[Paper Review] Scalable and Efficient Training of Large Convolutional Neural Networks with Differential Privacy

Xiao Tian (23R51042)

Tokyo Institute of Technology
Japan 152-8550
tian.x.af@m.titech.ac.jp

Abstract

This report provides a detailed summary and review of the paper [3] stated in the title, which was authored by Zhiqi Bu, Jialin Mao and Shiyun Xu from University of Pennsylvania and published at NeurIPS 2022. In the paper, the authors propose a new technique to train large convolutional neural networks with differential privacy, namely mixed ghost clipping, which significantly improves the scalability and efficiency of the training process.

1 Introduction

Machine learning (ML) may use data that contains private or sensitive information such as income, medical records or browsing history whose owners are not willing to publicly share. However, even if the learner (e.g., companies) does not expose the dataset used to train its ML model, such information can still be leaked from the trained model itself through techniques such as membership inference attacks [11] or model inversion attacks [13]. To protect data privacy, *differential privacy* (DP) [6] has been adopted as a measure of privacy. The core intuition behind DP is that the privacy of a data sample is preserved through a ML model training mechanism if the trained models with or without the sample in the training set are indistinguishable (i.e., the model parameters are similar). A number of such *differentially private machine learning* (DP-ML) mechanisms [1, 7, 8] have been proposed, among which NoisySGD [1] is specifically catered to neural networks and shown to perform well on image classification tasks [5, 12]. The main technique of NoisySGD is *per-sample gradient clipping*, where the gradient computed at each backpropagation step (based on the loss associated with a sample) is normalized and then made noisy so as to mask the actual effect of each sample.

However, the efficiency ($\mathbf{E}$) and scalability ($\mathbf{S}$) of NoisySGD on very large and deep convolutional neural networks (CNNs) remain a problem. Firstly ($\mathbf{E}$), per-sample gradient clipping needs to be done for each sample in a batch, for each weight in the neural network, and at each iteration. Besides, since the gradient at each iteration is noisy, the model takes more iterations to converge. These two factors have significantly slowed down the training process by as many as 1000 times empirically [2]. Secondly ($\mathbf{S}$), it is observed in practice that larger CNN models, trained with NoisySGD, perform worse than smaller ones. Therefore, it is questionable whether training with NoisySGD can be applied to larger-scale CNNs without sacrificing accuracy.

In this paper, the authors have addressed both $\mathbf{E}$ and $\mathbf{S}$ by proposing *mixed ghost clipping*, a mixture of *ghost clipping* and per-sample gradient clipping, whichever is faster. Ghost clipping avoids the need to compute and store the gradient for each sample, thus improving the efficiency ($\mathbf{E}$). The authors also show that by training larger-scale CNNs efficiently with mixed ghost clipping, it is possible to obtain larger-scale CNNs with good accuracy, thus empirically verifying the scalability ($\mathbf{S}$).

2 Backgrounds

2.1 Differential Privacy

Differential privacy (DP) [6] provides a measure of privacy-preserving capability of a ML model training mechanism. Conceptually, it measures how much the trained model parameters differ when each sample is in or is not in the training dataset. The smaller the difference is, the more privacy the training mechanism can preserve. Formally,

Definition 1. [(ϵ, δ)-Differential Privacy] *Two (training) datasets are said to be* ***neighboring*** *if they differ by only one sample. A randomized training mechanism $M : \mathcal{D} \to \mathbb{R}^r$, which takes in a dataset $D \in \mathcal{D}$ to train its r model parameters, is (ϵ, δ)-**differentially private** if for every pair of neighboring datasets D and D' and every subset $O \in \mathbb{R}^r$,*

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

2.2 NoisySGD

NoisySGD [1] is a variant of the stochastic gradient descent (SGD) algorithm to train deep neural networks with DP. Compared with standard SGD, NoisySGD has two additional steps (and thus two special parameters) at each iteration:

- Clipping gradients: The computed gradient is "clipped" with regards to a preset gradient norm bound C in order to limit the impact of each sample (larger impact implies larger risk of privacy breach), i.e., for each sample x_i, we clip the actual gradient $\mathbf{g}(x_i)$ to

$$\bar{\mathbf{g}}(x_i) \leftarrow \frac{\mathbf{g}(x_i)}{\max\left(1, \frac{\|\mathbf{g}(x_i)\|}{C}\right)}, \qquad (2)$$

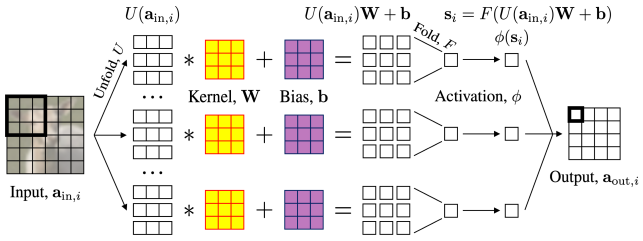

Figure 1: Illustration of a convolutional layer.

where the term $\frac{1}{\max\left(1, \frac{\|\mathbf{g}(x_i)\|}{C}\right)} = C_i$ is called *clipping factor*;

- Injecting noises: Gaussian noises of scale σ are then injected to the clipped gradients in order to mask the impact of each sample, i.e., when taking the average of L gradients in a batch, we do

$$\tilde{\mathbf{g}}(x_i) \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}(x_i) + \mathcal{N}(\mathbf{0}, \sigma^2 C^2 \mathbf{I})\right). \quad (3)$$

Since these two steps have to be done for each sample, the computational cost of NoisySGD is much larger than that of standard SGD.

3 Summary of Methodology

3.1 Ghost Clipping

Consider a convolutional layer with the following forward pass (see Fig. 1 for explanation) of sample x_i:

$$\mathbf{a}_{\text{out},i} \leftarrow \phi(\mathbf{s}_i) = \phi\left(F(U(\mathbf{a}_{\text{in},i})\mathbf{W} + \mathbf{b})\right). \quad (4)$$

By differentiating the loss function $\mathcal{L}_i$ against the kernel weight $\mathbf{W}$, we obtain the following gradient for sample x_i:

$$\mathbf{g}(x_i) = \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}} = F^{-1}\left(\frac{\partial \mathcal{L}}{\partial \mathbf{s}_i}\right)^\top U(\mathbf{a}_{\text{in},i}). \quad (5)$$

The gradient norm, $\|\mathbf{g}(x_i)\|$, can be computed directly without having to compute $\mathbf{g}(x_i)$:

$$\|\mathbf{g}(x_i)\|^2 = \left(U(\mathbf{a}_{\text{in},i})U(\mathbf{a}_{\text{in},i})^\top\right)\left(F^{-1}\left(\frac{\partial \mathcal{L}}{\partial \mathbf{s}_i}\right)F^{-1}\left(\frac{\partial \mathcal{L}}{\partial \mathbf{s}_i}\right)^\top\right). \quad (6)$$

Eq. (6) is the core of ghost clipping. Instead of explicitly computing all the gradients $\mathbf{g}(x_i)$ and storing them in the memory, ghost clipping only computes the gradient **norms** during backward propagation and thus significantly saves the memory space. Fig. 2 gives a comparison between standard per-sample gradient clipping and ghost clipping on how they process Eq. (2) and (3) respectively.

3.2 Mixed Ghost Clipping

As shown in Fig. 2, although ghost clipping significantly saves the memory space, it requires an additional second backward propagation, so it is not necessary that ghost clipping always performs better than per-sample gradient clipping at any layer. To determine which layer to use which clipping technique, the authors give a precise condition based on the layer specifications as follows:

$$\begin{cases} \text{Ghost clipping} & \text{, if } 2H_{\text{out}}^2 W_{\text{out}}^2 < pdk_H k_W; \\ \text{Per-sample clipping} & \text{, otherwise.} \end{cases} \quad (7)$$

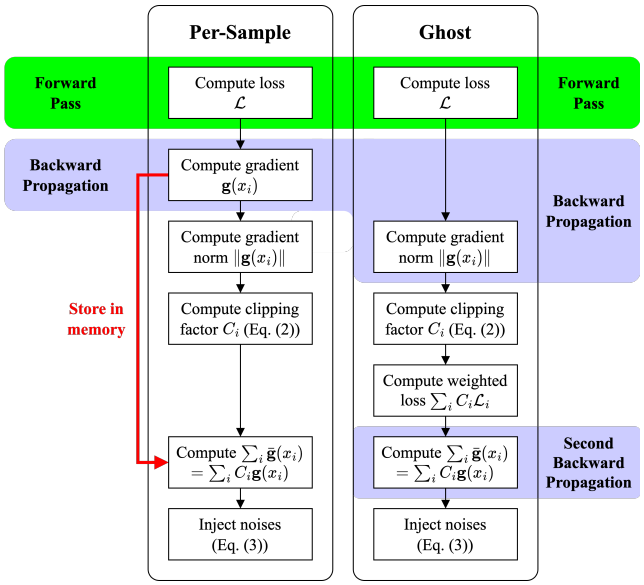

Figure 2: Comparison between per-sample gradient clipping (left) and ghost clipping (right). Per-sample gradient clipping is memory-expensive since it needs to store the gradients, whereas ghost clipping requires extra time for the additional second backward propagation.

Here H_{out} and W_{out} are the output dimension of the layer, p is the number of output channels, d is the number of input channels, and k_H and k_W are the kernel dimension.

Since the dimension of each layer in a large CNN can vary significantly, mixed ghost clipping achieves a much better performance than only per-sample gradient clipping, or only ghost clipping. More specifically, the authors find that ghost clipping is more efficient at the deeper layers (i.e., further away from the input layer) and per-sample gradient clipping is more efficient at the shallower layers.

4 Review

The privacy-utility tradeoff [10] has been a widely known issue of DP-ML. Taking NoisySGD as an example, since we are injecting noises to the gradient computed at each iteration (to "hide" the actual effect of each sample), this will not only slow down the convergence rate but also prevent the model from reaching the minima of loss function in the end. This is because when we are getting closer to the minima, the gradient becomes smaller and eventually comparable to the injected noises. This issue is especially critical to very large and deep neural networks which contain a huge number of parameters and take very long to converge even without DP, highlighted by many previous attempts [2, 9]. However, since data privacy is essential (Sec. 1), research on improving the efficiency (**E**) and scalability (**S**) of current DP-ML techniques for deep neural networks is essential.

This paper proposes a technique, mixed ghost clipping, that clearly improves both **E** and **S**: Theoretically, the authors give a clear and detailed analysis of the time and space complexity of mixed ghost clipping versus previous tech-

niques; empirically, the authors are the first ones who manage to train very large CNN models such as convolutional vision transformers on the standard CIFAR-10/100 datasets and achieve dominating results. This has empirically verified that DP can be applied to large CNNs while maintaining good model performance.

An important finding in this paper is that the choice of DP technique is related to the model architecture and may need to vary across models or even layers within a model. This is because the training of ML models is heavily based on numerical computation and approximation (e.g., matrix computation), whose efficiency and performance depends on the nature of numerical inputs (e.g., dimension, sparsity). This finding has also motivated other works [4] to further improve on such layer-wise hybrid implementations.

However, there are also some potential challenges to the layer-wise hybrid implementations. Firstly, such an implementation requires a selection criterion to choose the appropriate technique for each layer. The selection criterion needs be easy and efficient to implement so as not to undermine the benefits of such implementation. However, it is questionable whether there would always be such a selection criterion as simple as the one in this paper. Secondly, this paper provides one alternative to the original per-sample gradient clipping technique. In future, there will be more alternatives and corresponding selection criteria, thus the selection of suitable technique for each layer will be increasingly difficult.

The technique proposed in this paper is restricted to CNN models as the derivation only works for convolutional layers and fully connected linear layers. Since it employs the fact that convolutional layers are equivalent as linear layers, it is hard to adapt the technique to other deep learning models.

Future work could explore the possibility to extend the concept of layer-wise hybrid implementations to other deep learning models, such as recurrent neural networks (RNN) and transformers. They could also come up with more alternative clipping techniques that are more suitable for certain layer specifications. Furthermore, since this work has proven the practicality of training very large and deep CNNs with DP, more research can be done on maximizing the model performance under certain DP requirements. such research would be highly meaningful these days when deep learning is prevalent and data privacy is drawing much attention.

Acknowledgement

This project serves as the final project of the *SCE.I435 Visual and Knowledge Information Processing* course conducted by Prof. Kawakami Rei at Tokyo Institute of Technology.

References

[1] Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep Learning with Differential Privacy. In *Proc. ACM SIGSAC conference on computer and communications security*, 308–318.

[2] Bu, Z.; Gopi, S.; Kulkarni, J.; Lee, Y. T.; Shen, H.; and Tantipongpipat, U. 2021. Fast and Memory Efficient Differentially Private-SGD via JL Projections. In *Proc. NeurIPS*, 19680–19691.

[3] Bu, Z.; Mao, J.; and Xu, S. 2022. Scalable and Efficient Training of Large Convolutional Neural Networks with Differential Privacy. In *Proc. NeurIPS*, 38305–38318.

[4] Bu, Z.; Wang, Y.-X.; Zha, S.; and Karypis, G. 2023. Differentially private optimization on large model at small cost. In *Proc. ICML*, 3192–3218. PMLR.

[5] De, S.; Berrada, L.; Hayes, J.; Smith, S. L.; and Balle, B. 2022. Unlocking High-Accuracy Differentially Private Image Classification Through Scale. *arXiv preprint arXiv:2204.13650*.

[6] Dwork, C. 2006. Differential Privacy. In *International Colloquium on Automata, Languages, and Programming*, 1–12. Springer.

[7] Lei, J. 2011. Differentially Private m-Estimators. In *Proc. NeurIPS*.

[8] Li, T.; Li, J.; Liu, Z.; Li, P.; and Jia, C. 2018. Differentially Private Naive Bayes Learning Over Multiple Data Sources. *Information Sciences*, 444: 89–104.

[9] Li, X.; Tramer, F.; Liang, P.; and Hashimoto, T. 2021. Large Language Models Can Be Strong Differentially Private Learners. *arXiv preprint arXiv:2110.05679*.

[10] Makhdoumi, A.; and Fawaz, N. 2013. Privacy-Utility Tradeoff Under Statistical Uncertainty. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 1627–1634. IEEE.

[11] Shokri, R.; Stronati, M.; Song, C.; and Shmatikov, V. 2017. Membership Inference Attacks Against Machine Learning Models. In *Proc. IEEE Symposium on Security and Privacy (SP)*, 3–18. IEEE.

[12] Tramer, F.; and Boneh, D. 2020. Differentially Private Learning Needs Better Features (or Much More Data). *arXiv preprint arXiv:2011.11660*.

[13] Zhang, Z.; Liu, Q.; Huang, Z.; Wang, H.; Lee, C.-K.; and Chen, E. 2022. Model Inversion Attacks Against Graph Neural Networks. In *Proc. IEEE Transactions on Knowledge and Data Engineering*. IEEE.