or normalization to 1) virtually plays the role of the Lipschitz constant L in clipped SGD for *privacy analysis*. However, as we show below, from the utility perspective, we cannot simply use clipping or normalization to force the objective function to be virtually Lipschitz continuous without compromise.

2.2. Observation from Practical Deep Learning

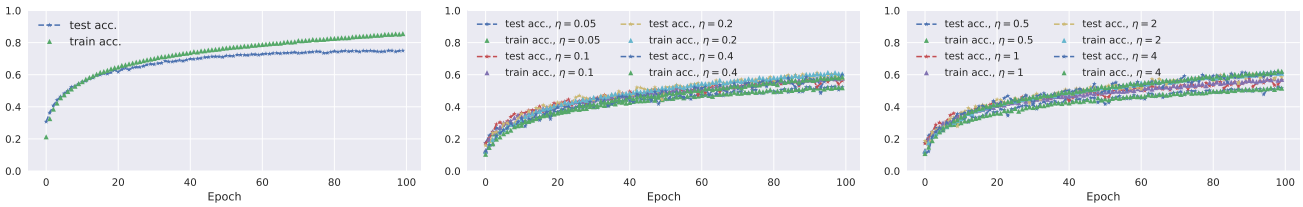
Before we present the formal theoretical analysis, we want to provide some high-level pictures on the effects of clipping in practice. In Fig. 1, we consider training a standard ResNet20 network [47] by only replacing the BatchNorm layer [48] with GroupNorm layer [28] as many prior works did to enable per-sample gradient calculation on CIFAR10, a canonical image set of 50,000 training samples in 10 classes. In each iteration, we apply i.i.d. (Poisson) sampling with $q = 2000/50000$ to generate a batch of samples, where the expected batch size is $2,000$. In Fig. 1a, we include the statistics of the norm of per-sample gradients when we implement standard SGD *without clipping or perturbation*. To be specific, we include the sampling noise (see Definition 2.5), the 25% and 75% quantile of the per-sample gradient's norm, and the norm of the averaged gradient across the batch in each iteration. In Fig. 1d, we keep track of the training and test accuracy. To compare, in Fig. (1b,1e) and (1c, 1f), we implement clipped SGD (*without perturbation*) with clipping thresholds $c = 10$ and $c = 1$, respectively. Note that similar statistical measurements shown in Fig. (1b, 1c) are with respect to the original per-sample gradient *before* the clipping operation. Especially, in Fig. (1e, 1f), we include the performance of clipped SGD with various learning rates. Moreover, in Fig. 2, we further include the test/training accuracy of training ResNet32, a larger model, on CIFAR100, a more challenging classification task, with carefully-selected decaying learning rates across 1000 epochs. We have the following observations when comparing the various cases.

1) **Bias**: The clipping thresholds $c = 10$ and $c = 1$ selected are mostly smaller than the 25% quantile, i.e., most per-sample gradients get clipped during optimization. From both the training and test accuracy, clipping does cause bias in gradient estimation. Such bias in general *cannot be averaged out as iteration count increases* and there is an obvious gap between the performance of standard SGD and clipped SGD. This is more obvious from Fig. 2, where regular SGD achieves 100% training accuracy within 250 epochs, while clipped SGD ($c = 1$) can only achieve 76.2% even after 1000 epochs. However, for those small clipping thresholds where most per-sample gradients get clipped, there is no big difference between the performance of the ultimate models trained out. In

3. We adopt i.i.d. sampling mainly for privacy analysis purposes, since its privacy amplification is easier to analyze through RDP [45].



(a) Gradient Statistics, Regular SGD (b) Gradient Statistics, Clipped SGD ($c = 10$) (c) Gradient Statistics, Clipped SGD ($c = 1$)



(d) Accuracy, Regular SGD (e) Accuracy, Clipped SGD ($c = 10$) (f) Accuracy, Clipped SGD ($c = 1$)

Figure 1: Measurements on per-sample gradient's norm statistics (a, b, c) and model's accuracy (d, e, f). For the first row, the vertical axis is the l_2-norm and measurements are taken at each iteration. We down-sampled (with ratio $1/20$) the data points along the (iteration axis) horizontal axis to have clearer presentation. We choose $batchsize = 2000$ for all cases; For regular SGD, we report the best performance among learning rate η choices $\{0.005, 0.01, 0.02, 0.04\}$; For clipped SGD, we report all performance curves for each learning rate we tested.

other words, the effect of clipping bias is almost the same when c is small. Thus, intuitively such clipping bias cannot be characterized by the clipping threshold c only and there should be a more involved relationship with the gradient distribution.

2) **Sampling Noise and Polarization**: Compared to the standard SGD, per-sample gradient clipping enlarges the sampling noise of stochastic gradient. Besides, as iteration count increases, the norm of per-sample gradients is distributed in a polarized fashion. The gradients of a part of samples become very small (close to 0), which implies that they already fit current model state. Meanwhile, there are other parts of samples' gradients that remain large, which form the main component of gradient used for descent. In general, we do not want a stark polarization to avoid either overfitting or unstable convergence due to extremely large gradients. Comparing the quantile lines in Fig. 1a, 1b and 1c we observe that clipping intensifies such polarization (25% and 75% quantile lines diverge in the clipped SGD case) compared to standard SGD, and extremely large gradients constantly appear during the training process. This coincides with the unstable performance of DP-SGD reported in previous works. But still similarly to 1), for different clipping thresholds, the statistics of the resultant per-sample gradients are similar.

As a summary, for small clipping threshold c which is empirically necessary to ensure meaningful privacy-utility tradeoff, we cannot simply view clipped SGD as standard SGD on some loss function with an artificial Lipschitz constant. Even without any noise perturbation, gradient clipping brings utility loss that cannot be ignored. The

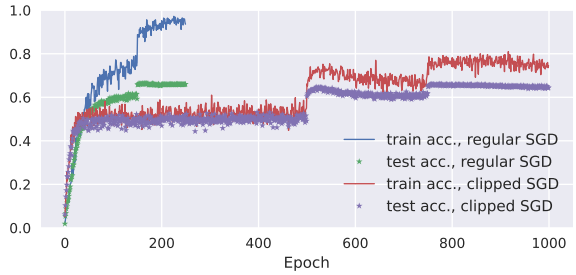


Figure 2: Performance comparison between training ResNet32 with regular SGD and clipped SGD ($c = 1$) on CIFAR100.

distribution of stochastic gradient also dramatically changes, where the norm of per-sample gradient increases on average and polarization intensifies after clipping. On the other hand, comparing different clipping thresholds c with similar performance, the utility loss should be more fundamentally characterized by some other quantities, where smaller c does not necessarily imply heavier compromise on the convergence rate.

3. Theoretical Analysis on Bias in DP-SGD

In this section, generic theoretical analyses of DP-SGD with either a clipping or normalization operation are presented for smooth non-convex optimization. We make minimal assumptions on the stochastic gradient distribution and only assume its second moment is bounded, which is formally stated below.

Assumption 3.1 (Bounded Second Moment of Stochastic Gradient)**.** *For the given dataset $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$ and the loss function $F(w) = \frac{1}{n} \cdot \sum_{i=1}^{n} f(w, x_i, y_i)$, for a randomly selected sample (x, y) from $\mathcal{D}$ and any w, the sampling noise is bounded by τ^2, i.e.,*

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}[\|\nabla F(w) - \nabla f(w, x, y)\|^2] \leq \tau^2.$$

3.1. Normalized DP-SGD

We first consider the per-sample-gradient normalized DP-SGD (Autoclipping [25]), which can be viewed as a special case of clipping described as follows. In the k-th iteration, we i.i.d. sample a minibatch S_k of B_k many samples of rate q, and calculate the gradient in the following way,

$$G_k \leftarrow \sum_{(x_i,y_i)\in S_k} \frac{\nabla f(w_{k-1}, x_i, y_i)}{\|\nabla f(w_{k-1}, x_i, y_i)\|}, \tag{8}$$

where w_{k-1} is the intermediate update from the prior iteration and the norm of each subsampled per-sample gradient is normalized to 1. We then do gradient descent as

$$w_k = w_{k-1} - \eta(G_k + \Delta_k).$$

Here, Δ_k is the zero-mean noise added in the k-th iteration. For simplicity, we uniformly set $\mathbb{E}[\|\frac{\Delta_k}{nq}\|^2] = \sigma^2 d$ for any k. The following theorem captures normalized DP-SGD's utility loss for generic smooth non-convex optimization.

Theorem 3.1. *Under Assumption 3.1, if function $F(\cdot)$ is β-smooth, then for $\eta = O(\frac{1}{nq\sqrt{T}(1+\sigma\sqrt{d})})$, normalized DP-SGD ensures that for $k = 1, 2, ..., T$,*

$$\mathbb{E}[\min_k \|\nabla F(w_{k-1})\|] \leq \sqrt{\frac{32\beta F(w_0)(2+\sigma^2 d)}{T}} + 15\tau. \tag{9}$$

Proof. See Appendix A. □

Our analysis is inspired by the study on normalized momentum [49]. Theorem 3.1 has the following important implications. Provided normalized per-sample gradients, DP-SGD enjoys an $O(\frac{1+\sigma\sqrt{d}}{\sqrt{T}} + \tau)$ convergence rate. The bias is controlled by the sampling noise τ and cannot be cancelled out via a larger iteration number T. This matches our previous observation in Fig. 1. Indeed, such bias characterization is tight, where in general there is no guarantee that normalized per-sample-gradient DP-SGD can exactly converge to the optimum. We may consider the following simple example to optimize the square loss function $F(w) = (w-1)^2 + (w+3)^2$ of two samples $s_1 = 1$ and $s_2 = -3$, where the global minimum is at $w = -1$. Suppose we start with $w = 0$, where the per-sample gradients are -2 and 6, respectively. After normalization, even without any noise perturbation, G_k calculated via (8) is 0, i.e., no update will be performed towards the optimum.

3.2. Clipped DP-SGD

We then proceed to consider the more generic clipped DP-SGD, in which, different from normalizing the per-sample gradients, per-sample gradients are clipped up to some threshold c. Similarly, in the k-th iteration, via i.i.d. sampling of rate q, we get a minibatch S_k of B_k samples and calculate the gradient as

$$\begin{aligned} G_k &\leftarrow \sum_{(x_i,y_i)\in S_k} \mathcal{CP}\big(\nabla f(w_{k-1}, x_i, y_i), c\big) \\ &= \sum_{(x_i,y_i)\in S_k} \nabla f(w_{k-1}, x_i, y_i) \cdot \min\{1, \frac{c}{\|f(w_{k-1}, x_i, y_i)\|}\}. \end{aligned} \tag{10}$$

Then, with the same notations, we update $w_k = w_{k-1} - \eta(G_k + \Delta_k)$. In the following, we use $\mathbb{P}_k = \Pr_{(x,y)}(\|\nabla f(w_{k-1}, x, y)\| < c)$ to denote the probability that for a randomly selected sample (x, y) from $\mathcal{D}$, the norm of its gradient at w_{k-1} is smaller than the threshold c at the k-th iteration. As the sensitivity is scaled by c, we assume $\mathbb{E}[\|\frac{\Delta_k}{nq}\|^2] = (c\sigma)^2 d$ in the clipped case.

Theorem 3.2. *Under Assumption 3.1, if function $F(\cdot)$ is β-smooth, for $\eta = O(\frac{1}{nq\sqrt{T}(1+c\sigma\sqrt{d})})$, clipped DP-SGD ensures that for $k = 1, 2, ..., T$,*

$$\begin{aligned} &\mathbb{E}\big[\min_k \big(\mathbb{P}_k \|\nabla F(w_{k-1})\|^2 \\ &\qquad + \big(c\frac{(1-\mathbb{P}_k)}{4} - \sqrt{1-\mathbb{P}_k}\tau\big) \cdot \|\nabla F(w_{k-1})\|\big)\big] \\ &< \sqrt{\frac{2\beta c^2(2+\sigma^2 d)F(w_0)}{T}} + \frac{15c\tau\sum_{k=1}^{T}\mathbb{E}[\sqrt{(1-\mathbb{P}_k)}]}{4T}. \end{aligned} \tag{11}$$

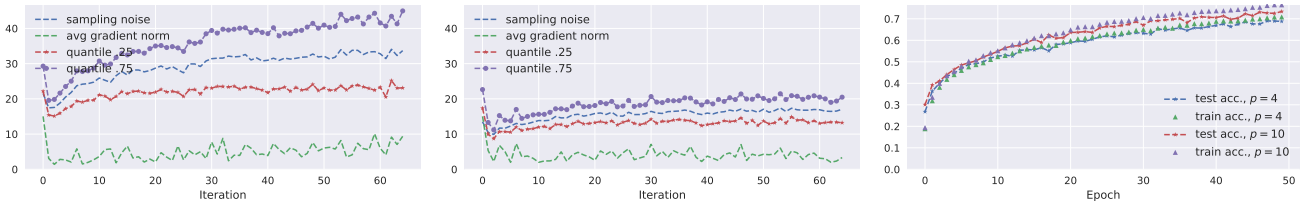
Proof. See Appendix B. □

The utility loss of clipped DP-SGD described in (11) has a very similar form compared to that in Theorem 3.1. In practice for small $c = O(1)$ selected where the chance to get clipped $(1-\mathbb{P}_k) = \Theta(1)$, the bias is dominated by $O(\tau)$, which still cannot be cancelled out as T increases. On the other hand, as c increases, it is noted that $\mathbb{P}_k \to 1$ and the left hand side of (11) approaches $\min_k \mathbb{E}[\|\nabla F(w_{k-1})\|]$, while the right hand side (utility loss) scales up with c. This matches the previous work [9] where via grid searching the optimal privacy-utility tradeoff results in the selection of small c. In the non-private case when $\sigma = 0$ and c is sufficiently large, the clipping bias approaches 0 as $(1-\mathbb{P}_k) \to 0$ and (11) matches the $O(1/\sqrt{T})$ convergence rate of standard SGD.

In the following, we will provide a series of experiments to validate the theory, where the bias characterization via τ is tight, not only asymptotically, but also in practice.

3.3. Implications and p-Averaged Gradient Clipping

In this subsection, we set out to validate our theory and elaborate on its implications to private learning. To improve the sampling noise τ, arguably the most straightforward solution is to clip the averaged gradient of a small group of



(a) Averaged Gradient Statistics, $c = 1$, $p = 4$ (b) Averaged Gradient Statistics, $c = 1$, $p = 10$ (c) Accuracy, Averaged-Clipped SGD

Figure 3: Experiments on SGD with Clipped p-Averaged Gradient.

p ($p > 1$) samples rather than a single one. So far, we only considered per-sample-gradient clipping. If we randomly sample a group S of p samples out of the dataset $\mathcal{D}$ and take the group's empirical gradient mean as a stochastic gradient estimation, the sampling noise decreases as p increases. This idea has also been studied in [50] (Microbatch SGD) with different motivations. To be specific, under Assumption 3.1, we have

$$\mathbb{E}_S[\|\nabla F(w) - \frac{1}{p}\sum_{(x_i,y_i)\in S}\nabla f(w, x_i, y_i)\|^2] \leq \tau^2/p,$$

where the gradient variance is scaled by $O(1/p)$. To this end, if we consider a DP-SGD which clips the averaged gradient of p samples, Theorems 3.1 and 3.2 also apply. The only difference is that in (9) and (11), τ will become $\tau/\sqrt{p}$, where theoretically less bias is incurred as sampling noise decreases.

However, we must stress that from a privacy perspective, such trivial aggregation before clipping *does not* produce a meaningful tradeoff at least in an asymptotic view. Though the clipping bias bounded by τ is improved by a factor of $1/\sqrt{p}$, meanwhile the sensitivity bound (or sampling rate) virtually increases by a larger factor $2p$, since each aggregate gradient to be clipped is computed by p different samples. As a consequence, the scale of DP noise σ needs to also increase by a factor $\Theta(p)$ to compensate for the larger privacy leakage.

Though the trivial p-averaged gradient clipping may not be useful in producing a sharpened utility-privacy tradeoff, we will use this method to validate our theory and show several important implications. In the following, we conduct a series of similar experiments to those in Fig. 1. We still do not take privacy into account. With the same ResNet20 network and image classification task on CIFAR10, we turn to test the clipped SGD which clips the averaged gradient of a group p of samples. In Fig. 3, we consider the selection of $p = 4$ and $p = 10$, respectively, with fixed clipping threshold $c = 1$. In Appendix E Fig. 7, we further include the cosine similarity between the true gradient and p-averaged clipped gradient. We summarize the observations with comparison to Fig. 1 as follows.

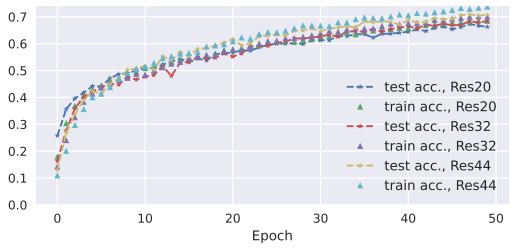
1) **Bias**: A larger p brings more significant improvement with a faster convergence rate. Indeed, when $p = 4$, the performance gap compared to standard SGD (Fig. 1d) is almost closed.
2) **Sampling Noise and Polarization**: When p increases, the sampling noise decreases as expected. Besides, it is worthwhile to note that *the improvement rate of sampling noise is even larger than* $1/\sqrt{p}$. This is because as shown in Fig. 1, compared to the original non-clipping SGD, clipping also enlarges the per-sample gradient's norm on average and amplifies the norm polarization. Thus, the sampling noise upper bound τ in Assumption 3.1 also decreases in practical deep learning with a larger p.

In the following, we proceed to show more fundamental implications of the above results, where the bias caused by clipping is underestimated in previous DP-SGD works. As mentioned before, under standard l_2-norm clipping, DP noise scales as $\Theta(\sqrt{d})$ and a previous perception is that for small/medium datasets, one cannot enjoy the power of a deep neural network where the huge magnitude of noise will offset the strong learning capacity of deep learning. Though the curse of dimensionality does theoretically still exist, we argue that the tipping point of the model size could be underestimated. The key reason behind this is that the clipping bias could dominate the utility loss even under the current conservative noise bound.

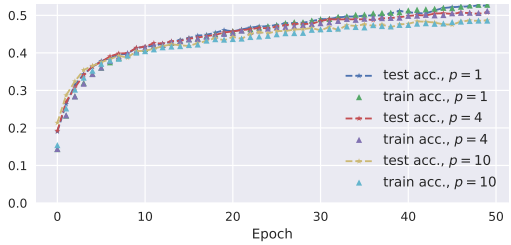
In Fig. 4, we continue the experiments of p-averaged gradient clipping but from a DP-SGD angle. We consider increasing the network size and inject a standard Gaussian noise from $\mathcal{N}(\mathbf{0}, \sigma^2 \cdot \boldsymbol{I}_d)$ in each iteration, where dimension d is the number of parameters of the model selected. We train CIFAR10 over ResNet 20, 32, 44 of $d = 270K$, $d = 463K$, $d = 657K$, respectively, and resultant performance is included in Fig. 4a. A small clipping threshold $c = 1$ and $p = 10$ is selected and fixed in all experiments. The above setup implies that when model size d increases, there is a larger perturbation in a scale of $\Theta(\sigma\sqrt{d})$ with high probability across the iterations, but the norm (power) of gradient maintains as a same constant captured by $c = 1$. Somewhat surprisingly, the performance keeps improving with larger noise in deeper neural networks. However, we have to stress that in such a noise setup, given group number $p = 10$, we do not produce the same DP guarantee compared to standard DP-SGD, as the sensitivity also increases by a factor of 20. The point of this experiment is to show given the same amount of noise *per coordinate*, with less biased clipped gradient, we can exploit much larger networks.

In Fig. 4b, we show the performance of different cases with varying p but assigning a scaled DP noise matching the

selection of p to ensure an identical DP guarantee ($\epsilon = 4, \delta = 10^{-5}$). As analyzed in the beginning of this subsection, from a privacy perspective, simple aggregation does *not* produce an improved utility tradeoff.



(a) Accuracy, (Fake) DP-SGD with 10-Averaged Gradient Clipping, ResNet20, 32, 44



(b) Accuracy, DP-SGD with p-Averaged Gradient Clipping, ResNet20, ($\epsilon = 4, \delta = 10^{-5}$)

Figure 4: Experiments on DP-SGD with Clipped p-Averaged Gradient

To summarize, the implication from the above set of experiments is twofold:

1) In contrast to the previous perception of DP-SGD, the magnitude of DP noise, scaling with model dimension, may not be the main bottleneck in private learning once we have access to less biased gradient. Note that in the above experiments, the norm ratio between the gradient and the noise is fixed to $O(1/\sqrt{d})$. This suggests a promising and fundamental way to improve private deep learning with DP-SGD once the sampling noise in stochastic gradient to be clipped can be reduced.
2) As an expected negative result, simple aggregation to average out sampling noise cannot produce a sharpened utility-privacy tradeoff under current trivial sensitivity bound scaling with p. We will propose a more careful aggregation and clipping method, called *BatchClipping*, in Section 5.2 to address this challenge.

With the above understanding, in the following we proceed to present a series of sampling noise reduction methods *subject to per-sample-gradient clipping*. We want to stress that all proposed improvements do not require new privacy analyses but only require the classic iterative sub-sampling aggregation of standard DP-SGD [45].

4. Optimization Perspective: Inner-Outer Momentum

Though per-sample-gradient clipping is a purely artificial operation for privacy preservation, and, to our knowledge, we have pointed out the relation between its caused bias and sampling noise of stochastic gradient for the first time, variance reduction techniques are widely studied in many other applications with different motivations. In particular for gradients, one of the most popular methods is momentum. Roughly speaking, momentum captures a moving average of past evaluated gradients. We refer interested readers to [49] for more details. The key challenge here is how to adapt momentum into DP-SGD without additional privacy issues and reduce per-sample-gradients' sampling noise.

In Algorithm 1, we present a strawman solution to apply momentum in per-sample-gradient clipping or normalization. For simplicity, we only adopt the normalization method but all the following analysis and algorithms can be generalized to the clipping operation without fundamental difference. Compared to standard DP-SGD, Algorithm 1 is in a DP-GD form, where we keep track of all individual momentum, expressed as an exponential average of gradients of a single sample over past iterates. Instead of clipping per-sample-gradients at the current state, we propose to clip the individual momentum, where our hope is that the sampling noise across different iterations can be averaged out.

Before we state the results, we introduce an additional term $\xi_k^{(i)} = \nabla f(w_{k-1}, x_i, y_i) - \nabla F(w_{k-1})$, which measures the estimation error from the i-th sample's gradient at the k-th iteration.

Definition 4.1 (Exponential Average of sampling noise)**.** *For the i-th sample (x_i, y_i) in $\mathcal{D}$, let $\mathsf{EA}_k^{(i)}$ be the exponential average of sampling noise in its stochastic gradients across the first k iterations, defined as*

$$\mathsf{EA}_k^{(i)} = \mathbb{E}\big[\|\sum_{l=1}^{k}(1-\gamma)^{k-l}\xi_l^{(i)}\|\big],$$

where $\xi_l^{(i)} = \big(\nabla F(w_{l-1}) - \nabla f(w_{l-1}, x_i, y_i)\big)$ represents the sampling noise in the stochastic gradient from the i-th sample in the l-th iteration.

Given Assumption 3.1, we have a simple upper bound on $\sum_{i=1}^n \mathsf{EA}_k^{(i)} \leq \frac{n\tau}{\gamma}$, since for any w, $\sum_{i=1}^n \|\nabla F(w) - \nabla f(w, x_i, y_i)\| \leq n\tau$. On the other hand, suppose the sampling noise in evaluating the stochastic gradients of different w_l and $w_{l'}$, $l \neq l'$, are independent, i.e., $\mathbb{E}\big[(\xi_l^{(i)})^T \xi_{l'}^{(i)}\big] = 0$, then the upper bound of $\sum_{i=1}^n \mathsf{EA}_k^{(i)}$ can be sharpened to $\frac{n\tau}{\sqrt{1-(1-\gamma)^2}} < \frac{n\tau}{\sqrt{\gamma}}$ for $\gamma \in (0,1)$, since

$$\mathsf{EA}_k^{(i)} \leq \sqrt{\mathbb{E}\big[\sum_{l=1}^{k}(1-\gamma)^{2(k-l)}\|\xi_l^{(i)}\|^2\big]}.$$

Theorem 4.1. *Under Assumption 3.1, if the loss function $f(w, x, y)$ is β-smooth with respect to w for any (x, y), and*

let $\eta = O(T^{-3/4}/n)$ *and* $\gamma = O(T^{-1/2})$*, then Algorithm 1 ensures that for* $k = 1, 2, ..., T$*,*

$$\mathbb{E}[\min_k \|\nabla F(w_{k-1})\|] = O\Big(\frac{F(w_0)+\sigma}{T^{1/4}} + \frac{\tau}{\sqrt{T}} + \frac{\sum_{i=1}^n \sum_{k=1}^T \mathsf{EA}_k^{(i)}}{nT^{3/2}}\Big). \quad (12)$$

Proof. See Appendix C. □

Algorithm 1 DP-GD with Momentum

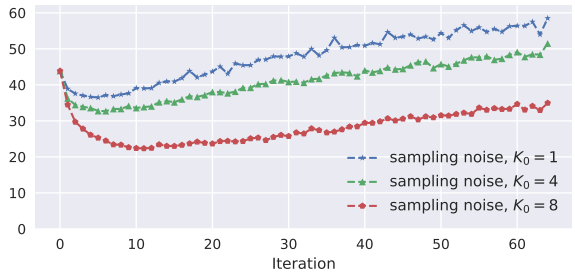
Input: Private dataset $\mathcal{D} = \{(x_{[1:n]}, y_{[1:n]})\}$, loss function $F(w) = \frac{1}{n}\sum_{i=1}^n f(w, x_i, y_i)$, step size η, momentum parameter γ, iteration number T, and noise sequence $\{\Delta_{[1:T]}\}$ with initialization w_0.
Initialize $m_0^{(i)} = \nabla f(w_0, x_i, y_i)$, for $i = 1, 2, ..., n$.
for $k = 1, 2, ..., T$ **do**
  Compute $G_k = \sum_{i=1}^n \frac{m_{k-1}^{(i)}}{\|m_{k-1}^{(i)}\|}$.
  Update $w_k \leftarrow w_{k-1} - \eta(G_k + \Delta_k)$.
  for $i = 1, 2, ..., n$ **do**
    $m_k^{(i)} \leftarrow \gamma \nabla f(w_k, x_i, y_i) + (1-\gamma) m_{k-1}^{(i)}$.
  end for
end for
Output: w_T.

From Theorem 4.1, in the ideal case when the sampling noise $\xi_{[1:T]}^{(i)}$ are independent, $\sum_{i=1}^n \sum_{k=1}^T \mathsf{EA}_k^{(i)} = O(nT^{5/4})$. Thus, the bias produced by clipping is reduced to $O(\tau/T^{1/4})$, which could be cancelled out as T increases. However, in the worst case when $\xi_{[1:T]}^{(i)}$ are highly correlated and identical in different iterations, we then only have the trivial upper bound $\sum_{i=1}^n \sum_{k=1}^T \mathsf{EA}_k^{(i)} = O(nT^{3/2})$. In this case, momentum clipping cannot provide us better convergence rate since the bias is still $O(\tau)$.

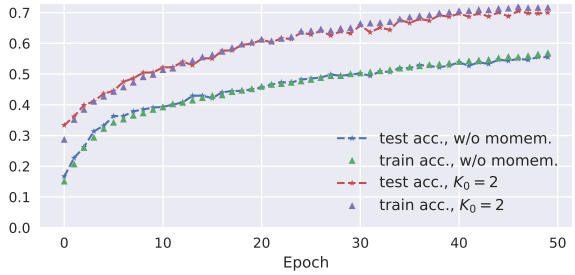
In Fig. 5a, we again take CIFAR10 as an example and measure the correlation amongst the sampling noise $\xi_l^{(i)}$ for adjacent iterations. To be specific, we put a sliding window of length K_0 on the sequence $\xi_{[1:T]}^{(i)}$ and report the average of $\|\frac{1}{K_0} \cdot \sum_{l=k}^{k+K_0-1} \xi_l^{(i)}\|$ over $i = 1, 2, ..., n$. From Fig. 5a, we see that as K_0 increases, the average of per-sample sampling noise gets smaller. Though $\xi_{[1:T]}^{(i)}$ are not fully independent, parts of the sampling noises can be cancelled out in the individual momentum computation.

The above observation provides preliminary verification to apply per-sample gradient momentum to reduce clipping/normalization bias. However, Algorithm 1 is computationally intensive since it is essentially a full-batch gradient descent and we need $O(n)$ memory to store all per-sample momentum. To this end, we present an efficient version of DP-SGD with inner and outer momentum.

In Algorithm 2, to improve both space and computation complexity, we split the momentum computation into two parts. First, during optimization we only need to store the K_0 latest iterates. In each iteration, we apply i.i.d. sampling to generate a batch of samples and evaluate their gradients over the past K_0 iterations, respectively. Per-sample inner



(a) Independence Measurement amongst Per-sample Noise $\xi_l^{(i)}$



(b) Performance with/out Momentum

Figure 5: Sampling Noise and Inner-Outer Momentum

Algorithm 2 DP-SGD with Inner and Outer Momentum

Input: Private dataset $\mathcal{D} = \{(x_{[1:n]}, y_{[1:n]})\}$, loss function $F(w) = \frac{1}{n}\sum_{i=1}^n f(w, x_i, y_i)$, sampling rate q, step size η, inner momentum parameter γ_0, outer momentum parameter γ_1, momentum length K_0, iteration number T, and noise sequence $\{\Delta_{[1:T]}\}$ with initialization w_0.
Initialize outer momentum $M_0 = 0$.
for $k = 1, 2, ..., T$ **do**
  Implement i.i.d. sampling to generate a sample batch S_k of size B_k.
  For each sample (x_i, y_i) selected in S_k, compute inner per-sample momentum

$$m_{k-1}^{(i)} = \sum_{l=k-1-K_0}^{k-1} \gamma_0^{k-1-l} \nabla f(w_l, x_i, y_i). \quad (13)$$

  Compute $G_k = \sum_{(x_i,y_i)\in S_k} \frac{m_{k-1}^{(i)}}{\|m_{k-1}^{(i)}\|}$.
  Compute outer momentum

$$M_k = (1-\gamma_1) M_{k-1} + G_k + \Delta_k. \quad (14)$$

  Update $w_k \leftarrow w_{k-1} - \eta M_k$.
end for
Output: w_T.

momentum $m_k^{(i)}$, defined in (13), is an exponential average of the subsampled i-th sample's gradients over past K_0 iterations of rate γ_0. Afterwards, we normalize and aggregate $m_k^{(i)}$ as G_k. Then, we privately release the outer momentum M_k, defined in (14), as an exponential average of M_{k-1} and G_k. Since M_{k-1} is previously privately published from the last iteration and post-processing does not incur additional

privacy loss, we only need noise Δ_k to ensure sufficient privacy guarantee upon releasing G_k whose sensitivity is 1. One can simply determine the noise parameter as standard DP-SGD using Rényi divergence [45].

In Fig. 5b, we include the performance of Algorithm 2 with comparisons to the standard clipped-SGD without momentum. In particular for Algorithm 2, we select $K_0 = 2$, $\gamma_0 = 0.3$ and $\gamma_1 = 0.6$. In the same setup, inner-outer momentum strengthens the performance.

Before concluding this section, we discuss a bit more about some other possible directions for variance reduction at the optimization level. Besides momentum, another idea is to apply some filter to rule out some undesired samples or adaptively reweight the loss function according to certain metric. It is noted that in the standard ERM we adopt, the weight is uniformly set to be $1/n$ in (1). Importance sampling [51], [52], [53] is a well-known approach to adaptively adjust the weight selection depending on the per-sample-gradient norm. However, we need to stress that similar to classic momentum, importance sampling helps the variance reduction of the *aggregate non-clipped gradient* rather than the per-sample gradient to be clipped. To efficiently incorporate those techniques into DP-SGD, in general we need further understanding on its relationship to clipped gradient.

5. Network Architecture Perspective: Normalization Layer

In Section 4, we improve the DP-SGD itself subject to per-sample-gradient clipping, where we take the learning model (or the loss function to optimize) as given. In this section, we present several novel modifications at the model level to enhance the internal normalization such that the clipping operator on the gradient of those objective functions incurs less bias and meanwhile strong generalization is preserved.

5.1. Normalization Enhancement

Different selection of the learning model will lead to different ERM objectives to optimize, which then consequently influence the noise of stochastic gradient and the clipping bias. Over the last decade, after extensive experiments on massive data, many popular architectures such as CNN [47], Recurrent Neural Network (RNN) [54], and Transformer [55] have emerged to handle different tasks. However, behind their enormous success, deep neural networks are a complicated system requiring very careful coordination of hidden layers, otherwise training becomes very difficult [56]. To serve this goal, many network normalization architectures such as *Batch Normalization* (BN) [48], *Group Normalization* (GN) [57], and *Layer Normalization* (LN) [58], are proposed to stabilize the training process. In particular for our clipping bias control, we observe that network normalization also plays a key role, whereas it was largely overlooked in previous works. In the following, we will study their adaptation to per-sample gradient computation as detailed below.

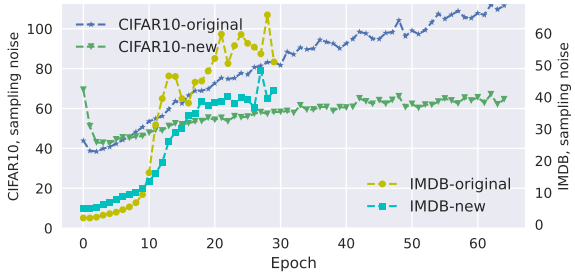
In Section 3.3, we show that gradient averaging before clipping can significantly improve the bias, though this may not be an efficient privacy-utility tradeoff. If we do not take privacy issues into account, given access to a batch of samples S, indeed there exist more efficient approaches to reduce the sampling noise from the learning model aspect. Recall that so far we adopt the ERM objective which is a sum of loss measured on each sample, i.e., $\sum_{i=1}^{n} f(w, x_i, y_i)$. We may consider more generic loss functions measured by a group of samples, say $f(w, S)$. The loss function $f(w, S)$ internally can further implement normalization over samples in S, which may produce more efficient noise variance control compared to simply averaging at the end. In computer vision tasks, one popular architecture which supports this goal is BN.

A comprehensive understanding of BN and its effects in deep learning is still under discussion [26]. Here, we mainly focus on its application for clipping bias reduction. To be self-contained, we formally describe the BN function as Algorithm 4 in Appendix D. At a high level, a network $\mathcal{NW}$ with BN is trained as follows. Given a sample set $S = \{(x_1, y_1), ..., (x_B, y_B)\}$ of B samples where the B features $\boldsymbol{x} = \{x_1, x_2, ..., x_B\}$ are the input of $\mathcal{NW}(w, \cdot)$, the output $\mathcal{NW}(w, \boldsymbol{x})$ is a B-dimensional vector as the prediction of the corresponding B labels $\boldsymbol{y} = (y_1, y_2, ..., y_B)$. The objective function to optimize is defined as $l(\mathcal{NW}(w, \boldsymbol{x}), \boldsymbol{y})$ for some loss function $l(\cdot, \cdot)$. Recall the structure of the neural network described in Section 2.1, when the batch of features $\boldsymbol{x}$ pass through a layer, we obtain B outputs, which then become the input to the next layer. The job of BN is to normalize those outputs from previous layer and then linearly transform them uniformly before forwarding them to the next layer. In Fig. 8 (Appendix E), we continue the experiments on SGD with clipped p-averaged gradient in Section 3.3. But instead of simple averaging, we use the p-group samples to compute the gradient of ResNet20 with BN. Compared to Fig. 3, the sampling noise with p-group BN is only around 50% of that from p-averaging, and the convergence rate is also boosted, where the accuracy on average is improved by around 0.05 in absolute value in the same setup.

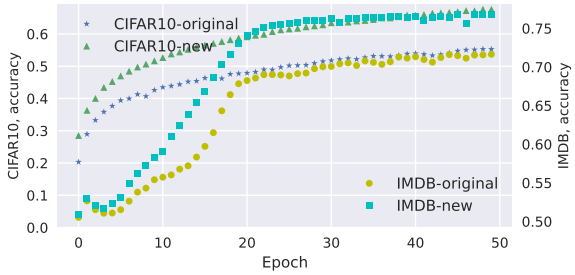
However, it is noted that for a neural network with BN, we can no longer derive the per-sample gradient from the loss function $l(\mathcal{NW}(w, \boldsymbol{x}), \boldsymbol{y})$ anymore. BN introduces inter-dependence amongst samples $\boldsymbol{x}$ in the computation of $\mathcal{NW}(w, \boldsymbol{x})$. To artificially enable per-sample gradient computation, most existing DP-SGD works adopt GN instead of BN. Different from normalizing along the dimension of samples, in CNNs, GN divides convolution channels into multiple groups and implements normalization within each group [57]. We have to stress that in many popular CNNs, such as ResNet, BN is incompatible with GN. Besides, in the non-private regime without restriction on either per-sample gradient computation or batchsize, BN is the more common selection producing better performance in general. Therefore, most existing *CNN architectures are optimized for the setup with a BN rather than a GN layer*, and we find that mere replacement of BN with GN in those networks leads to sub-optimal performance with a breakdown of many original design intuitions. In the following, we summarize the new architectures optimized for GN in CNN, and new

normalization techniques for RNN for NLP applications. Illustration and implementation can be found in our code.

1) In ResNet network [47], instead of maintaining the original layer structure and the non-linear activation function, after each convolutional layer, we propose to place the GN layer after non-linear activation. We also modify the non-linear activation function from Rectified Linear Unit (ReLU) to Exponential Linear Unit (ELU). With a similar idea, we also adopt *Weight Standardization* [59] to further force normalization on learnable parameters of CNN layers before convolution action.
2) With the same principle to normalize per-sample's activation in the forward pass, for RNN (used for NLP tasks) where GN is not applicable, we enforce normalization on the coordinates of each step's hidden state vector. We also find that such normalization action leads to better performance when it is applied to the input of word embedding at each time step.



(a) Per-sample Gradient Sampling Noise



(b) Convergence Rate of Clipped SGD with $c = 1$

Figure 6: Enhanced Network Normalization

In Fig. 6, we include the sampling noise of per-sample gradient and performance when running clipped-SGD ($c = 1$) in modified ResNet20 and the RNN structure proposed in [60] on CIFAR10 and the Internet Movie Database (IMDb) dataset (details can be found in Section 7), respectively, with comparisons to those in the original structures. From Fig. 6(a), the sampling noise in new architectures are close to one half of that in the original ones, and meanwhile the convergence rate, shown in Fig. 6(b) is significantly accelerated in the same setup.

5.2. BatchClipping with Public Data

As mentioned earlier, though BN is not applicable to per-sample gradient computation, as a milestone technique in non-private deep learning, it is in general a much more powerful normalization tool compared to GN. To enable gradient computation with BN while ensuring efficient sensitivity control, one simple idea is to apply public data. Similar to the p-averaged gradient clipping mentioned in Section 3.3, we can mix a single private sample with other $(p-1)$ public data samples, and afterwards compute and clip the gradient from the group. It is not hard to see that such aggregation does not change the sensitivity bound compared to that in the standard per-sample gradient clipping.

However, the key challenge in the above approach is how to mitigate over-fitting if we are only given a very limited number, say a hundred, of public data samples $\mathcal{D}_{pub}$. As the small amount of public data $\mathcal{D}_{pub}$ will be heavily involved in each gradient computation across iterations, if we follow a standard training mechanism to take the whole $\mathcal{D}_{pub}$ as a part of training data, the trained-out model will have tremendous over-fitting on $\mathcal{D}_{pub}$. In the following we present an elegant method termed *BatchClipping* in Algorithm 3 to address this problem. We will use $\boldsymbol{x}_{pub}$ as the feature set of $\mathcal{D}_{pub}$.

Algorithm 3 DP-SGD with BatchClipping

1: **Input:** A neural network with BN layer $\mathcal{NW}(w, \cdot)$, loss function $l(\cdot, \cdot)$, a private dataset $\mathcal{D} = \{(x_{[1:n]}, y_{[1:n]})\}$ and a public feature set $\boldsymbol{x}_{pub}$, step size η, total number of iterations T, sampling rate q, initialization weight w_0 and noise sequence $\Delta_{[1:T]}$.
2: **for** $k = 1, 2, ..., T$ **do**
3: Implement i.i.d. sampling to generate a sample batch S_k of size B_k.
4: For each sample (x_i, y_i) selected in S_k, define a loss function

$$f(w, x_i, \boldsymbol{x}_{pub}, y_i) = l(\mathcal{NW}(w, x_i, \boldsymbol{x}_{pub})[1], y_i), \quad (15)$$

where $\mathcal{NW}(w, x_i, \boldsymbol{x}_{pub})[1]$ is the first coordinate of the prediction output vector of $\mathcal{NW}(w, x_i, \boldsymbol{x}_{pub})$. Calculate the gradient $g_{k-1}^{(i)} = \nabla f(w_{k-1}, x_i, \boldsymbol{x}_{pub}, y_i)$.
5: Calculate $G_k = \sum_{(x_i, y_i) \in S_k} \frac{g_{k-1}^{(i)}}{\|g_{k-1}^{(i)}\|}$.
6: Update $w_k = w_{k-1} - \eta(G_k + \Delta_k)$.
7: **end for**
8: **Output**: w_T

In each iteration of Algorithm 3, the per-sample gradient to be clipped/normalized is computed with the assistance of public feature $\boldsymbol{x}_{pub}$. However, the loss function selected is different from the classic $l(\mathcal{NW}(w, \boldsymbol{x}), \boldsymbol{y})$ described before. In (15), we *do not* use any labels of the public data $\mathcal{D}_{pub}$, while instead we only measure the loss of the prediction on the private sample's label y_i. In other words, we only utilize the features of public data to compute the neural network function $\mathcal{NW}(w, \cdot)$ with a BN layer rather than really training on $\mathcal{D}_{pub}$. This is key to BatchClipping avoiding over-fitting. On the other hand, when we apply the trained-out model $\mathcal{NW}(w_T, \cdot)$ for prediction on some test data x_{test},

the inference computation also requires the public feature $\boldsymbol{x}_{pub}$ and we similarly take $\mathcal{NW}(w_T, x_{test}, \boldsymbol{x}_{pub})[1]$ as our prediction result.

In Fig. 9 (Appendix E), we conduct similar experiments on training ResNet20 on CIFAR10 but assume 100 public features $\boldsymbol{x}_{pub}$ for BatchClipping. Compared to Fig. 6, the sampling noise is further reduced with accelerated convergence rate. Interestingly, with a comparison to Fig. 8 which implements standard training over BN with a group of samples, the BatchClipping (without complete training) still achieves competitive performance. However, to have a fair comparison with existing DP-SGD works from scratch, we do not include BatchClipping in our final experiments reported in Section 7, and we leave the efficient application of BN in DP-SGD without assistance of any public data as an open problem.

6. Input Preprocessing: Data Normalization and Augmentation

So far, we have studied the bias reduction from the perspectives of optimization and learning model. Another key factor that influences ERM is the training data itself. In this section, we study two recently-proposed data preprocessing methods mainly for computer vision tasks.

In [9], the authors propose to split some privacy budget to first privately train a scattering network [61], which is then used to normalize the data. It is shown that the performance of linear or small CNN models trained over the normalized data afterwards with DP-SGD can be significantly improved.

A different method based on data augmentation is presented in [28], called *self-augmentation*. Data augmentation, which plays an important role in computer vision, represents a large class of methods to improve robustness and reduce memorization (instead of generalization) by generating virtual samples. Those virtual samples are produced by applying random cropping [62], erasing [63] or mixing [64] on the raw data. In DP-SGD, instead of clipping a single gradient, [28] considers applying different data augmentations on each raw datapoint and clipping the average of the gradients evaluated by those self-augmented virtual samples. From a privacy perspective, self-augmentation does not cause additional privacy issues since the averaged gradient to be clipped is still only determined by a single private datapoint.

In Fig. 10 and 11 (Appendix E), we still take CIFAR10 as an example and include the sampling noise of stochastic gradient to be clipped after scattering network normalization and self-augmentation, respectively. We find that both scattering normalization (feature extraction) and careful self augmentation improve the sampling noise accompanied with boosted performances, and our theory can be used to explain those empirical successes.

Before the end of this section, we have several comments on self-augmentation. Though the average of gradients evaluated on virtual samples will reduce the sampling noise, we should also note that the modified loss function defined by the augmented samples becomes harder to train. It is observed from Fig. 11 that the convergence rate with self augmentation in the first several epochs is slower than that without any augmentation. Moreover, the performance with $p = 32$ times augmentation is no better than that of $p = 16$ and $p = 8$. In general, a prerequisite to enjoy the gain of self augmentation is a network with powerful enough learning capacity to handle the augmented data. As reported in Table 1 later, we will see that the performance improvement of self-augmentation in ResNet20 is less than that in Wide-ResNet-40 [28]. As a summary, we believe that the most ideal benefit from self augmentation requires careful model and parameter selection depending on the training data. Furthermore, its generalizations to NLP data is an interesting topic for further work.

7. Further Experiments

In this section, we combine all proposed improvements together and provide further experiments on three benchmark datasets, CIFAR10, [65], SVHN [66], and IMDb [67]. CIFAR10 consists of 60,000 color images, where 50,000 are for training and 10,000 for test. The Street View House Numbers (SVHN) dataset has 73,257 images of real world house digits for training and 26,032 for test. We do not use the extra 600K data provided in SVHN, which would make the problem too easy. IMDb set is a canonical NLP sample set, which contains 50,000 movie reviews with obvious bias (positive or negative). Among them, 25,000 are in the training set, while the remaining 25,000 are in the test set. In all the following experiments, we assume the training set is private, which we apply DP-SGD on, and we report the accuracy of the trained-out model on the corresponding test data. For all the experiments in this section, we select $q = \frac{2000}{n}$, i.e., the expected batchsize is 2000; clipping threshold is set to be $c = 5$; learning rate $\eta = 0.1$; outer-momentum $\gamma_1 = 0.9$ and inner-momentum $\gamma_0 = 0.08$ with length $K_0 = 2$; iteration number $T = \lceil \frac{60}{q} \rceil$, i.e., epoch is set to be 60. All experiments are repeated for 5 times with different random seeds. We include our codes in the GitHub Link [4] with implementation details.

For different privacy budgets, we include the corresponding performance in Table 1. As a comparison, we also test DP-SGD upon the original network architectures without normalization enhancement proposed. In addition, we also replace the inner-outer momentum by a standard outer momentum. The results are included in Table 2. We repeat 5 trials for each privacy budget selection and report the median accuracy number. It is noted that for CIFAR10, with budget $(\epsilon = 8, \delta = 10^{-5})$, we achieve 76.0% on ResNet 20 which outperforms 72.6% in [38] with additional 2,000 public data for low-rank gradient embedding. For IMDb data, with budget $(\epsilon = 4, \delta = 10^{-5})$, we achieve 77.5% which outperforms the best-known 70.2% in recent work [53] via additional importance sampling.

4. https://github.com/zihangxiang/A-Theory-to-Instruct-Differentially-Private-Learning-via-Clipping-Bias-Reduction.git

ϵ ($\delta = 10^{-5}$)	4	5	6	7	8
CIFAR10	$71.5 \pm .3$	$72.1 \pm .3$	$73.0 \pm .1$	$73.8 \pm .2$	$74.5 \pm .1$
CIFAR10, 8 self-aug.	$72.4 \pm .3$	$73.7 \pm .3$	$74.3 \pm .7$	$75.8 \pm .2$	$74.9 \pm .3$
CIFAR10, 16 self-aug.	$72.2 \pm .5$	$73.7 \pm .5$	$74.8 \pm .4$	$75.1 \pm .3$	$76.0 \pm .3$
SVHN	$86.3 \pm .3$	$87.4 \pm .3$	$87.8 \pm .2$	$88.1 \pm .2$	$88.6 \pm .3$
SVHN, 8 self-aug.	$88.2 \pm .2$	$89.1 \pm .1$	$89.5 \pm .2$	$89.3 \pm .2$	$90.1 \pm .3$
SVHN, 16 self-aug.	$88.5 \pm .3$	$89.0 \pm .3$	$89.4 \pm .1$	$89.8 \pm .2$	$90.1 \pm .2$
IMDb	$77.5 \pm .3$	$78.1 \pm .2$	$78.4 \pm .1$	$78.8 \pm .2$	$79.2 \pm .3$

TABLE 1: **Test Accuracy** (%) with Normalization Enhancement and Inner-Outer Momentum

ϵ ($\delta = 10^{-5}$)	4	5	6	7	8
CIFAR10	$51.3 \pm .7$	$54.1 \pm .7$	$59.1 \pm .4$	$61.1 \pm .3$	$61.2 \pm .2$
SVHN	$70.5 \pm .3$	$78.1 \pm .3$	$80.5 \pm .2$	$85.1 \pm .2$	$85.3 \pm .3$
IMDb	$69.6 \pm .4$	$70.3 \pm .2$	$70.4 \pm .1$	$72.6 \pm .3$	$73.5 \pm .9$

TABLE 2: **Test Accuracy** (%) with Original Network Architecture and only Outer Momentum

8. Conclusion and Prospects

In this paper, we develop a theoretical foundation of practical implementation of DP-SGD and initiate a study on clipping bias reduction subject to the artificial sensitivity control. Our theory points out a promising direction for systematic improvement via sampling noise reduction. A meaningful lesson learned from our results is that, rather than using brute-force exploration, we need to adapt learning algorithms to existing DP privatization. This requires a more fundamental understanding with respect to the effects of artificial privacy-preservation operations on state-of-the-art machine learning methods.

We show that the clipping bias is underestimated in previous works, which also heavily influences the robustness of practical deep learning with DP-SGD. As a first step, we present several preliminary methods from different angles to design efficient learning schemes friendly to DP-SGD. Incorporating a more advanced variance reduction method with a broader study on other popular neural network architectures will be a very interesting generalization to further close the gap between private and non-private learning. Beyond gradient clipping, our work may also be of independent interest to the study of truncation/clipping bias in other applications.

Acknowledgement

We would like to thank Jun Wan for very helpful discussion and for reading several drafts of this paper. We also thank the anonymous reviewers for their constructive feedback. Hanshen Xiao was supported in part by DSTA, Singapore, and a MathWorks fellowship.

Di Wang and Zihang Xiang were supported by BAS/1/1689-01-01, URF/1/4663-01-01, FCC/1/1976-49-01, RGC/3/4816-01-01, and REI/1/4811-10-01 of King Abdullah University of Science and Technology (KAUST) and KAUST-SDAIA Center of Excellence in Data Science and Artificial Intelligence.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[3] Pablo Villalobos, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, Anson Ho, and Marius Hobbhahn. Machine learning model sizes and the parameter gap. *arXiv preprint arXiv:2207.02852*, 2022.

[4] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[5] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[6] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in Neural Information Processing Systems*, 32:14774–14784, 2019.

[7] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private sgd? *Advances in Neural Information Processing Systems*, 33:22205–22216, 2020.

[8] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[9] Florian Tramer and Dan Boneh. Differentially private learning needs better features (or much more data). In *International Conference on Learning Representations*, 2020.

[10] Nicolas Papernot, Steve Chien, Shuang Song, Abhradeep Thakurta, and Ulfar Erlingsson. Making the shoe fit: Architectures, initializations, and tuning for learning with privacy. 2019.

[11] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pages 263–275. IEEE, 2017.

[12] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006.

[13] Xiaokui Xiao and Yufei Tao. Output perturbation with query relaxation. *Proceedings of the VLDB Endowment*, 1(1):857–869, 2008.

[14] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[15] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[16] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[17] Kunal Talwar, Abhradeep Guha Thakurta, and Li Zhang. Nearly optimal private lasso. *Advances in Neural Information Processing Systems*, 28, 2015.

[18] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473. IEEE, 2014.

[19] Di Wang, Minwei Ye, and Jinhui Xu. Differentially private empirical risk minimization revisited: Faster and more general. In *Advances in Neural Information Processing Systems*, pages 2722–2731, 2017.

[20] Di Wang, Hanshen Xiao, Srinivas Devadas, and Jinhui Xu. On differentially private stochastic convex optimization with heavy-tailed data. In *International Conference on Machine Learning*, pages 10081–10091. PMLR, 2020.

[21] Lijie Hu, Shuo Ni, Hanshen Xiao, and Di Wang. High dimensional differentially private stochastic optimization with heavy-tailed data. *ACM SIGMOD PODS*, 2021.

[22] Adam Smith, Abhradeep Thakurta, and Jalaj Upadhyay. Is interaction necessary for distributed private learning? In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 58–77. IEEE, 2017.

[23] Alexey Kurakin, Steve Chien, Shuang Song, Roxana Geambasu, Andreas Terzis, and Abhradeep Thakurta. Toward training at imagenet scale with differential privacy. *arXiv preprint arXiv:2201.12328*, 2022.

[24] Yannis Cattan, Christopher A Choquette-Choo, Nicolas Papernot, and Abhradeep Thakurta. Fine-tuning with differential privacy necessitates an additional hyperparameter search. *arXiv preprint arXiv:2210.02156*, 2022.

[25] Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. Automatic clipping: Differentially private deep learning made easier and stronger. *arXiv preprint arXiv:2206.07136*, 2022.

[26] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

[27] Da Yu, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. In *International Conference on Learning Representations*, 2020.

[28] Soham De, Leonard Berrada, Jamie Hayes, Samuel L Smith, and Borja Balle. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650*, 2022.

[29] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[30] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

[31] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

[32] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2019.

[33] Xiangyi Chen, Steven Z Wu, and Mingyi Hong. Understanding gradient clipping in private sgd: A geometric perspective. *Advances in Neural Information Processing Systems*, 33:13773–13782, 2020.

[34] Shuang Song, Om Thakkar, and Abhradeep Thakurta. Characterizing private clipped gradient descent on convex generalized linear problems. *arXiv preprint arXiv:2006.06783*, 2020.

[35] Xiaodong Yang, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. Normalized/clipped sgd with perturbation for differentially private non-convex optimization. *arXiv preprint arXiv:2206.13033*, 2022.

[36] Fengli Gao and Huicai Zhong. Study on the large batch size training of neural networks based on the second order gradient. *arXiv preprint arXiv:2012.08795*, 2020.

[37] Galen Andrew, Om Thakkar, Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems*, 34:17455–17466, 2021.

[38] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. Large scale private learning via low-rank reparametrization. In *International Conference on Machine Learning*, pages 12208–12218. PMLR, 2021.

[39] Yingxue Zhou, Steven Wu, and Arindam Banerjee. Bypassing the ambient dimension: Private sgd with gradient subspace identification. In *International Conference on Learning Representations*, 2020.

[40] Hilal Asi, John Duchi, Alireza Fallah, Omid Javidbakht, and Kunal Talwar. Private adaptive gradient methods for convex optimization. In *International Conference on Machine Learning*, pages 383–392. PMLR, 2021.

[41] Zelun Luo, Daniel J Wu, Ehsan Adeli, and Li Fei-Fei. Scalable differential privacy with sparse network finetuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5059–5068, 2021.

[42] Jingcheng Liu and Kunal Talwar. Private selection from private candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 298–309, 2019.

[43] Nicolas Papernot and Thomas Steinke. Hyperparameter tuning with renyi differential privacy. *arXiv preprint arXiv:2110.03620*, 2021.

[44] Prateek Jain and Abhradeep Thakurta. Differentially private learning with kernels. *Journal of Machine Learning Research*, 2013.

[45] Ilya Mironov, Kunal Talwar, and Li Zhang. R\'enyi differential privacy of the sampled gaussian mechanism. *arXiv preprint arXiv:1908.10530*, 2019.

[46] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[48] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[49] Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex sgd. *Advances in neural information processing systems*, 32, 2019.

[50] H Brendan McMahan, Galen Andrew, Ulfar Erlingsson, Steve Chien, Ilya Mironov, Nicolas Papernot, and Peter Kairouz. A general approach to adding differential privacy to iterative training procedures. *arXiv preprint arXiv:1812.06210*, 2018.

[51] Surya T Tokdar and Robert E Kass. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60, 2010.

[52] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

[53] Jianxin Wei, Ergute Bao, Xiaokui Xiao, and Yin Yang. Dpis: An enhanced mechanism for differentially private sgd with importance sampling. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, page 2885–2899, 2022.

[54] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[56] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[57] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[58] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[59] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Micro-batch training with batch-channel normalization and weight standardization. *arXiv preprint arXiv:1903.10520*, 2019.

[60] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[61] Edouard Oyallon, Sergey Zagoruyko, Gabriel Huang, Nikos Komodakis, Simon Lacoste-Julien, Matthew Blaschko, and Eugene Belilovsky. Scattering networks for hybrid representation learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2208–2221, 2018.

[62] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[63] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020.

[64] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[65] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[66] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[67] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

Appendix A. Proof of Theorem 3.1

Based on the updating rule of DP-SGD with normalized per-sample gradient $w_k = w_{k-1} - \eta(G_k + \Delta_k)$, we use the fact that $F(w)$ is smooth and obtain

$$\begin{aligned} &F(w_k) - F(w_{k-1}) \\ &\leq \langle \nabla F(w_{k-1}), w_k - w_{k-1} \rangle + \frac{\beta \|w_k - w_{k-1}\|^2}{2} \\ &= -\eta \langle \nabla F(w_{k-1}), G_k + \Delta_k \rangle + \frac{\beta \eta^2 \|G_k + \Delta_k\|^2}{2}. \end{aligned} \tag{16}$$

Here, $G_k = \sum_{(x_i,y_i)\in S_k} \frac{\nabla f(w_{k-1},x_i,y_i)}{\|\nabla f(w_{k-1},x_i,y_i)\|}$ is the sum of normalized per-sample gradients as defined in (8), where S_k is the minibatch selected in the k-th iteration by q-i.i.d. sampling.

As for G_k, we use $\bar{G}_k = \mathbb{E}[G_k/(nq)]$, i.e., the expectation of $\mathbb{E}[G_k]$ scaled by $1/(nq)$. We use g_k to denote the stochastic gradient at w_{k-1} for (x, y) randomly selected from the training set $\mathcal{D}$. It is noted that on average qn samples are selected in S_k and thus conditional on w_{k-1}, the expectation $\mathbb{E}[G_k/(nq)]$ equals

$$\bar{G}_k = \mathbb{E}[G_k/(nq)] = \mathbb{E}_{(x,y)}\big[\frac{\nabla f(w_{k-1}, x, y)}{\|\nabla f(w_{k-1}, x, y)\|}\big] = \mathbb{E}[\frac{g_k}{\|g_k\|}].$$

Now, we take the expectation on both sides of (16) conditional on w_{k-1}, and then have

$$\begin{aligned} &\mathbb{E}\big[F(w_k) - F(w_{k-1})\big] \\ &\leq -\eta \mathbb{E}[\langle \nabla F(w_{k-1}), G_k + \Delta_k \rangle] + \mathbb{E}\big[\frac{\beta \eta^2 \|G_k + \Delta_k\|^2}{2}\big] \\ &= -\eta \langle \nabla F(w_{k-1}), \mathbb{E}[G_k] \rangle] + \mathbb{E}\big[\frac{\beta \eta^2 \|G_k + \Delta_k\|^2}{2}\big] \\ &\leq -\eta' \langle \nabla F(w_{k-1}), \bar{G}_k \rangle + \frac{\beta \eta'^2 (2 + \sigma^2 d)}{2}. \end{aligned} \tag{17}$$

In the third line of (17), it is noted that the DP noise is of zero mean, $\mathbb{E}[\Delta_k] = 0$, and we use $\eta' = \eta \cdot (nq)$ to represent the scaled learning rate. In the last inequality of (17), we use the following fact to upper bound $\mathbb{E}\big[\frac{\beta \eta^2 \|G_k + \Delta_k\|^2}{2}\big]$. Let $v_i = \frac{\nabla f(w_{k-1},x_i,y_i)}{\|\nabla f(w_{k-1},x_i,y_i)\|}$ for $i = 1, 2, ..., n$, and $\|v_i\| = 1$. We use n independent Bernoulli variables $\mathbf{1}_{[1:n]}$ of parameter q, i.e., $\Pr(\mathbf{1}_i = 1) = q$, as indicators to show whether the i-th sample is selected. Now, we have that

$$\begin{aligned} \mathbb{E}[\|G_k\|^2] &= \mathbb{E}[\|\sum_i \mathbf{1}_i v_i\|^2] \\ &\leq \sum_{i \neq j} \mathbb{E}[\mathbf{1}_i \cdot \mathbf{1}_j] \|v_i\| \|v_j\| + \sum_{i=1}^{n} \mathbb{E}[\mathbf{1}_i \cdot \mathbf{1}_i] \|v_i\|^2 \\ &= n(n-1)q^2 + qn. \end{aligned} \tag{18}$$

Therefore,

$$\mathbb{E}[\beta \eta^2 \|G_k\|^2] \leq \frac{\beta \eta'^2 \big(n(n-1)q^2 + nq\big)}{(nq)^2} < 2\beta \eta'^2,$$

since $nq \geq 1$ as assumed. Combining that $\mathbb{E}[\|G_k + \Delta_k\|^2] = \mathbb{E}[\|G_k\|^2 + \|\Delta_k\|^2]$, we have the bound claimed.

In the following, we analyze the term $\langle \nabla F(w_{k-1}), \bar{G}_k \rangle$. It is noted that, if we let $\xi_k = g_k - \nabla F(w_{k-1})$ be the sampling noise, then

$$\bar{G}_k = \mathbb{E}_{(x,y)}[\frac{g_k}{\|g_k\|}] = \mathbb{E}_{(x,y)}[\frac{\nabla F(w_{k-1}) + \xi_k}{\|\nabla F(w_{k-1}) + \xi_k\|}].$$

Now, $\langle \nabla F(w_{k-1}), \bar{G}_k \rangle$ can be rewritten as

$$\begin{aligned} \langle \nabla F(w_{k-1}), \bar{G}_k \rangle &= \mathbb{E}[\langle \nabla F(w_{k-1}), \frac{\nabla F(w_{k-1}) + \xi_k}{\|\nabla F(w_{k-1}) + \xi_k\|} \rangle] \\ &= \mathbb{E}[\frac{\|\nabla F(w_{k-1})\|^2 + \langle \nabla F(w_{k-1}), \xi_k \rangle}{\|\nabla F(w_{k-1}) + \xi_k\|}]. \end{aligned} \tag{19}$$

We consider the following two cases. Let $p > 1$ be some parameter to be determined. For any given values of $\|\xi_k\|$,

(1).When $\|\nabla F(w_{k-1})\| > p\|\xi_k\|$, or equivalently, $\|\nabla F(w_{k-1})\|/p > \|\xi_k\|$

$$\begin{aligned}
&\langle \nabla F(w_{k-1}), \frac{\nabla F(w_{k-1})+\xi_k}{\|\nabla F(w_{k-1})+\xi_k\|}\rangle \\
&= \frac{\|\nabla F(w_{k-1})\|^2 + \langle \nabla F(w_{k-1}), \xi_k\rangle}{\|\nabla F(w_{k-1})+\xi_k\|} \\
&> \frac{\|\nabla F(w_{k-1})\|^2}{(1+1/p)\|\nabla F(w_{k-1})\|} - \frac{(1/p)\|\nabla F(w_{k-1})\|^2}{(1-1/p)\|\nabla F(w_{k-1})\|} \\
&= (\frac{p}{p+1} - \frac{1}{p-1})\|\nabla F(w_{k-1})\| \\
&= \frac{p^2-2p-1}{p^2-1}\|\nabla F(w_{k-1})\|.
\end{aligned}$$

(2). When $\|\nabla F(w_{k-1})\| \leq p\|\xi_k\|$, since $\frac{\nabla F(w_{k-1})+\xi_k}{\|\nabla F(w_{k-1})+\xi_k\|} = 1$,

$$\begin{aligned}
&\langle \nabla F(w_{k-1}), \frac{\nabla F(w_{k-1})+\xi_k}{\|\nabla F(w_{k-1})+\xi_k\|}\rangle \geq -\|\nabla F(w_{k-1})\| \\
&= (\frac{p^2-2p-1}{p^2-1} - \frac{p^2-2p-1}{p^2-1} - 1)\|\nabla F(w_{k-1})\| \\
&\geq \frac{p^2-2p-1}{p^2-1}\|\nabla F(w_{k-1})\| - (\frac{p^2-2p-1}{p^2-1}+1)\cdot p\|\xi_k\| \\
&\geq \frac{(p^2-2p-1)\|\nabla F(w_{k-1})\| - 2p(p^2-p-1)\|\xi_k\|}{p^2-1}.
\end{aligned}$$

Putting the two cases together and back to (19), we have that

$$\begin{aligned}
&\langle \nabla F(w_{k-1}), \bar{G}_k\rangle \geq \\
&\quad \frac{p^2-2p-1}{p^2-1}\|\nabla F(w_{k-1})\| - \frac{2p(p^2-p-1)}{p^2-1}\mathbb{E}[\|\xi_k\|].
\end{aligned}$$

Let $p=3$, we have that

$$\langle \nabla F(w_{k-1}), \bar{G}_k\rangle \geq \frac{1}{4}\|\nabla F(w_{k-1})\| - \frac{15}{4}\mathbb{E}[\|\xi_k\|]. \quad (20)$$

Now, put (20) back to (17), and we have

$$\begin{aligned}
\|\nabla F(w_{k-1})\| \leq &\frac{4\mathbb{E}\big[F(w_{k-1})-F(w_k)\big]}{\eta'} + 15\mathbb{E}[\|\xi_k\|] \\
&+ 2\beta\eta'(2+\sigma^2 d).
\end{aligned} \quad (21)$$

Now, summing up both sides of (21) for $k=1,2,...,T$, and we have

$$\begin{aligned}
&\mathbb{E}[\min_k \|\nabla F(w_{k-1})\|] \leq \mathbb{E}[\frac{\sum_{k=1}^T \|\nabla F(w_{k-1})\|}{T}] \\
&\leq \frac{4F(w_0)}{\eta' T} + 2\beta\eta'(2+\sigma^2 d) + 15\frac{\sum_{k=1}^T \mathbb{E}[\|\xi_k\|]}{T} \\
&\leq 2\sqrt{\frac{8F(w_0)\beta(2+\sigma^2 d)}{T}} + 15\tau,
\end{aligned}$$

where we select $\eta' = \sqrt{\frac{4F(w_0)}{2\beta T(2+\sigma^2 d)}}$ and use the fact that the sampling noise $\mathbb{E}[\|\xi_k\|] \leq \tau$ as assumed.

Appendix B. Proof of Theorem 3.2

We adopt the same notations as those used in the proof of Theorem 3.1. Now,

$$G_k = \sum_{(x_i,y_i)\in S_k} \mathcal{CP}\big(\nabla f(w_{k-1}, x_i, y_i), c\big)$$

becomes the sum of clipped per-sample gradients in the minibatch and $\bar{G}_k = \mathbb{E}_{(x,y)}[\mathcal{CP}(g_k, c)]$ is the expectation of G_k scaled by $1/(nq)$. Similarly, by the property of smooth functions and conditional on w_{k-1}, we obtain that

$$\begin{aligned}
&\mathbb{E}\big[F(w_k) - F(w_{k-1})\big] \\
&\leq \mathbb{E}\big[\langle \nabla F(w_{k-1}), w_k - w_{k-1}\rangle + \frac{\beta\|w_k - w_{k-1}\|^2}{2}\big] \\
&= \mathbb{E}\big[-\eta\langle \nabla F(w_{k-1}), G_k + \Delta_k\rangle + \frac{\beta\eta^2\|G_k+\Delta_k\|^2}{2}\big] \\
&\leq -\eta'\langle \nabla F(w_{k-1}), \bar{G}_k\rangle + \underline{\frac{\beta c^2\eta'^2(2+\sigma^2 d)}{2}},
\end{aligned} \quad (22)$$

We still use $g_k = \nabla f(w_{k-1}, x, y)$ to represent the stochastic gradient at w_{k-1} for (x,y) randomly selected. The underlined term in (22) is similarly derived from (18). The only difference is that now the clipped norm $\|v_i\|$ is no larger than c due to clipping rather than fixed to be 1 in the normalization scenario, where $v_i = \mathcal{CP}(\nabla f(w_{k-1}, x_i, y_i), c)$.

In the following, we focus on the term $\langle \nabla F(w_{k-1}), \bar{G}_k\rangle$. We use $\xi_k = g_k - \nabla F(w_{k-1})$ to represent the sampling noise. It is noted that

$$\begin{aligned}
\bar{G}_k &= \mathbb{E}_{(x,y)}[\mathcal{CP}(g_k, c)] = \mathbb{E}_{(x,y)}[g_k \cdot \min\{1, \frac{c}{\|g_k\|}\}] \\
&= \mathbb{E}_{\xi_k}[(\nabla F(w_{k-1}) + \xi_k)\cdot \min\{1, \frac{c}{\|(\nabla F(w_{k-1})+\xi_k)\|}\}].
\end{aligned}$$

Thus, we can rewrite $\langle \nabla F(w_{k-1}), \bar{G}_k\rangle$ as

$$\begin{aligned}
&\langle \nabla F(w_{k-1}), \bar{G}_k\rangle \\
&= \underbrace{\mathbb{E}\big[\mathbf{1}_{\|\nabla F(w_{k-1})+\xi_k\|<c}\cdot\langle \nabla F(w_{k-1}), \nabla F(w_{k-1})+\xi_k\rangle\big]}_{(A)} \\
&+ \underbrace{c\cdot\mathbb{E}\big[\mathbf{1}_{\|\nabla F(w_{k-1})+\xi_k\|\geq c}\cdot\frac{\langle \nabla F(w_{k-1}), \nabla F(w_{k-1})+\xi_k\rangle}{\|\nabla F(w_{k-1})+\xi_k\|}\big]}_{(B)}.
\end{aligned}$$

Let $\mathbb{P}_k = \Pr(\|\nabla F(w_{k-1})+\xi_k\| < c)$. Then, for the term (A), we have that

$$\begin{aligned}
(A) = &\mathbb{P}_k\|\nabla F(w_{k-1})\|^2 \\
&\underbrace{-\mathbb{E}\big[\mathbf{1}_{\|\nabla F(w_{k-1})+\xi_k\|\geq c}\cdot\langle \nabla F(w_{k-1}), \xi_k\rangle\big]}_{(C)}.
\end{aligned}$$

Here, we use the fact that $\mathbb{E}[\xi_k] = 0$ and thus

$$\begin{aligned}
&\mathbb{E}[\mathbf{1}_{\|\nabla F(w_{k-1})+\xi_k\|<c}\cdot\langle \nabla F(w_{k-1}), \xi_k\rangle] \\
&\quad = -\mathbb{E}[\mathbf{1}_{\|\nabla F(w_{k-1})+\xi_k\|\geq c}\cdot\langle \nabla F(w_{k-1}), \xi_k\rangle].
\end{aligned}$$

Now, we focus on term (C). For simplicity, in the following we use $\mathsf{S}_{k,\geq c}$ to denote the set of selections of ξ_k such that

$\|\nabla F(w_{k-1})+\xi_k\| \geq c$, and $\mathsf{P}_{k,z,c} = \Pr(\xi_k \in \mathsf{S}_{k,\geq c}, \|\xi_k\| = z)$ to denote the probability (or density function) that ξ_k is within $\mathsf{S}_{k,\geq c}$ and its norm equals z. Then, we can lower bound term (C) as follows,

$$
\begin{aligned}
&(C)\\
&\geq -\mathbb{E}\big[\mathbf{1}_{\xi_k \in \mathsf{S}_{k,\geq c}} \cdot \|\nabla F(w_{k-1})\| \cdot \|\xi_k\|\big]\\
&= -\|\nabla F(w_{k-1})\| \cdot \int_0^{+\infty} \mathsf{P}_{k,z,c} \cdot z \; dz\\
&= -\|\nabla F(w_{k-1})\| \cdot \int_0^{+\infty} \sqrt{\mathsf{P}_{k,z,c}} \cdot \sqrt{z^2 \cdot \mathsf{P}_{k,z,c}} \; dz\\
&\geq -\|\nabla F(w_{k-1})\| \sqrt{\big(\int_0^{+\infty} \mathsf{P}_{k,z,c} dz\big) \cdot \big(\int_0^{+\infty} z^2 \mathsf{P}_{k,z,c} dz\big)}\\
&\geq -\|\nabla F(w_{k-1})\| \cdot \sqrt{(1-\mathbb{P}_k)\cdot \tau^2}.
\end{aligned} \tag{23}
$$

In the first and the fourth inequality of (23), we use the fact that $\langle a, b\rangle \geq -\|a\| \cdot \|b\|$ and Hölder's inequality, respectively. In the last inequality of (23), it is noted that

$$\int_0^{+\infty} z^2 \mathsf{P}_{k,z,c} dz \leq \mathbb{E}[\|\xi_k\|^2] \leq \tau^2,$$

as assumed in Assumption 3.1. As for the term (B), we may follow the analysis in the proof of Theorem 3.1 and obtain,

$$(B) \geq \mathbb{E}\big[c \cdot \mathbf{1}_{\|\nabla F(w_{k-1}))+\xi_k\| \geq c} \cdot \big(\frac{\|\nabla F(w_{k-1})\|}{4} - \frac{15\|\xi_k\|}{4}\big)\big] \tag{24}$$

$$\geq \mathbb{E}\big[\frac{c(1-\mathbb{P}_k)\|\nabla F(w_{k-1})\|}{4} - \frac{15c\sqrt{(1-\mathbb{P}_k)}\tau}{4}\big]. \tag{25}$$

In (25), we apply tricks similar to those developed in (23) on the term $-\mathbb{E}[\mathbf{1}_{\|\nabla F(w_{k-1})+\xi_k\| \geq c} \cdot \|\xi_k\|]$ again, which can be lower bounded by $-\sqrt{(1-\mathbb{P}_k)}\tau$.

With this preparation, we now go back to (22) with (A,C) and (B), where $\langle \nabla F(w_{k-1}), \bar{G}_k\rangle = (A)+(B) = \mathbb{P}_k\|\nabla F(w_{k-1})\|^2 + (C) + (B)$, and with the fact that $\mathbb{P}_k \leq 1$, we have

$$
\begin{aligned}
&\mathbb{P}_k\|\nabla F(w_{k-1})\|^2 + \big(c(1-\mathbb{P}_k)/4 - \sqrt{1-\mathbb{P}_k}\tau\big)\|\nabla F(w_{k-1})\|\\
&\leq \frac{\mathbb{E}[F(w_{k-1}) - F(w_k)]}{\eta'} + \frac{\beta c^2 \eta'(2+\sigma^2 d)}{2}\\
&\quad + \frac{15c\tau\mathbb{E}[\sqrt{(1-\mathbb{P}_k)}]}{4}.
\end{aligned} \tag{26}
$$

Summing up both sides of (26) from $k = 1, 2, ..., T$, with expectation, we have

$$
\begin{aligned}
&\mathbb{E}\big[\min_k \big(\mathbb{P}_k\|\nabla F(w_{k-1})\|^2\\
&\qquad + \big(c(1-\mathbb{P}_k)/4 - \sqrt{1-\mathbb{P}_k}\tau\big)\cdot\|\nabla F(w_{k-1})\|\big)\big]\\
&\leq \frac{F(w_0)}{\eta' T} + \frac{\beta c^2\eta'(2+\sigma^2 d)}{2} + \frac{\sum_{k=1}^T 15c\tau\mathbb{E}[\sqrt{(1-\mathbb{P}_k)}]}{4T}\\
&= 2\sqrt{\frac{\beta c^2(2+\sigma^2 d)F(w_0)}{2T}} + \frac{15c\tau\sum_{k=1}^T\mathbb{E}[\sqrt{(1-\mathbb{P}_k)}]}{4T},
\end{aligned}
$$

when we select $\eta' = \sqrt{\frac{2F(w_0)}{T\beta c^2(2+\sigma^2 d)}}$.

Appendix C. Proof of Theorem 4.1

We use $\hat{\xi}_k^{(i)} = m_k^{(i)} - \nabla F(w_k)$ to denote the estimation error between the exponentially averaged per-gradient and the true gradient in the k-th iteration in the following. Recall the updating rule (line 7) described in Algorithm 1, we have the following recursion with respect to $\hat{\xi}_k^{(i)}$,

$$
\begin{aligned}
\hat{\xi}_k^{(i)} &= (1-\gamma)m_{k-1}^{(i)} + \gamma\nabla f(w_k, x_i, y_i) - \nabla F(w_k)\\
&= (1-\gamma)(\nabla F(w_{k-1}) + \hat{\xi}_{k-1}^{(i)}) + \gamma\nabla f(w_k, x_i, y_i) - \nabla F(w_k)\\
&= (1-\gamma)\hat{\xi}_{k-1}^{(i)} + \gamma(\nabla f(w_k, x_i, y_i) - \nabla F(w_k))\\
&\quad - (1-\gamma)\big(\nabla F(w_k) - \nabla F(w_{k-1})\big).
\end{aligned} \tag{27}
$$

First, due to the smooth assumption which implies that gradient is β-Lipschitz and the normalized gradient which ensures that $\mathbb{E}[\|w_k - w_{k-1}\|] \leq \eta'(1+\sigma\sqrt{d})$, we have that $\mathbb{E}[\|\nabla F(w_k) - \nabla F(w_{k-1})]\|]$ is upper bounded by $\beta\eta'(1+\sigma\sqrt{d})$. Now, when we unravel the recursion (27) with the union bound, we have that

$$\mathbb{E}[\|\hat{\xi}_k^{(i)}\|] \leq (1-\gamma)^k\|\hat{\xi}_0^{(i)}\| + \gamma\mathsf{EA}_k^{(i)} + \frac{2\beta\eta'(1+\sigma\sqrt{d})}{\gamma}. \tag{28}$$

Therefore, summing up (28) for $i = 1, 2, ..., n$ and $k = 1, 2, ..., T$, since

$$\frac{\sum_{i=1}^n \|\hat{\xi}_0^{(i)}\|}{n} = \frac{\sum_{i=1}^n \|\nabla f(w_0, x_i, y_i) - \nabla F(w_0)\|}{n} \leq \tau,$$

from Assumption 3.1, we obtain

$$
\begin{aligned}
\sum_{i=1}^n\sum_{k=1}^T \frac{\mathbb{E}[\|\hat{\xi}_k^{(i)}\|]}{nT} &\leq \frac{\tau}{\gamma T} + \frac{\gamma\sum_{i=1}^n\sum_{k=1}^T \mathsf{EA}_k^{(i)}}{nT}\\
&\quad + \frac{2\beta\eta'(1+\sigma\sqrt{d})}{\gamma}.
\end{aligned} \tag{29}
$$

Now, we may apply the results (21) in the proof of Theorem 3.1 where we still let $n\eta' = \eta$ and obtain

$$
\begin{aligned}
&\mathbb{E}[\min_k |\nabla F(w_k)\|] \leq \frac{4F(w_0)}{\eta' T} + 2\beta\eta'(2+\sigma^2 d)\\
&\quad + 15\big(\frac{\tau}{\gamma T} + \frac{\gamma\sum_{i=1}^n\sum_{k=1}^T \mathsf{EA}_k^{(i)}}{nT} + \frac{2\beta\eta'(1+\sigma\sqrt{d})}{\gamma}\big).
\end{aligned} \tag{30}
$$

Now, in (30), we select $\eta' = O(T^{-3/4})$ and $\gamma = O(T^{-1/2})$, we have that

$$
\begin{aligned}
&\min_k \mathbb{E}[\|\nabla F(w_k)\|]\\
&= O\big(\frac{F(w_0)+1+\sigma\sqrt{d}}{T^{1/4}} + \frac{\tau}{T^{1/2}} + \frac{\sum_{i=1}^n\sum_{k=1}^T \mathsf{EA}_k^{(i)}}{nT^{3/2}}\big).
\end{aligned}
$$

Appendix D. Batch Norm Layer

Algorithm 4 Batch Norm [26]

Input: Output vectors from previous layer $\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_B$, $\boldsymbol{u}_i \in \mathbb{R}^{d_0}$, evaluated by a minibatch B of samples. Trainable parameters $\boldsymbol{\chi}$ and $\boldsymbol{\theta}$ to be learned.
Calculate the empirical average $\boldsymbol{\mu} \leftarrow \frac{1}{B}\sum_{i=1}^{B} \boldsymbol{u}_i$.
for $j = 1, 2, ..., d_0$ **do**
  Compute coordinate-wise variance

$$\sigma_j^2 = \frac{1}{B} \cdot \sum_{i=1}^{B} (\boldsymbol{u}_i(j) - \boldsymbol{\mu}(j))^2.$$

  for $i = 1, 2, ..., n$ **do**
    Implement coordinate-wise normalization

$$\hat{\boldsymbol{u}}_i(j) \leftarrow \frac{\boldsymbol{u}_i(j) - \boldsymbol{\mu}(j)}{\sigma_j}.$$

  end for
end for
Output: $\boldsymbol{\chi} \circ \hat{\boldsymbol{u}}_i + \boldsymbol{\theta}$, $i = 1, 2, ..., B$.

Appendix E. Additional Experiments

In Fig. 7, we report the cosine similarity between the full gradient and the p-averaged clipped gradient. For two vectors u and v, its cosine similarity is defined as $\frac{\langle u,v \rangle}{\|u\|\|v\|}$. It is observed that when $p \geq 4$, the direction of estimated p-averaged gradient is almost identical to that of the true gradient.

In Fig. 8, we include the results of experiments for SGD with clipped gradient on a group of samples with BN. To be specific, we consider the two cases where we split each batch in each iteration into groups of $p = 4$ and $p = 10$ samples [5], respectively, and then calculate and clip the group gradient of ResNet20 with BN on CIFAR10.

In Fig. 9, we implement BatchClipping (Algorithm 3) on ResNet20 with BN. We randomly select 100 features from CIFAR10's test set as the public feature $\boldsymbol{x}_{pub}$. To have a comparison with the experiments in Fig. 8, in each iteration, for each sub-sampled sample, we randomly assign $p = 3$ and $p = 9$ public features from $\boldsymbol{x}_{pub}$ for the BatchClipping computation. As for the model inference, for each test data, we also randomly select p public features and compute the prediction. We repeat this prediction mechanism 10 times and take the majority as the final prediction result.

In Fig. 10, we adopt the three-layer CNN used in [9], and compare the sampling noise of per-sample gradient and the convergence rate of DP-SGD ($c = 1$) on CIFAR10 preprocessed with or without Scattering Network.

In Fig. 11, we test self-augmentation, where we implement $p = 8, 16, 32$ independent augmentations on each subsampled sample and clip the averaged gradient. We record the norm of averaged gradient, sampling noise, per-sample-augmented gradient norm, and the convergence rate in the standard ResNet20 with normalization enhancement proposed, and a comparison to the case without self-augmentation (i.e., the augmentation number is 1).

5. If the batchsize is not divided by the p, we simply take the remainder samples as the last group.

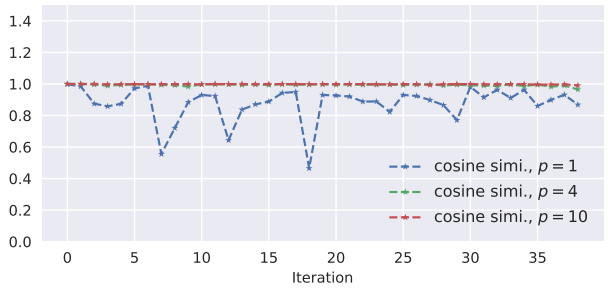
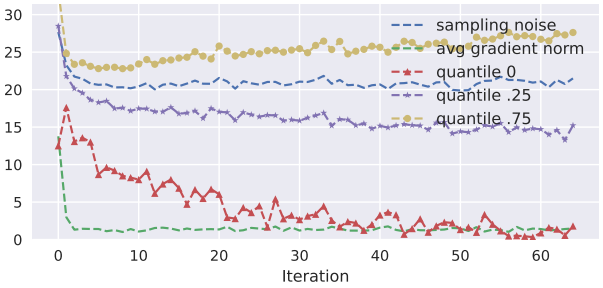
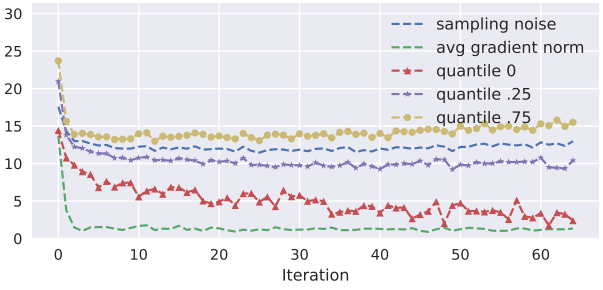


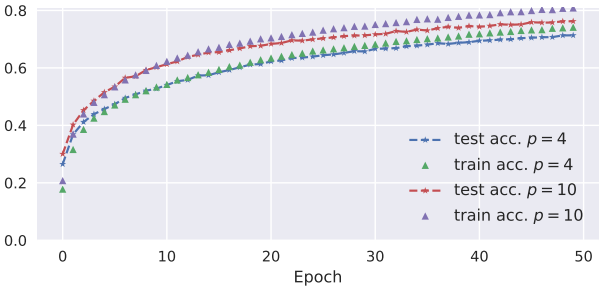
Figure 7: Cosine Similarity between True Gradient and p-averaged Clipped Gradient



(a) Group BN Gradient Statistics, $c = 1$, $p = 4$

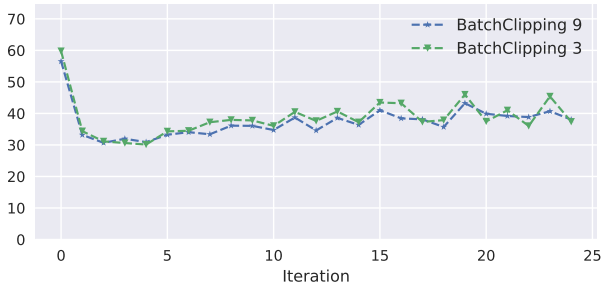


(b) Group BN Gradient Statistics, $c = 1$, $p = 10$

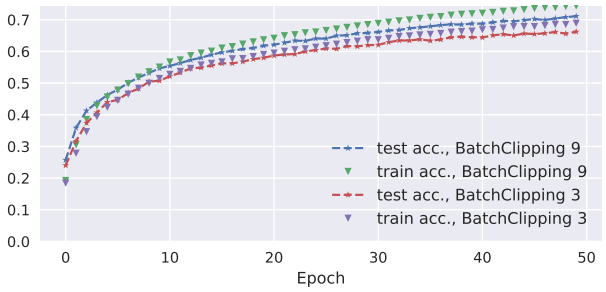


(c) Accuracy, Group-BN-Clipped SGD

Figure 8: Experiments on SGD with Clipped p-Group Gradient with BatchNorm, CIFAR10, ResNet20

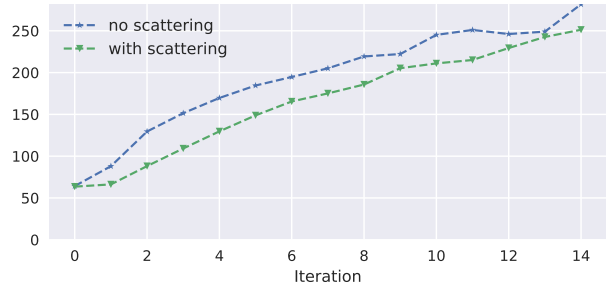


(a) Per-sample Gradient Sampling Noise

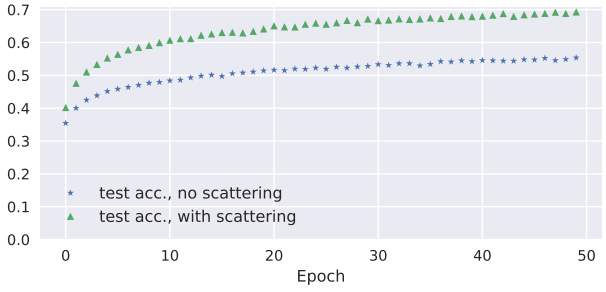


(b) Convergence Rate of Clipped SGD with $c = 1$

Figure 9: Experiments on SGD with BatchClipping with 100 Public Feature, CIFAR10, ResNet20

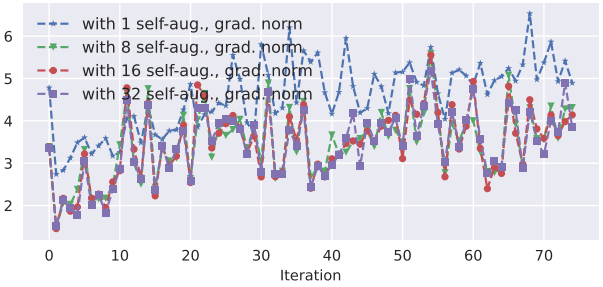


(a) Per-sample Gradient Sampling Noise

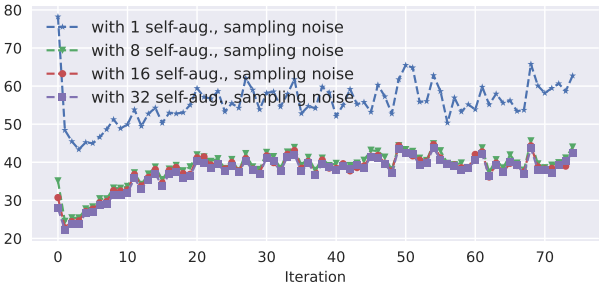


(b) Convergence Rate of Clipped SGD with $c = 1$

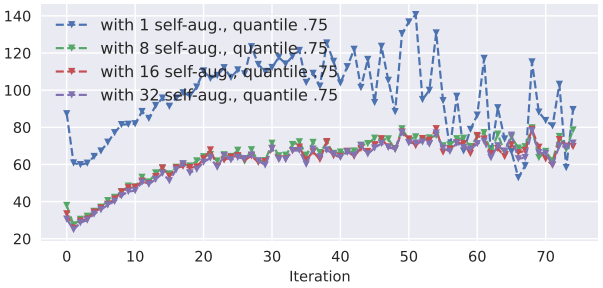
Figure 10: Effect of Scattering Feature Extraction



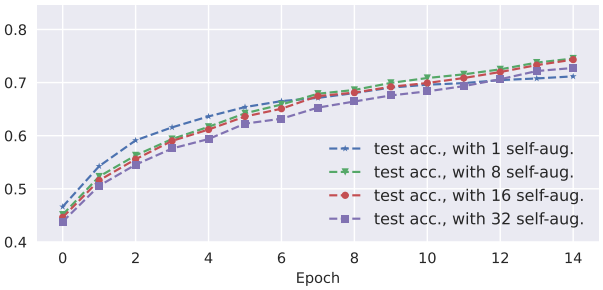
(a) Norm of Averaged Gradient



(b) Per-sample Gradient Sampling Noise



(c) 75% Quantile of Per-sample Gradient Norm



(d) Convergence Rate of Clipped SGD with $c = 1$

Figure 11: Effect of Self-Augmentation