Combining Data Privacy and Byzantine Resilience in Distributed Machine Learning

Qingyi Chen
University of Michigan
chenqy@umich.edu

Yile Gu
University of Michigan
yilegu@umich.edu

Lilong Teng
University of Michigan
lilongt@umich.edu

Abstract

Our project investigates how to practically combine data privacy and Byzantine resilience in distributed machine learning. Our work is inspired by [11], where the paper predicts that the training batch size b must grow as fast as the square root of the parameter size d to achieve both data privacy and Byzantine resilience. However, this conclusion, when applied on large neural network, yields unrealistically large batch size, making it impossible to combine data privacy and Byzantine resilience. Therefore, in this project, we will verify and examine the original paper's idea, then experiment with an alternative data privacy algorithm other than the one used in the original paper. Finally, we will revisit the role of batch size in combining data privacy and Byzantine resilience and argue that batch size does not need to grow infinitely as parameter size grows. Therefore, it is possible and practical to combine data privacy and Byzantine resilience, even for huge neural networks.

1. Introduction

Machine learning (ML) is currently a widely researched field to generalize data and make predictions for a wide range of applications. However, modern neural network models are usually trained with huge number of parameters up to the magnitude of billions, using massive data. Therefore, to relieve the stress of computation time and computation resources on a single machine, distributed machine learning systems are introduced and put into practice.

Distributed machine learning (distributed ML) is multi-node machine learning systems that are designed to improve performance, increase accuracy and scale to larger input data sizes[8]. Multiple nodes in a distributed system will collaborate and train on the dataset collectively, producing a joint model. There are several challenges in designing such distributed machine learning systems, including high requirement of network bandwidth, synchronization and latency of the system[16]. In this project, we will be specifically dealing with the problem of data privacy, Byzantine resilience, and combining them in the context of distributed ML.

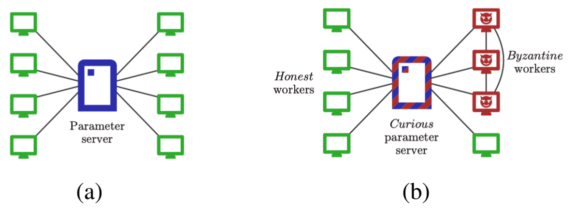


Figure 1: Parameter server model, credits to [11] (a) trusted parameter server with 8 honest workers (b) honest but curious parameter server with 8 workers, including 3 Byzantine nodes

Our distributed ML scheme is formulated as follows and illustrated as 1, borrowing the picture from [11]. At the beginning of each iteration, a central trusted entity, **the parameter server**[16], sends to the nodes the current model (parameters). The nodes then compute their own gradients with their own data using Stochastic Gradient Descent algorithm (SGD), and send their gradients back to the parameter server, where the parameter server collects and aggregates the gradients to update the model parameters.

This scheme, while serves well the purpose of distributing work among multiple workers and thus accelerating learning, also gives rises to two major problems: data privacy and Byzantine gradients. As our problem setting, we consider the possibility of an honest-but-curious parameter server that might take advantage of the gradients from nodes to steal data, and the possibility of Byzantine workers that might send Byzantine gradients to interfere learning.

1.1. Data Privacy

In our distributed ML model, it is imperative that nodes send gradients to the central parameter server. However, this causes undesired potential leakage of each nodes' own data, as gradients are known to leak original data[21].

1.2. Byzantine Gradients

Byzantine nodes, or faulty nodes in the system may send two types of Byzantine gradients: erroneous gradients that does not represent the true training data due to arbitrary faults, or malicious gradients that attempts to poison the learning on purpose [11]. The existence of either type of Byzantine gradients may prevent the model from converging, and thus the model does not reach a satisfying state with accuracy close to that trained in a fault-free system.

2. Related Work

2.1. Data Privacy in Distributed Machine Learning

Communicating with gradients is the main method of distributed SGD, and the gradients are previously believed to be safe to share. Therefore, collaborative learning communicating with gradients are widely used in studies where training data should remain private, such as training medical models using patient data among hospitals [13]. However, [21] points out that the publicly shared gradients in fact contain information of the private training data, and the recovery could be pixel-wise accurate for images and token-wise for texts. They develop the DLG attack[21] demonstrating how a curious parameter server can steal all training data from received gradients.

In response, there are several implementations of distributed SGD designed to preserve data privacy, one of which is noise injection. Gaussian or Laplace noise is injected to the gradients so that differential privacy (DP) can be preserved [11]. For any of two adjacent data batches ξ and ξ' and any possible output set O, a randomized algorithm $\mathcal{M}$ is (ϵ, δ)-differentially private if

$$P[\mathcal{M}(\xi) \in O] \leq e^{\epsilon} \times P[\mathcal{M}(\xi') \in O] + \delta \qquad (1)$$

where $\epsilon > 0$ and $\delta \in [0, 1]$.

2.2. Byzantine Fault Tolerance in Distributed Machine Learning

In a distributed system, some of the nodes may be unreliable and perform Byzantine behaviors. Therefore, Byzantine attacks are developed to test the robustness of distributed ML systems. Two state-of-the-art attacks, *A Little is Enough*[1] and *Fall of Empires*[19] use the same core principle as follows: at each step t, each Byzantine worker submits the same Byzantine gradient $\bar{g}_t + v a_t$, where $\bar{g}_t$ is an approximation of the real gradient $\nabla Q(w_t)$, $a_t \in \mathbb{R}^d$ is an attack vector depending on the type of attack, and $v \in \mathbb{R}^*$ is a non-negative constant [11].

To make a distributed system Byzantine-resilient, gradient aggregation rules (GAR), such as Krum[2], MDA[7], and Median[20] are designed to tolerate a limited degree of Byzantine failures in the system [11]. In our study, we will follow MDA as the aggregation rule in the parameter server. We use (α, f)-Byzantine resilience introduced by Blanchard et al. [3] as a standard for a GAR system that tolerates up to f Byzantine failures.

2.3. Combining Data Privacy and Byzantine Resilience in Distributed Machine Learning with SGD

Guerraoui et al. mentions a sufficient condition in [11] that ensures Byzantine resilience. It is an inequality called variance-to-norm ratio (VN ratio) [11]. Combining differential privacy and (α, f)-Byzantine resilience based on VN ratio introduces a bound that limits the size of the model and the batch size. The VN ratio can only hold when the batch size is big enough or the proportion of Byzantine nodes is small enough [11], and thus differential privacy and Byzantine resilience is not always compatible.

Guerraoui et al. recently proposed a relaxation of the VN condition, called η-approximated VN condition, and a hyperparameter optimization (HPO) scheme to combine data privacy and Byzantine resilience by experimenting on neural networks [10]. Compared to regression models, neural networks usually have more data points for each training data. The experiment using neural networks shows that though combining differential privacy and Byzantine resilience is possible, it is extremely expensive, and the batch size is still the bottleneck.

3. Methods

We start by studying and developing theoretical upper bounds for combining data privacy and Byzantine resilience. We use Gaussian noise injection[6] to provide privacy for the gradients submitted to the master. Experiments based on theoretical findings show that batch size is the key factor to ensure training convergence of the SGD algorithm. We then conduct further experiments to study the effects of using alternative data privacy measures such as gradient sparsification[21]. At last, we revisit the batch size from a theoretical view and demonstrate that the reason why batch size is important lies in its effect on data privacy instead of guaranteeing Byzantine resilience. In light of this observation, we argue that while a large batch size is important for the model to converge in the context of combining Data Privacy and Byzantine resilience, the demand on the batch size is not unlimited; instead, a certain (even a large constant) batch size is good enough: it does not need to unboundedly grow with model's parameter size.

3.1. VN ratio condition: Is it the necessary condition?

VN ratio condition [17] is a sufficient condition for a GAR F to provide $(\alpha, f)-$Byzantine resilience, where α is the vector angle bound to ensure that the true gradient and the aggregated gradient via GAR are sufficiently close and f is the number of workers tolerable to be Byzantine.

Let G_t be the gradient to submit to the master at time t, then by definition of SGD algorithm $\mathbb{E}\left[G_t\right]$ is equal to true gradient $\nabla Q\left(w_t\right)$, where Q is the cost function to optimize for ML problem and w_t is the set of weight parameters for the model at time t. The VN ratio condition is given by

$$\frac{\sqrt{\mathbb{E}\left[\left\|G_t-\mathbb{E}\left[G_t\right]\right\|^2\right]}}{\left\|\mathbb{E}\left[G_t\right]\right\|} \leq k_F(n, f) \qquad (2)$$

VN ratio condition states that the standard deviation of the submitted gradients should be smaller than the true gradient multiplied by a constant $k_n(n, f)$ that is dependent on specific GARs [2, 7].

VN ratio condition with differential privacy To combine data privacy with Byzantine resilience, [11] derives relationship between Byzantine tolerance and model settings such as number of trainable parameters and batch size. They chose Gaussian noise injection to provide differential privacy. Let b be the batch size of a ML model, n be the total number of distributed workers and d be the total number of trainable parameters in the ML model. Table 1 shows the upper bounds for various GARs concluded from [11] to satisfy the VN ratio condition.

One key takeaway is that when combining VN ratio condition with differential privacy, the batch size of the model has to be at least in the order of square root of number of parameters to guarantee $(\alpha, f)-$Byzantine resilience. This is hardly a feasible condition to match for current hardware, since typical deep learning models like ResNet-50[12] have dozens of millions of trainable parameters, resulting in a batch size typically greater than 1000.

Relaxing the VN ratio condition Since the VN ratio condition only serves as a sufficient condition for Byzantine resilience, the relationships given in Table 1 are not the tightest bounds. [10] provides theoretical insights to relax the VN ratio condition. Let η be a threshold for gradient, define $\eta-$approximated VN condition as below. For $\eta \geq 0$, the $\eta-$approximated VN condition is satisfied if for all $\|\mathbb{E}[G(\theta)]\| > \eta$,

$$\kappa_F(n, f)^2 \mathbb{E}\left[\|G_t-\mathbb{E}[G_t]\|^2\right] < \|\mathbb{E}[G_t]\|^2 \qquad (3)$$

The main difference to the original VN condition is that now a subset of gradients smaller than the threshold (and assumed to be close to optimized minimum) does not abide the inequality. Let v be the standard deviation of the loss function in the ML model. It is then concluded that if no Gaussian noise is injected, the $\eta-$approximated VN condition holds true for $\eta \geq (\frac{\sqrt{8}f}{n-f})v$, thus the convergence of training of the model is independent of batch size.

Krum[2], Bulyan[17]	MDA[7]
$b \in \Omega(\sqrt{nd})$	$\frac{f}{n} \in O\left(\frac{b}{\sqrt{d}+b}\right)$

Table 1: Upper bounds for various GARs to satisfy the VN condition.

Despite theoretical improvements on upper bounds of the problem, combining Byzantine resilience with data privacy still imposes requirements on batch size to ensure the convergence of the training of SGD algorithm. [11] only verifies their findings on a simple logistic regression model. We think it is crucial to conduct in-depth experiments on various ML models with different batch sizes for the empirical upper bounds of batch size. The experiment results are introduced in 4.1. We find out that in the presence of gradient attacks, the batch size needed to achieve the same accuracy as the vanilla model is increased by at least two orders of magnitude, verifying that it is non-trivial to combine Byzantine resilience with data privacy.

3.2. Besides Gaussian Noise: Gradient Sparsification as the Data Privacy Algorithm

While the VN condition for combining data privacy and Byzantine resilience is verified to hold through experiments, [11] bases their proof and experiments only on Gaussian Noise as their data privacy algorithm. Notably, it is exactly the noise to ensure data privacy that introduces the term regarding batch size b into the originally data privacy-free VN condition, and yields the new VN condition $\frac{f}{n} \in O\left(\frac{b}{\sqrt{d}+b}\right)$. Therefore, in an attempt to examine whether this new VN condition is a drawback specific to Gaussian Noise algorithm, we conduct experiments with another data privacy algorithm mentioned in [21], "Gradient Compression and Sparsification".

Specifically, we apply gradient sparsification where components of small magnitude in the gradients are pruned to zero. As suggested in [5], this is done by choosing a fixed proportion of gradient to prune (note that pruning is done separately within positive and negative components). Also, [21]'s experiments have shown that a portion of 20% is sufficient to defend against data privacy attacks, so we choose the proportion to be 25%.

As shown in the experiments though, using gradient sparsification as the data privacy algorithm does not demonstrate an independence of performance from batch size. Large batch size is still the key factor to enable the model to learn well when combining data privacy and Byzantine resilience. To make sense of it, we argue that gradient sparsification can be viewed as another form of noise whose effect is cancelling the small components in gradients to zero.

Therefore, it is essentially similar to Gaussian noise algorithm, and original conclusion applies: a large batch size is important in order for the model to learn well.

3.3. Revisiting Batch Size in Combining data privacy and Byzantine resilience

Although we've replaced Gaussian noise injection with gradient sparsification, the batch size of model remains to be the key factor ensuring training convergence of the SGD algorithm. This observation urges us to revisit the effect of batch size in combining data privacy and Byzantine resilience.

Augmented VN condition for differential privacy Recall that Table 1 lists theoretical upper bounds for various GARs to satisfy the VN ratio condition. It is important to realize how to derive these bounds from a single inequality. In the VN ratio condition, E $\left[\|G_t - \mathbb{E}\left[G_t\right]\|^2\right]$ is the variance of the submitted gradient. Notice that if we account for Gaussian noise in gradients, then $G_t = G_t^s + O_t$, where G_t^s is the original submit gradient, and O_t is the Gaussian noise with mean equals to 0 and standard deviation $s = \frac{2G_{\max}\sqrt{2\log(1.25/\delta)}}{b\epsilon}$, where G_{max} is the maximum possible value of gradient in the model and δ and ϵ are two hyperparameters for Gaussian noise (this is also known as (ϵ, δ)-differential privacy). Then the left side of Equation 2 can be simplified as:

$$\frac{\sqrt{\mathbb{E}\left[\|G_t^s + O_t - \mathbb{E}\left[G_t^s + O_t\right]\|^2\right]}}{\|\mathbb{E}\left[G_t^s + O_t\right]\|} = \frac{\sqrt{Var[G_t^s + O_t]}}{\|\mathbb{E}\left[G_t^s + O_t\right]\|}$$

$$= \frac{\sqrt{Var[G_t^s] + Var[O_t]}}{\|\mathbb{E}\left[G_t^s\right]\|}$$

Now if we plug in the value for variance of noise, we obtain the augmented VN condition for differential privacy:

$$\frac{\sqrt{\mathbb{E}\left[\|G_t^s - \mathbb{E}\left[G_t^s\right]\|^2\right] + 8d\frac{G_{\max}^2}{\epsilon^2 b^2}\log\left(\frac{1.25}{\delta}\right)}}{\|\mathbb{E}\left[G_t^s\right]\|} \leq k_F(n, f)$$

Compared to the original VN condition in Equation 2, the only additional term is the variance of noise on the denominator. And since variance is always positive, adding this term increases the left side of the equation and makes satisfying the VN ratio condition more difficult. More importantly, the variance of Gaussian noise is inversely proportional to the square of batch size b. Therefore, as batch size increases, the variance of noise decreases, making it more possible to be bounded by $k_F(n, f)$.

Therefore, we are safe to conclude that it is due to preserving differential privacy that batch size is included in the VN condition to guarantee Byzantine resilience.

Increasing batch size is sufficient to preserve data privacy We continue to study if it is necessary to introduce Gaussian noise during training for data privacy. [21] is the first to raise awareness of data leakage from gradients. It is possible to recover original training image by training a network that minimizes between the original gradient and the constructed gradient. Another finding is that as the batch size increases, the number of training steps needed to recover the original image grows in factorial.

[9] is able to conduct a scalable experiment on reconstructing training images from gradients for various batch sizes. We conduct an experiment in 4.3 to study how batch size affects reconstruction of data.

Our key finding is that when batch size is large, it is infeasible to reconstruct original images from gradient. This is because either the network returns meaningless noise, or the time it takes for reconstruction is unrealistic in a distributed ML setting. Therefore, we conclude that it is sufficient to increase batch size for preserving data privacy.

[11] states that batch size has to grow with model's parameter size in order to combine data privacy and Byzantine resilience. Their conclusion is derived from Gaussian noise injection which binds batch size with VN condition to guarantee Byzantine resilience. This makes the model vulnerable to gradient attacks when Gaussian noise is present. Therefore, enlarging batch size makes sure that the servers have stronger Byzantine resilience. However, we observe that batch size should serve as the measure to preserve data privacy instead of guaranteeing Byzantine resilience, and thus Gaussian noise injection is not necessary when batch size is large. As a result, batch size does not grow with respect to model's parameter size and is not necessarily bounded by the upper bounds in Table 1.

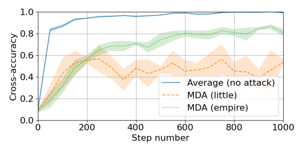
4. Experiments

4.1. Verification of VN Condition

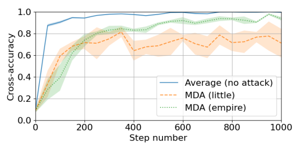
An implication of Table 1 is that, in the context of combining data privacy and Byzantine resilience, in order for the model to converge (learn well), the batch size must be large enough. Therefore, we conduct experiments and plot the learning process of the model using different data privacy (as differentiated by ϵ), Gradient Aggregation Rules (as "Average" or "MDA"), and Byzantine attacks (for example, little[1] and empire[19] attack) versus various batch sizes. Note that when averaging is used as the Gradient Aggregation Rule, we take it as the baseline and no attacks are present; otherwise, among $n = 11$ servers, $f = 5$ are Byzantine. The model applied here is a basic Convolutional Neural Network (CNN), and the dataset it is learning is MNIST[4], a simple dataset of numbers from 0 to 9.

The plots on CNN without data privacy (i.e., $\epsilon = \infty$ for Gaussian noise) on different batch sizes $b =$

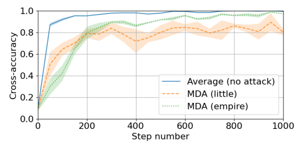
$10, 50, 100, 500$ are presented as 2a, 2b, 2c, 2d, respectively.



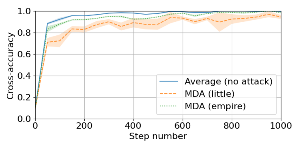
(a) $b = 10$



(b) $b = 50$



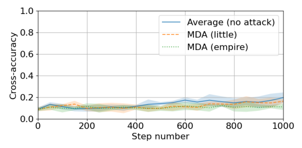
(c) $b = 100$



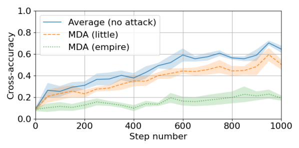
(d) $b = 500$

Figure 2: Training process of CNN with no Gaussian Noise and batch size $b = 10, 50, 100, 500$

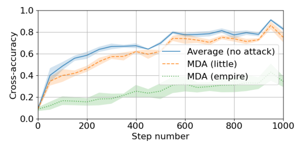
The plots on CNN with strong data privacy (i.e., $\epsilon = 0.2$ for Gaussian Noise) on different batch sizes $b = 10, 50, 100, 500$ are presented as 3a, 3b, 3c, 3d, respectively.



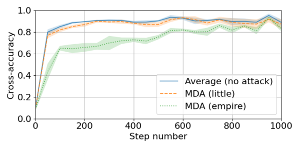
(a) $b = 10$



(b) $b = 50$



(c) $b = 100$



(d) $b = 500$

Figure 3: Training process of CNN with Gaussian Noise and batch size $b = 10, 50, 100, 500$

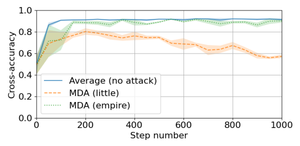
The plots verify that:

1. The introduction of data privacy hinders the model's learning (even without Byzantine gradients). Comparing the model's learning process with and without noise as 3a and 2a, we can see that the model is learning much better without Gaussian Noise.

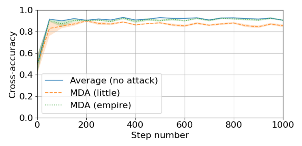
2. Larger batch size recovers model's learning ability. As the batch size grows as 3a, 3b, 3c, 3d, model is performing better with all the Gradient Aggregation Rules.

Also, we conduct similar experiments on another machine learning model - Support Vector Machine (SVM). SVM features simpler structure and fewer parameter size, so it is worthwhile to examine if fewer batch size will work for SVM. The dataset it is trained on is phishing dataset[1], a dataset collecting legitimate as well as phishing website instances.

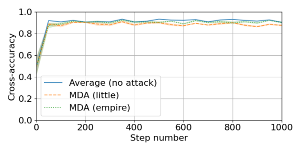
The plots on SVM without data privacy (i.e., $\epsilon = \infty$ for Gaussian Noise) on different batch sizes $b = 10, 50, 100, 500$ are presented as 4a, 4b, 4c, 4d, respectively.



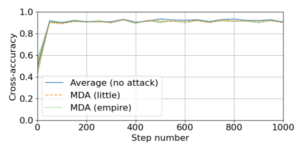
(a) $b = 10$



(b) $b = 50$



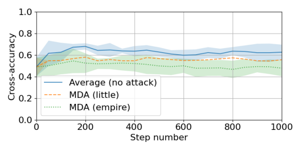
(c) $b = 100$



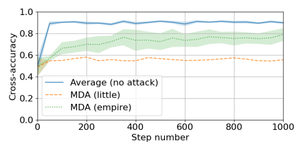
(d) $b = 500$

Figure 4: Training process of SVM with no Gaussian Noise and batch size $b = 10, 50, 100, 500$

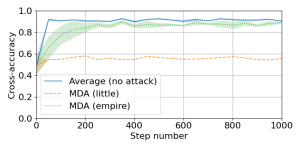
The plots with strong data privacy (i.e., $\epsilon = 0.2$ for Gaussian noise) on different batch sizes $b = 10, 50, 100, 500$ are presented as 5a, 5b, 5c, 5d, respectively.



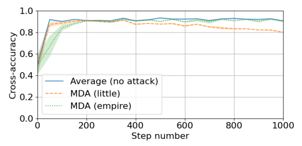
(a) $b = 10$



(b) $b = 50$



(c) $b = 100$



(d) $b = 500$

Figure 5: Training process of SVM with Gaussian Noise and batch size $b = 10, 50, 100, 500$

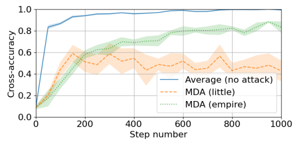
The plots on SVM demonstrate similar patterns as the ones on CNN. An intuitive explanation is that, the introduction of Gaussian Noise modifies the original gradient and therefore interferes with the learning process, while larger batch size yields larger and more stable gradients, making the noise added less influential and more negligible.

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

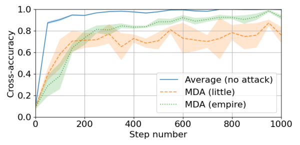
Across the two models CNN and SVM, we would also like to make a point qualitatively here by comparing 3d and 5c that, batch size is indeed required to increase as the model parameter size grows. For SVM whose parameter size $d_{svm} = 69$, with both data privacy and Byzantine resilience preserved, batch size $b = 100$ is nearly sufficient for the learning process to recover to normal; on the other hand, for CNN whose parameter size $d_{cnn} = 431080$, when combining both data privacy and Byzantine resilience, the learning process does not recover until batch size $b = 500$ is reached.

4.2. Using Gradient Sparsification as the Data Privacy Algorithm

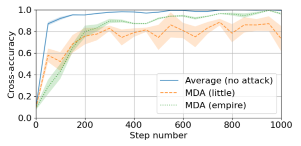
Using gradient sparsification (with the pruning proportion chosen to be 25%) as the data privacy algorithm in place of Gaussian noise yields the plots as 6 with batch size $b = 10, 50, 100, 500$.



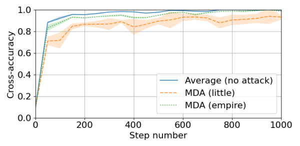
(a) $b = 10$



(b) $b = 50$



(c) $b = 100$



(d) $b = 500$

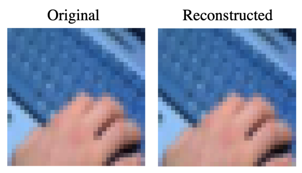
Figure 6: Training process of CNN with Gradient Sparsification and batch size $b = 10, 50, 100, 500$

Still, we observe the phenomenon that a larger batch size is important for the Byzantine-resilient model to learn well with the alternative data privacy algorithm, gradient sparsification. We would like to conclude that the new VN condition is not specific to Gaussian noise, but is more likely to apply generally to data privacy algorithms.

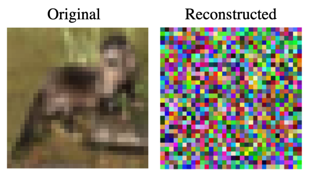
4.3. Reconstructing training images from gradients

We first follow [21] and evaluate on reconstructing training images from gradients. We train a LeNet[15] model on CIFAR-100[14] dataset. In the middle of the training, we initialize dummy images and dummy gradients, and use training gradients calculated from a batch to minimize the difference between the two gradient sets. Figure 7 shows reconstruction results for batch size $b = 1$ and $b = 4$ respectively.

We run 300 steps when batch size is 1 and 1000 steps when batch size is 4. [21] claims that reconstruction is possible when batch size is greater than 1. However, we find out that this conclusion is data-specific and in most cases we get meaningless noise for greater batch size, although the difference between gradients are minimized.

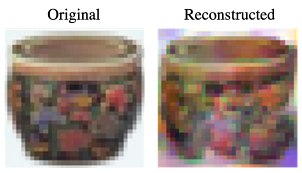


(a)

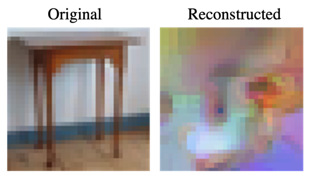


(b)

Figure 7: Reconstruction result with LeNet when batch size (a) $b = 1$ (b) $b = 4$



(a)



(b)

Figure 8: Reconstruction result with ResNet32 for batch size 100 (a) most recognizable (b) average result

Then we follow [9] to train a larger ResNet32 network for reconstructing gradients from CIFAR-100 datasets. We set batch size to be 100, one image for each class. The reconstruction result is shown in Figure 8.

We find that even the most recognizable classes (i.e. bowl) have conspicuous artifacts. On average, the original classes (i.e. table) can hardly be recognized from the reconstructed images. What's worse, the network is trained for more than 20,000 steps to produce reconstruction results, which takes more than 12 hours. Therefore, we believe that it is impractical for the master server to reconstruct data on the fly from gradients when batch size is large. Enlarging batch size provides enough data privacy for distributed ML.

5. Discussion

Our assumption and induction are mainly based on view of the VN ratio condition and gradients. From a broader view of distributed ML systems, we believe there can be also other practical solutions to the problem, such as detection of Byzantine gradients by suspicion-based GARs[18], more crafted protocols to exchange gradients, etc.

In addition, although relaxing VN ratio condition provides theoretical improvements on the upper bounds for combining data privacy and Byzantine resilience, previous analysis is based on the assumption of Gaussian noise. We believe it is possible to derive tighter upper bounds from a theoretical point of view if Gaussian method is replaced.

6. Conclusion

We confirm that including data privacy via Gaussian noise injection in the meantime of preserving Byzantine resilience in a distributed ML system makes it more difficult to converge, and the learning ability recovers as the batch size increases. Our finding shows that the noise injection may not be the best method to achieve data privacy in a distributed ML system where Byzantine nodes is a problem. The key point of ensuring data privacy should be the batch size, regardless of which of the current data privacy method is used. A large enough but feasible batch size may ensure data privacy and Byzantine resilience at the same time, and future work can explore a relationship for aggregating privacy and Byzantine resilience with respect to batch size.

7. Work Distribution

Qingyi Chen ran experiments on verifying [11], implemented and experimented gradient sparsification, literature reviewed papers on data leakage through gradients and combining data privacy and Byzantine resilience, and wrote the report.

Yile Gu conducted experiments on verifying [11], implemented data reconstruction from gradients with varying batch size, evaluated results from [21] and [9] and reviewed papers on combining data privacy and Byzantine resilience, and wrote the report.

Lilong Teng reviewed literature on differential privacy and Byzantine resilience in SGD [11] and combining differential privacy and Byzantine resilience [10], and write up the background, related work and conclusion.

References

[1] M. Baruch, G. Baruch, and Y. Goldberg. A little is enough: Circumventing defenses for distributed learning, 2019. 2, 4

[2] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2, 3

[3] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Staine. Machine learning with adversaries: Byzantine tolerant gradient descent. International Conference on Neural Information Processing Systems, 2017. 2

[4] L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 4

[5] N. Dryden, T. Moon, S. A. Jacobs, and B. Van Essen. Communication quantization for data-parallel training of deep neural networks. In *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*, pages 1–8, 2016. 3

[6] C. Dwork. Differential privacy. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 2

[7] E.-M. El-Mhamdi, R. Guerraoui, A. Guirguis, L. N. Hoang, and S. Rouault. Genuinely distributed byzantine machine learning, 2020. 2, 3

[8] A. Galakatos, A. Crotty, and T. Kraska. Distributed machine learning, 2018. 1

[9] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients – how easy is it to break privacy in federated learning?, 2020. 4, 6, 7

[10] R. Guerraoui, N. Gupta, R. Pinot, S. Rouault, and J. Stephan. Combining differential privacy and byzantine resilience in distributed sgd, 2021. 2, 3, 7

[11] R. Guerraoui, N. Gupta, R. Pinot, S. Rouault, and J. Stephan. Differential privacy and byzantine resilience in sgd: Do they add up?, 2021. 1, 2, 3, 4, 7

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. 3

[13] A. Jochems, T. M. Deist, I. E. Naqa, M. Kessler, C. Mayo, J. Reeves, S. Jolly, M. Matuszak, R. T. Haken, J. van Soest, and et al. Developing and validating a survival prediction model for nsclc patients through distributed learning across 3 countries, 2017. 2

[14] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-100 (canadian institute for advanced research). 6

[15] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 6

[16] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the parameter server, 2014. 1

[17] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault. The hidden vulnerability of distributed learning in byzantium, 2018. 2, 3

[18] C. Xie, O. Koyejo, and I. Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance, 2019. 6

[19] C. Xie, S. Koyejo, and I. Gupta. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation, 2019. 2, 4

[20] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates, 2021. 2

[21] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients, 2019. 1, 2, 3, 4, 6, 7