Stationarity of Crucial Gradients in Deep Neural Networks

Anonymous Authors[1]

Abstract

Modern deep learning models trained under federated learning setting contain hundreds of millions of parameters. This leads to considerable storage space consumption on mobile systems, and significant communication bandwidth requirements for exchanging gradients between edge devices and central server. Recently many pruning methods attempting to learn a sparse model have been suggested to tackle the memory concerns. To control the communication overhead, many gradient sparsification techniques have been suggested based on the sparse distribution of local gradients vectors. Such sparsification techniques like top-k and rTop-k transmit only few large-magnitude gradients, referred to in this paper as the ”crucial gradients”. Our work aims to connect the model pruning literature and gradient sparsfication literature, by making the important observation that the positions of crucial gradients in deep neural networks is static, which leads to bulk of the model parameters never getting updated under such sparsification techniques. We apply this observation to MNIST image classifiers to prune up to 99% network, causing considerable reduction in storage space requirement. We also study the tradeoff between exploration and exploitation in large gradient vectors for obtaining high model accuracy. Numerical experiments show that our pruning approach does not damage the final accuracy of the global model under top-k and rTop-k sparsification techniques.

1. Introduction

Nowadays, vast amount of data is generated by autonomous driving, smart phones, extended reality technologies, various Internet-of-things (IoT) devices, and so on. Machine learning approaches are developed to exploit these massive datasets to perform intelligent predictions and inferences. The most of the current machine learning approaches are based on centralized algorithms. However, centralized algorithms are too costly since offloading the data in local devices to a central server cannot be practicable due to privacy and latency constraints.

Federated Learning (FL) is proposed to address these issues by (McMahan et al., 2017). FL is a distributed machine learning approach that enables training on the decentralized data in devices like smart phones, IoT devices and so on. It can be simply described as an approach that brings the training model to the data instead of data to the training model (Bonawitz et al., 2019). It is a novel approach that edge devices collaboratively learn a shared model under the orchestration of a central server without sharing their training data.

Although FL is a promising paradigm for the problems arising from centralized approaches, it also faces several challenges. Communication overhead is one of the major challenges (Kairouz et al., 2019). Specifically, edge devices download the shared model from the central server at each global iteration. They compute their local updates with this shared model by using their own datasets and these local updates are gathered to update the shared model. Although only model updates are transmitted to the central server by edge devices instead of the training data, model updates can contain hundreds of millions of parameters in modern ML models, e.g. deep neural networks, depending on the complexity of the model. It results in high bandwidth usage (Hu et al., 2020).

Some works focused on designing new gradient descent methods to improve the communication efficiency by reducing the update frequency of the global model (McMahan et al., 2017; Li et al., 2020; Sahu et al., 2018). In addition to them, there has been recently significant interest in reducing the communication overhead by using various techniques such as gradient quantization, sparsification, and thresholding (Konecný et al., 2016; Suresh et al., 2017; Reisizadeh et al., 2020; Chen et al., 2018; Barnes et al., 2020). Especially, rTop-k sparsification technique proposed by (Barnes et al., 2020), which concatenates random-k and top-k sparsification ideas in the literature, provides promising results. Moreover, network pruning is an alternative approach to increase the communication efficiency between edge devices and the central server since the network complexity has an explicit impact on the size of the parameter update vector that is needed to be exchanged between edge devices and the central server. The work by (Jiang et al., 2019) shows

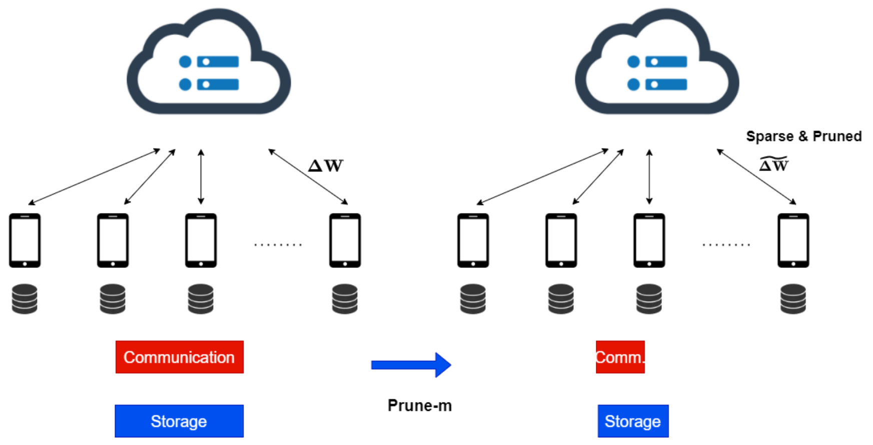

Figure 1. Prune-m reduces communication and storage bandwidth.

that the pruned model converges to the very similar accuracy of the original model in FL setting with much smaller model size.

Contributions. In this work, we compose network pruning and rTop-k techniques. We aim to develop a model that provides better accuracy than only rTop-k and top-k techniques with the less system complexity.

2. Background

2.1. Federated Learning

Federated learning (McMahan et al., 2017) is a machine learning technique which trains a high-quality centralised model while preserving user privacy with its unique distributed learning approach. Unlike other collaborative learning methods, in a federated learning system, all the participants(called workers) receive a copy of the global model from a central server, that can be trained on the participant's private data. Then the model updates or gradients from the individual local models are sent back to the central server, that aggregates these gradients and updates the global model again. This training process is then repeated over multiple communications rounds until a desired level of accuracy is attained (usually one communication round corresponds to one epoch in federated learning setting). In this manner, none of the training data is ever transmitted from the participants, only the updates related to the model are.

A typical federated learning setting with learning rate η has W workers, with worker w having n_w samples. Worker w performs performs one step of full-batch gradient descent on its local data with model θ_t at iteration t to compute the average local gradient g_w as follows,

$$g_w = \frac{1}{n_w} \sum_{i=1}^{n_w} \nabla f(x_i, \theta_t) \tag{1}$$

where $f(x_i, \theta_t)$ is the loss computed from data sample x_i of local dataset. The central server then takes a weighted average of these gradients and applies the following update to the global model.

$$\theta_{t+1} = \theta_t - \eta \sum_{w=1}^{W} \frac{n_w}{n} g_w \tag{2}$$

This step is called federated averaging, which produces best performance when all workers start by downloading the model with same initialisation, n being the aggregate number of samples from all workers. This technique removes the need to transmit entire dataset to the central server, thus saving network latencies, along with preserving user privacy. For multiple communications in a single epoch, the central server performs the following update.

$$\theta_{t+1} = \theta_t - \eta \frac{1}{|B|} \sum_{w=1}^{W} \sum_{x \epsilon B_{w,t}} \nabla f(x, \theta_t) \tag{3}$$

where $|B| = \sum_{w=1}^{W} |B_{w,t}|$, with $|B_{w,t}|$ representing the size of mini-batch $B_{w,t}$ at worker w during iteration t.

2.2. Gradient Sparsification

For large models deployed in modern federated learning contexts with millions of parameters, it is well-known that full-precision gradient aggregation is costly. Additionally mobile devices have further communication limitations arising out of low network bandwidth lanes, irregular network connections, and costly mobile data plan, making training slow, erratic and costly.

To reduce the communication impediments, recent works have proposed communicating a sparsified, quantized or randomly subsampled version of the updates. Latest sparsification technqiues like top-k (Lin et al., 2018) and rTop-k

(Barnes et al., 2020) make use of the skewed and sparse nature of the local gradients observed in experiments. The top-k technique sends only top k gradients of largest magnitudes in each iteration, thus allowing us to exploit the most significant entries of the gradient vector. On the other hand, authors in (Barnes et al., 2020) analyse a sparse Bernoulli model in a distributed statistical parameter estimation setting to study the impact of a highly skewed distribution for gradient magnitudes. Taking inspiration from the analysis of this highly simplified model, the authors suggest the rTop-k algorithm, which sends a random subset of k gradients from top r gradients of largest magnitudes in the local gradient vector, thus reducing any bias introduced in top-k sparsification.

Our work shows that the top gradients, referred to in this paper as the ”crucial gradients”, are concentrated in specific regions of gradient vector throughout the training process, which limits our ability to explore all parameters of a model, thereby leading to majority of the model parameters never getting updated or explored under such sparsification techniques. We also study the tradeoff between exploration and exploitation of gradients for achieving the optimal performance in global model.

2.3. Network Pruning

Large scale deep neural networks can consume enormous amounts of memory and computation. These requirements not only increase infrastructure costs, but also make deployment of networks to resource-constrained edge devices in federated learning environments difficult (Blalock et al., 2020). A widely-accepted approach for reducing these infrastructure requirements is neural network pruning, which involves systematically removing parameters from an existing network to produce a smaller network whilst almost retaining model accuracy.

A given neural network configuration can be interpreted over time using an infinite family of potential functions $N(x;\theta)$ characterised by parameters θ, with x denoting the input to the neural network. Internally the neural network model can include any arrangement of parameters and sets of operations like convolution, pooling, batch normalisation, etc. Network pruning involves transforming a given model $N(x;\theta)$ to a new model $N(x;M\odot\theta')$, where the parameters θ' can be different from θ, $M\epsilon\{0,1\}^{|\theta'|}$ is a bitmask that sets certain parameters to 0, and $\odot$ is the elementwise product operator. In practice, rather than using an categorical mask, pruned parameters in θ are either frozen to zero or just removed from the network.

In most neural network pruning strategies, the first step is to train the network to convergence and assign to each parameter of the model a score, which is then used to determine the set of parameters to be pruned. The network is usually trained for a longer time than usual (known as fine-tuning) to help the model recover from the loss of accuracy resulting from pruning. The process of pruning and fine-tuning can be repeated many times, gradually compressing the network's size.

3. Spatial Distribution of Crucial Gradients

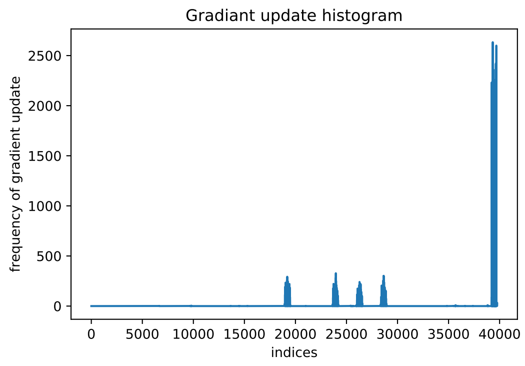

Figure 2. Frequency of updating indices in Top-k method.

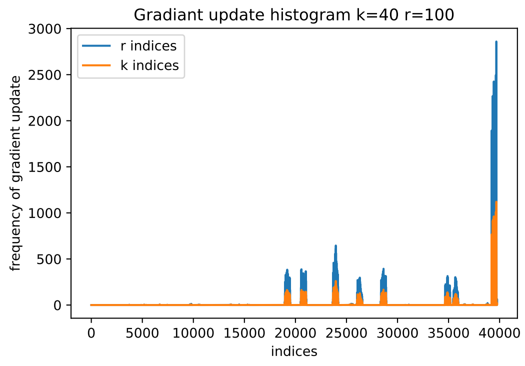

Figure 3. Frequency of updating indices in rTop-k method.

It is widely known that the gradient vectors in large deep learning models are sparse, which are the basis for experimental success of techniques like top-k and rTop-k. These sparsification techniques send out only $k(<<|\theta|)$ gradients from set of crucial gradients at every iteration of training. ($|\theta|$ here is the total number of parameters in the model). However, it is widely believed that the crucial gradients occur all over the gradient vector during different iterations, and hence cover all locations or indices of gradient vector in an i.i.d fashion. In this paper, we break this myth by showing that crucial gradients occur within a fixed set of positions in a large neural network. We plot the number of times a particular gradient location was chosen under top-k sparsification algorithm, as shown in Figure 2. It can seen that majority of the gradients never made it to the top

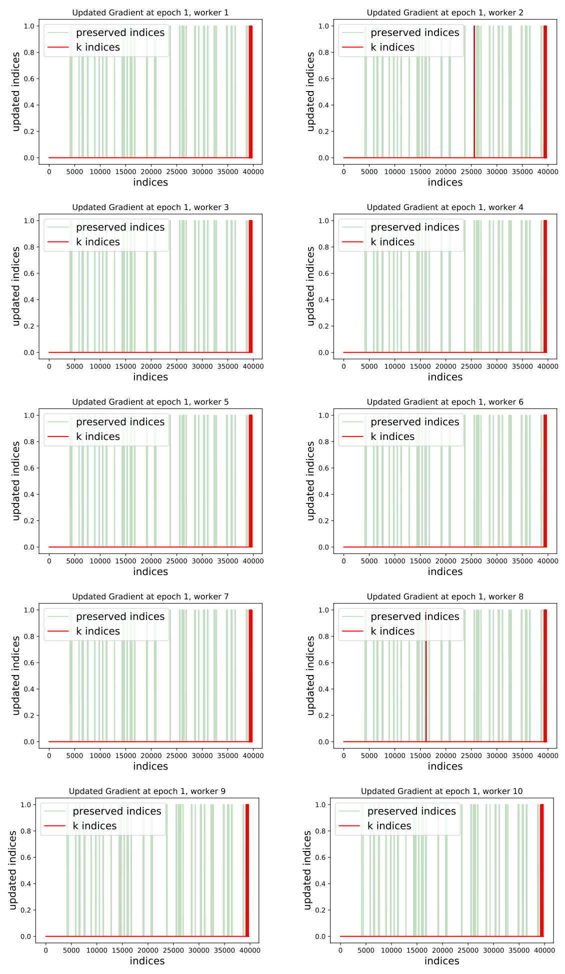

Figure 4. Updated indices in epoch 1.

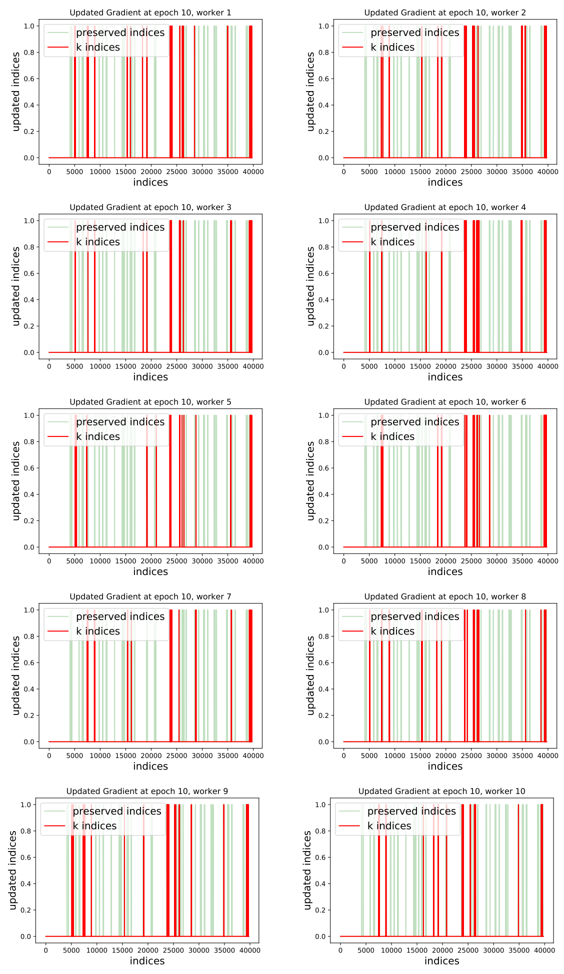

Figure 5. Updated indices in epoch 10.

k set at all during the entire training process. Thus the path traced by gradient descent along the loss landscape is almost completely determined by the a fixed set of positions in the gradient vector. Figure 3 similarly shows frequency of all model parameters under rTop-k sparsification, which again brings home the fact that vast majority of parameters in large neural network are never updated at all during the entire course of training.

3.1. Temporal variation in Spatial Distribution of Crucial Gradients

In the subset of gradients indices with non-zero frequency of selection, the positions of crucial gradients can possibly shift from one region of concentration to a different region of concentration with time. To examine if the position of crucial gradients are indeed time varying within this fixed set of positions, we plot the set of top k gradients (used in top-k technique on unpruned model) in red at epoch 1, epoch 10, epoch 20 and epoch 30 in Figures 4-7, out of total training for 30 epochs. Additionally, we plot the larger set of top $m (> k)$ gradients fixed at first epoch in green in all these graphs. We observe that top k gradients do demonstrate a phased shift to newer locations with increasing iterations, but they still restrict themselves to within the larger set of top m gradients fixed at the beginning. To gauge this phenomenon, we define the following metric to quantify the overlap between the two set of gradients.

$$\text{Overlap score} = \frac{|\{\text{top-k grads}\} \cap \{\text{top-m grads}\}|}{|\{\text{top-k grads}\}|} \times 100 \tag{4}$$

It can be seen from Table 1, that most overlap scores are near 90%. This again reinforces that a smaller group of crucial gradients guides gradient descent to convergence along the manifold of global loss function.

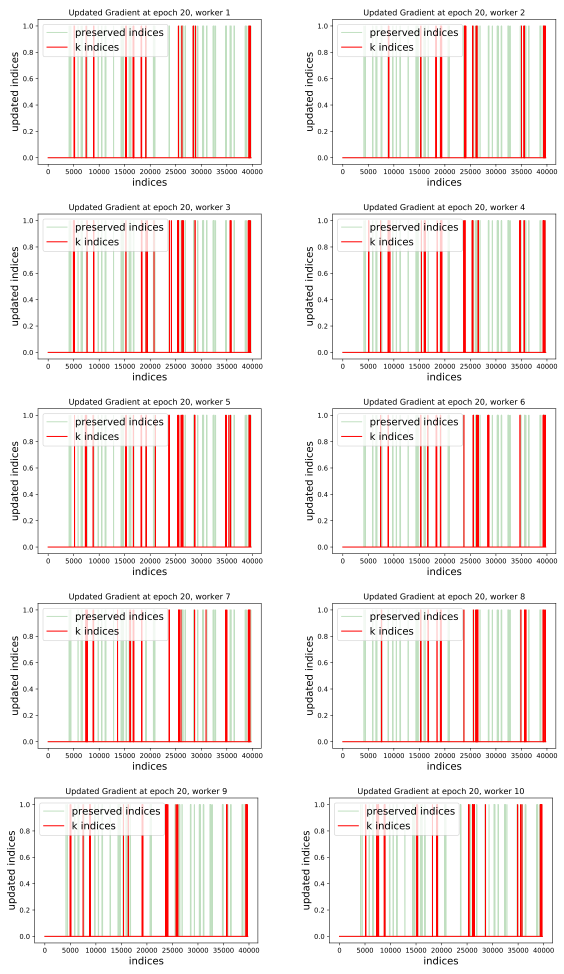

Figure 6. Updated indices in epoch 20.

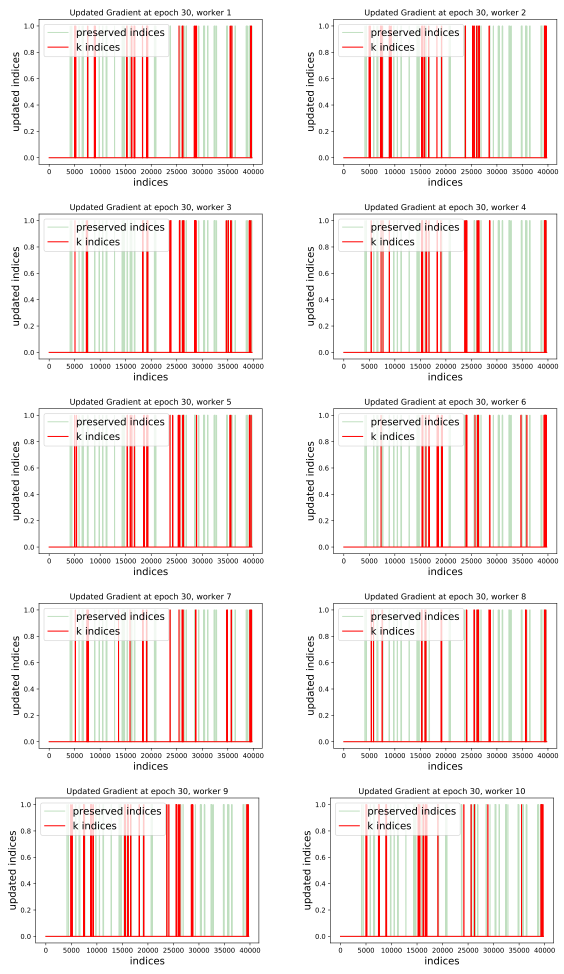

Figure 7. Updated indices in epoch 30.

Table 1. Overlap score for $W = 10$ workers with i.i.d. datasets ($k = 40, m = 4000, |\theta| = 37960$)

W NUM	EPOCH 1	EPOCH 10	EPOCH 20	EPOCH 30
W 1	100	100	97.5	95
W 2	100	95	90	87.5
W 3	95	90	87.5	85
W 4	97.5	87.5	87.5	85
W 5	97.5	85	92.5	90
W 6	95	87.5	87.5	82.5
W 7	95	90	92.5	90
W 8	90	87.5	90	82.5
W 9	90	90	87.5	87.5
W 10	100	95	90	87.5

3.2. Effect of Traininig Dataset on Spatial Distribution of Crucial Gradients

In homogeneous data scenarios where all workers have i.i.d. data, the positions crucial gradients are intuitively expected to be similar, which is also supported by Table 1. To examine if the same set of gradient indices can support the crucial gradients in a neural network irrespective of the dataset applied for training, we consider the non-i.i.d. scenario, where different workers hold different distributions of data. We accumulate all gradient vectors from all non-i.i.d. workers at the central server in first epoch, and choose the set of top m gradient indices after observing the aggregated gradient. We provide each worker with dataset pertaining to a unique category or label. We observe in Table 2 that the overlap scores are relatively lower than the i.i.d. case, suggesting that for crucial gradients there is a certain degree of dependency on dataset being used for training. Hence range of positions spanned by crucial gradients for one worker might not have good overlap with another worker.

Table 2. Overlap score for $W = 10$ workers with non-i.i.d. datasets ($k = 40, m = 4000, |\theta| = 37960$)

W NUM	EPOCH 1	EPOCH 10	EPOCH 20	EPOCH 30
W 1	100	62.5	57.5	60
W 2	100	72.5	75	77.5
W 3	100	80	87.5	82.5
W 4	100	60	70	77.5
W 5	100	92.5	80	85
W 6	100	72.5	82.5	77.5
W 7	100	62.5	60	57.5
W 8	100	60	62.5	70
W 9	100	80	82.5	80
W 10	100	92.5	95	87.5

3.3. Tradeoff between Exploration Vs. Exploitation in the Gradients on Model Accuracy

Motivated by the recent developement of sparsification technqiues like top-k and random-k which award significance to exploitation and exploration, respectively, in gradient vectors, and the superior performance of rTop-k, which exploits top gradients while removing the bias of top-k technique, we study the tradeoff between exploration and exploitation in large gradient vectors for obtaining high model accuracy. We do this analysis by modulating value of r in rTop-k technique, from $r = k$ representing top-k scenario to $r = |\theta|$ representing the random-k scenario, and make a semi-log plot of the global model accuracy as a function of r in Figure 8 . We witness the optimal accuracy to occur for a value of r which is intermediate of the two extremes. The random-k technique specifically demonstrates poor performance, highlighting that it is not a good idea to choose gradients in i.i.d. fashion from entire gradient vector. The graph substantiates that while the higher magnitude gradients are important to drive high the model accuracy, a certain degree of random exploration in the top strata of gradients can further help boost performance, which justifies why rTop-k significantly outperforms top-k or random-k.

4. Prune-m Algorithm

The spatial concentration of set of crucial gradients suggests that large part of network parameters never get updated throughout the training process under the top-k and rTop-k sparsification, since their corresponding gradients are not selected in top-k or top-r set.

$$\theta_{t+1,j} = \theta_{t,j} - \eta \sum_{w=1}^{W} \frac{n_w}{n} \times 0 = \theta_{t,j} \tag{5}$$

where $\theta_{t,j}$ is the jth gradient entry(non-crucial) of the full gradient vector θ at iteration t, for which the gradient value

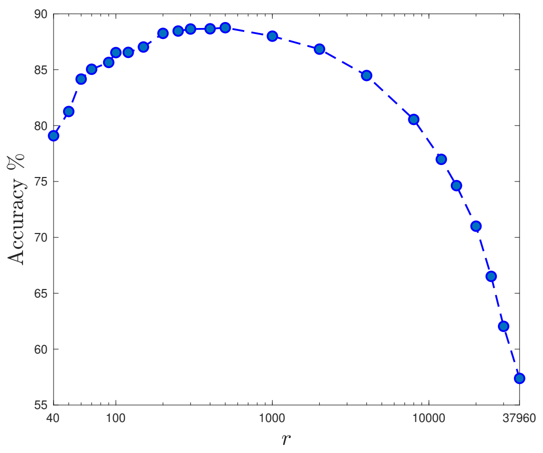

Figure 8. rTop-k parameters: $k = 40, m = 4000, |\theta| = 37960$. Maximum global model accuracy is obtained for an intermediate $r \approx 500$.

was masked/frozen to zero. If all parameters of the global model are initialised close to zero, then the non-crucial gradient locations never update their corresponding parameters, thus keeping them at zero, which leads to implicit pruning of the model. We validate this phenomenon using Prune-m algorithm, described in Algorithm 1.

Since we saw in subsection 3.3 that exploitation with a small amount of exploration helps achieve best accuracy, so our algorithm first picks top m ($k < r < m < |\theta|$) indices of the aggregated gradient vector at the global model, and prunes the remaining network(set the parameter value to zero). In actual federated learning setting, the global model can be first trained at the central server using an auxiliary dataset to determine the top-m gradient locations, and then deploy the pruned model on the mobile systems with limited resources. Next, k gradients are sent from each worker with the remaining model to the global model in each iteration, according to top-k or rTop-k approach.

It is to be noted that the Prune-m is a k-contraction gradient updating techniques, similar to top-k, random-k or rTop-k. In addition, it prunes the local models to just $m (< |\theta|)$ parameters, which can lead to large reduction in size of the model, leading to significant reduction in storage requirements.

5. Experiments

5.1. Experiment settings

We focus on testing Prune-m strategy coupled with the top-k (where top k gradients are communicated by each worker) and rTop-k (where a random subset of k gradients are picked from top r gradients and communicated to the

Algorithm 1 Prune-m (for Top-k /rTop-k)

```
Input: compression size $k$, number of workers W
Initialize weight of global and local models
for $epoch = 1$ to $E$ do
    for $worker = 1$ to $W$ do
        train the local copy of model with local dataset for one epoch
        calculate and update the gradients at the local model
        calculate topk (rtopk) of updated gradients
    end for
    calculate the average of compressed gradients
    if $epoch = 1$ then
        find $m$ top indices from average gradient vector
        prune the global model with the top $m$ indices
    end if
    update the global model
    update the local model based on global model
end for
```

central server) sparsification methods and observe the global model accuracy in federated learning setting. We compare our results with the top-k and rTop-k sparsification strategy on unpruned model and the baseline setting where there is no pruning or sparsification. We work with gradient compression 99% and 99.9% , and model pruning of 90% and 99%, defined at follows.

$$\text{Gradient Compression} = \frac{k}{|\theta|} \times 100 \tag{6}$$

$$\text{Model Pruning} = \frac{m}{|\theta|} \times 100 \tag{7}$$

Dataset.
MNIST database of handwritten digits (10 class)
Size = $1 \times 28 \times 28$
Total number of examples in training set = 60000
Number of workers = 10
Size of local dataset at each worker = 6000
Total number of examples in test set = 10000
Batch size for the test set = 1000
Images are distributed among workers in both iid and non-iid case.

CIFAR-10 (10 class)
Size = $3 \times 32 \times 32$
Total number of examples in training set = 50000
Number of workers = 10
Size of local dataset at each worker = 5000
Total number of examples in test set = 10000
Batch size for the test set = 1000

Model.

Table 3. Network Model.

NETWORK 1 TOTAL PAR (39760)	NETWORK 2 TOTAL PAR (62006)
FC(784,50)	CONV2D(3, 6, 5)+RELU
RELU	MAXPOOL2D(2, 2)
FC(50,10)	CONV2D(6, 16, 5)+RELU
SOFTMAX	FC(16 * 5 * 5, 120)+RELU
	FC(120, 84)+RELU
	FC(84, 10)

Hyperparameters.
m = number of parameters left in the model after pruning
k= number of gradients sent in any k-contraction gradient updating technique, like top-k and rTop-k.
r= the number of topmost gradients chosen in rTop-k, from which k gradients are randomly sent.
Learning rate = 0.1
Optimizer = Stochastic Gradient Descent (SGD)

Performance Metric. Classification accuracy of global model

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}} \times 100 \tag{8}$$

Convergence Guarantees. All models are trained for 50 epochs. Prune-m readily enjoys the same convergence guarantees as top-k, rand-k or rTop-k since it is also a k-contraction (Stich et al., 2018).

5.2. Results and Analysis

Table 4. Classification accuracy for different methods of gradient compression in the federated learning case ($W = 10$, $|\theta| = 37960$) i.i.d. case, dataset=MNIST, Network1.

COMPRESSION PRUNING	99.9% 99%	99.9% 90%	99% 90%
BASELINE	97.82	97.82	97.82
TOP-K	90.1	90.1	95.98
RTOP-K	92.44	92.44	96.36
PRUNE-M+TOP-K	92.82	92.91	96.07
PRUNE-M+RTOP-K	90.65	90.89	95.36

Prune-m with top-k. Tables 4, 5 and 6 show classifier accuracy for top-k with gradient compression 99% and 99.9% on base classifier and pruned classifier. The accuracy is surprisingly higher in prune-m case, which can be attributed to regularisation effect of removing less important parameters. The accuracies are very close to respective baseline setting

Table 5. Classification accuracy for different methods of compression in the federated learning case ($W = 10$, $|\theta| = 62006$) i.i.d. case, dataset=CIFAR10, Network2.

COMPRESSION	99.9 %	99 %
PRUNING	99%	90%
BASELINE	61.32	61.32
TOP-K	50.98	55.79
PRUNE-M+TOP-K	52.24	56.86

Table 6. Classification accuracy for different methods of compression in the federated learning case ($W = 10$, $|\theta| = 37960$) non-i.i.d. case, dataset=MNIST, Network1.

COMPRESSION	99.9 %	99 %
PRUNING	99%	90%
BASELINE	88.13	88.13
TOP-K	64.99	68.07
PRUNE-M+TOP-K	69.71	74.48

accuracies with gradient compression of 99% and model pruning of 90%.

Prune-m with rTop-k. rTop-k gives better performance than top-k on base classifier. Table 4 shows classifier accuracy for rTop-k on base classifier and pruned classifier. The performance loss under prune-m strategy is marginal. Since $r > k$, the spatial distribution of top-r crucial gradients is more wide spread than that of top-k, hence using model pruning of 90% and 99%, which proved to be suitable for top-k strategy, might lead to neglecting some crucial gradients in rtop-k strategy. Again the accuracies are very close to baseline setting accuracy of 97.82% with gradient compression of 99%.

Prune-m in heterogeneous setup. Table 6 exhibits the classifier model accuracy in heterogeneous environment where each worker has different distribution of data. There is significant reduction in model accuracy with Prune-m strategy, signifying that it is not performance optimal to prune the same set of gradients in all worker models, since data distribution can affect the spatial distribution of crucial gradients.

6. Conclusion

Latest sparsification technqiues like top-k and rTop-k make use of the skewed and sparse nature of the local gradients observed in experiments to send few crucial gradients to the global server. In this paper, we observe that the spatial distribution of crucial gradients in large neural networks is stationary, leading to large number of parameters never getting updated during the training process under these sparsification techniques. This can be exploited to prune and compress the local models at mobile edge devices which are faced with memory and network bandwidth constraints. We also studied the tradeoff between exploration and exploitation in large gradient vectors for obtaining high model accuracy, which showed that the optimal approach to send updates is neither too exploratory, nor too exploitatively biased towarads top gradients. We proposed the Prune-m alogorithm, which coupled with top-k or rTop-k sparsification, can yield large savings in terms storage and communication overheads. Numerical experiments show that our pruning approach does not damage the final accuracy of the global model under the sparsification techniques. An interesting future research direction for proposed protocol is to study model pruning strategies for non-i.i.d. case that allow better pruning than simply taking union of set of crucial gradients of workers, as this would allow us to deploy compressed models at edge devices even in heterogeneous data settings.

References

Barnes, L. P., Inan, H. A., Isik, B., and Özgür, A. rtop-k: A statistical estimation approach to distributed sgd. *ArXiv*, abs/2005.10761, 2020.

Blalock, D. W., Ortiz, J. G., Frankle, J., and Guttag, J. What is the state of neural network pruning? *ArXiv*, abs/2003.03033, 2020.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecný, J., Mazzocchi, S., McMahan, H., Overveldt, T. V., Petrou, D., Ramage, D., and Roselander, J. Towards federated learning at scale: System design. *ArXiv*, abs/1902.01046, 2019.

Chen, T., Giannakis, G. B., Sun, T., and Yin, W. Lag: Lazily aggregated gradient for communication-efficient distributed learning. In *NeurIPS*, 2018.

Hu, R., Gong, Y., and Guo, Y. Cpfed: Communication-efficient and privacy-preserving federated learning. *ArXiv*, abs/2003.13761, 2020.

Jiang, Y., Wang, S., Ko, B. J., Lee, W.-H., and Tassiulas, L. Model pruning enables efficient federated learning on edge devices. *ArXiv*, abs/1909.12326, 2019.

Kairouz, P., McMahan, H., Avent, B., Bellet, A., Bennis, M., Bhagoji, A., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z. A., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konecný, J., Korolova, A., Koushanfar, F., Koyejo, O., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F., Yu, H., and Zhao, S. Advances and open problems in federated learning. *ArXiv*, abs/1912.04977, 2019.

Konecný, J., McMahan, H., Yu, F., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. *ArXiv*, abs/1610.05492, 2016.

Li, X., xuan Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. *ArXiv*, abs/1907.02189, 2020.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, B. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *ICLR*, 2018.

McMahan, H. B., Moore, B., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., and Pedarsani, R. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *AISTATS*, 2020.

Sahu, A. K., Li, T., Sanjabi, M., Zaheer, M., Talwalkar, A., and Smith, V. On the convergence of federated optimization in heterogeneous networks. *ArXiv*, abs/1812.06127, 2018.

Stich, S. U., Cordonnier, J.-B., and Jaggi, M. Sparsified sgd with memory. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *NIPS*, pp. 4447–4458, 2018.

Suresh, A. T., Yu, F., Kumar, S., and McMahan, H. Distributed mean estimation with limited communication. In *ICML*, 2017.