Addressing the Heterogeneity of A Wide Area Network for DNNs

Hideaki Oguni
Tokyo Institute of Technology
Tokyo, Japan
oguni.h.aa@m.titech.ac.jp

Kazuyuki Shudo
Tokyo Institute of Technology
Tokyo, Japan
shudo@is.titech.ac.jp

***Abstract*—In general, deep neural networks (DNNs) achieve higher accuracy as the amount of training data increases. However, training data are often privacy sensitive, and they may not be collected. There are several methods that leave the training data decentralized in a wide area network and share models. These methods update the models locally based on stochastic gradient descent (SGD) and communicate to aggregate the models. The network bandwidth, training data, and machines are heterogeneous in a wide area network unlike distributed DNNs which use a computer cluster. Due to heterogeneity, the methods using synchronous communication, such as all-reduce SGD, are not suitable, and gossip SGD using asynchronous communication is a dominant method. In this paper, we show that when the network bandwidth is heterogeneous, conventional gossip SGD causes network congestion, and the learning efficiency is not greatly different from the case in which the network bandwidth is homogeneous.**

We show that the congestion problem can be solved by adjusting the communication frequency, that is, by training multiple times and communicating once. In many works, learning in local nodes and communicating with other nodes are alternated. Furthermore, we propose a warm-up technique to improve the learning efficiency. This proposed technique decreases the amount of communication with nodes that require a long communication time. We verify the effect of the proposed technique in experiments using CIFAR-10 and CIFAR-100.

***Index Terms*—a wide area network, gossip, DNNs**

I. Introduction

Deep neural networks (DNNs) are used as a machine learning method and have had an impact on many fields, including image recognition, speech recognition and language processing. When using DNNs, it is necessary to prepare the training data to increase accuracy. For example, data from 284335 patients were trained on in a study to predict age, gender, smoking status, systolic blood pressure and major adverse cardiac events from retinal images [1].

However, training data such as personal medical data and genetic data obtained from hospitals and images stored in smartphones are often privacy sensitive, and they may not be collected. In this case, the training data are left decentralized in a wide area network.

In the case of a wide area network, gossip SGD [2]–[4], using the asynchronous gossip communication method to share models, is the most common method.

In the case of normally distributed DNNs using a computer cluster but not a wide area network, methods using a parameter server to share the models [5]–[9] and all-reduce SGD [3], [10]–[15] using the synchronous all-reduce communication method to share the models are studied. However, neither type of method is suitable in a wide area network. Unlike the case of distributed DNNs using a computer cluster, in a wide area network, the network delay is long and the network bandwidth is narrow. Furthermore, the network bandwidth, the training data and the machines are heterogeneous. Due to this heterogeneity, communication between parameter servers and clients becomes a bottleneck. Since all-reduce SGD uses a synchronous communication method, it is difficult to absorb heterogeneity, and this method is inferior to gossip SGD using the asynchronous communication method in terms of fault tolerance.

We found that when the network bandwidth is heterogeneous, conventional gossip SGD causes network congestion. We show using experiments that when network congestion occurs, increasing wide-bandwidth nodes does not improve the learning efficiency. We show that the congestion problem arises due to the ratio of communication time to training time and can be solved by adjusting the communication frequency and decreasing the proportion of communication time. Furthermore, we propose a warm-up technique to maintain the accuracy of the models and improve learning efficiency. This proposed technique adjusts the communication destination selection probability according to the communication time.

The remainder of this paper is organized as follows: Section II introduces related work. Section III introduces conventional gossip SGD and the congestion problem in a wide area network. Section IV discusses the cause of the congestion problem and proposes a solution and a warm-up technique. Section V discusses the experiments and verifies the effect of the solution and a warm-up technique proposed in Section IV. Section VI provides conclusions and future work.

II. Related Work

Gossip learning [16] is a machine learning method in a wide area network. This method learns without moving the training data from nodes for privacy reasons, and it uses gossip as a communication method. However, the target of gossip learning is machine learning that uses on-line learning. Gossip learning does not apply to DNNs because DNNs use mini-batch learning.

Proc. 18th IEEE Consumer Communications & Networking Conference
(IEEE CCNC 2021), January 2021

Algorithm 1 Gossip SGD (pull; run on client i)

```
1: function CLIENT_LEARN
2:     Initialize:
3:         w_{0,i} := w_0, t:=0
4:     loop
5:         shuffle(X_i)
6:         for minibatch x ∈ X_i do
7:             w_{t+1,i} ← w_{t,i} − α∇F_i(w_{t,i}; x)
8:             t ← t + 1
9:             if t ≡ 0 mod T then
10:                choose j at random
11:                receive(w_j)
12:                w_{t,i} ←average(w_{t,i}, w_j)
13:            end if
14:        end for
15:    end loop
16: end function
```

There are many related works about distributed DNNs using clusters rather than a wide area network. Hop [17] and Prague [18] proposed heterogeneity-aware decentralized training. These works assume that the computation capability is heterogeneous but do not assume that the network is heterogeneous. Goyal et al. [10] proposes a method that improves the learning efficiency by using a large mini-batch size. There are several methods that use a large mini-batch size [11], [12]. Lin et al. [13] proposes a method that compresses the gradient in order to communicate without decreasing accuracy and achieves a gradient ratio of up to 600. Shi et al. [14] proposes MG-WFBP, which combines computation and communication to decrease the learning time. MG-WFBP is proposed to improve WFBP [15]. Yu et al. [9] proposes a method that is tolerant to unreliable networks. There are many other works in distributed DNNs [3]–[8].

Federated Learning [19] trains DNNs using decentralized data. This method does not move the training data, as gossip learning does. This method uses a parameter server and assumes that the performance of the server is much higher than that of the clients; therefore, this method does not assume a fully distributed environment, which is our target.

III. Congestion Problem in Gossip SGD

In this section, we first introduce gossip SGD, which is a distributed DNN method. Then, we present the congestion problem for gossip SGD in a wide area network.

A. Gossip SGD

Gossip SGD uses the asynchronous communication method gossip to share models. Gossip communicates asynchronously and autonomously to share complete information loosely.

Algorithm 1 is the pseudo-code of pull-gossip SGD. Each node uses the training data X_i of the node to update the model based on SGD. In this paper, we refer to the amount of mini-batch data trained in a single training iteration of a single node as the batch size. Each node regularly communicates asynchronously with a uniformly and randomly selected node j and takes the average of the models. The average of the models is the average of each parameter of w_i and w_j. These updates are repeated, and nodes share the model that reflects the data of whole nodes. T is the communication frequency used to share the model to other nodes. Jin et al. uses $T = 1$ [3].

Push-gossip SGD is the other gossip SGD method, but it is not suitable for use with the method proposed in Subsection IV-B. Each node uses the training data of the node to update the model. Then, each node regularly sends the model to uniformly and randomly selected nodes and takes the average of the models received during the training process.

B. Congestion in A Wide Area Network

Gossip SGD is a dominant method in a wide area network. The network bandwidth, training data, and machines are heterogeneous in a wide area network. Gossip SGD using asynchronous communication is a good match for heterogeneity.

However, we found that when the network bandwidth is heterogeneous, conventional gossip SGD causes network congestion in communicating with narrow-bandwidth nodes. Figure 1a shows the communication status of each node. A separate and continuous line corresponds to a single communication. The number of nodes is 8. The detailed experimental setting is presented in Section V. The horizontal axis represents the learning time, and the vertical axis represents the IDs of nodes. Nodes with IDs less than 6 have wide bandwidths, and the other nodes have narrow bandwidths. The model size in this experiment is 54 MiB, and the narrow bandwidth is 1.0 Gbps. Therefore, the communication time is at most:

$$\frac{54 \times 2^{20} \times 8}{1.0 \times 10^9} \simeq 0.5\,[\mathrm{s}].$$

However, in Figure 1a, there are communications with longer times than the maximum time of 0.5 seconds. This means that network congestion occurs.

Figure 1b shows a learning curve. "wide" indicates the number of wide-bandwidth nodes. As shown in Figure 1b, learning efficiency does not vary much as the number of wide bandwidth nodes increases.

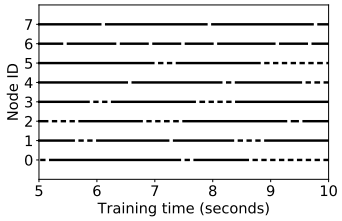
From the above results, when network congestion occurs, increasing the number of wide-bandwidth nodes does not improve the learning efficiency. This is the congestion problem of conventional gossip SGD in a wide area network that we found. When network congestion occurs, nodes act as if they are communicating synchronously with narrow-bandwidth nodes. Thus, conventional gossip SGD does not utilize asynchrony.

IV. A Warm-up Technique

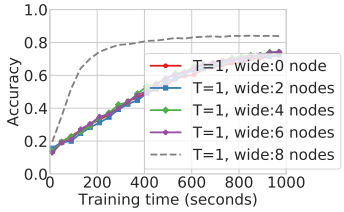
This section shows a cause and a solution of the congestion problem for gossip SGD in a wide area network. Furthermore, we propose a warm-up technique that improves learning efficiency.

A. Requirements of Communication Frequency

In gossip SGD, where bandwidth is heterogeneous in a wide area network, the communication time between wide-bandwidth nodes is long. However, the communication time between a wide-bandwidth node and a narrow-bandwidth node is short. Thus, narrow-bandwidth nodes frequently communicate with wide-bandwidth nodes, and the communication



(a) Communication status when the number of wide-bandwidth nodes is 6. Nodes with an ID less than 6 have wide bandwidths.



(b) Learning curves.

Fig. 1. Conventional gossip SGD.

time can be longer than the average communication interval. In this case, a narrow-bandwidth node has a waiting time for communication with other nodes, and network congestion occurs. In other words, if node i satisfies the following inequality, network congestion occurs.

$$\frac{1}{K-1}\sum_{j\neq i}\left(T\times\mathbb{E}\left[G_j\right]+\mathbb{E}\left[C_j\right]\right)<\mathbb{E}\left[C_i\right] \tag{1}$$

Let K, T, G, C, and $\mathbb{E}$ be the number of nodes, communication frequency, computation time of the gradient, communication time, and expected value, respectively. Here, since gossip SGD uses multiple nodes, $K>1$. Equation 1 shows that increasing $T\times\mathbb{E}\left[G_j\right]$ or T resolves network congestion.

Therefore, the requirement for all nodes to resolve network congestion in gossip SGD is:

$$T\geq\frac{(K-1)\,\mathbb{E}\left[C_i\right]-\sum_{j\neq i}\mathbb{E}\left[C_j\right]}{\sum_{j\neq i}\mathbb{E}\left[G_j\right]}.$$

B. Warm-up

We propose a warm-up technique to accelerate learning convergence. This method adjusts the communication destination selection probability according to the communication time at the beginning of learning. DNNs achieve higher accuracy as the amount of training data increases. If the selection probability is not uniformly random, the accuracy may be low.

However, training with little data can improve accuracy at the beginning of learning. Thus, the warm-up technique is effective because nodes frequently communicate with wide-bandwidth nodes to decrease the communication time.

Push-gossip SGD is not suitable for this method because the frequency of receiving models in narrow-bandwidth nodes is less than it is when the selection probability is uniformly random. On the other hand, pull-gossip SGD does not decrease the frequency of receiving models in all nodes.

We propose that each node should send the measured communication time information to each other and share it. Nodes need to share communication time information because a node cannot recognize the bandwidth of nodes that are wider than its bandwidth in its measured time. When a node sends a model, the sending node and receiving node send their communication time information. In this method, nodes cannot recognize the widest bandwidth. For example, if the bandwidth of one node is different, the nodes falsely conclude that all nodes have the same bandwidth.

We assume that the network scale is a local area network (LAN) or metropolitan area network (MAN) and that the main factor of a communication bottleneck is the performance of the nodes. When the network scale is larger than that of a MAN and the network latency is high, this method of estimating the communication time is not sufficient. We leave this issue as the future work.

Algorithm 2 is the pseudo-code of pull-gossip SGD and a warm-up technique. c_i is the communication time information, and p is the communication destination selection probability. Each node uses the training data X_i, as in Algorithm 1. Each node selects a communication destination based on the selection probability p; the selection is not uniformly random.

Algorithm 2 Gossip SGD (pull) + warm-up (run on client i)

function CLIENT_LEARN
  Initialize:
    $w_{0,i}:=w_0, t:=0, c_i:=\{\mathrm{INF},\mathrm{INF},...,\mathrm{INF}\}$
  loop
    shuffle(X_i)
    for minibatch $x\in X_i$ **do**
      $w_{t+1,i}\leftarrow w_{t,i}-\alpha\nabla F_i(w_{t,i};x)$
      $t\leftarrow t+1$
      if $t\equiv 0 \bmod T$ **then**
        $p:=\left\{\frac{1}{c_{i,k}}\right\}_{k=1}^{K}$
        $p_i\leftarrow 0$
        choose j on p
        send(c_i)
        receive(c_j, w_j)
        c :=measured_communication_time()
        $c_{i,j}\leftarrow$UPDATE_TIME($c_{i,j}, c$)
        $c_i\leftarrow$UPDATE_TIMES(c_i, c_j)
        $w_{t,i}\leftarrow$average($w_{t,i}, w_j$)
      end if
    end for
  end loop
end function
function UPDATE_TIME(c_i, c_j)
  if $(1-\mathrm{threshold})\times \mathrm{c_i} > \mathrm{c_j}$ **then**
    $c_i\leftarrow c_j$
  else if $(1+\mathrm{threshold})\times \mathrm{c_i} < \mathrm{c_j}$ **then**
    $c_i\leftarrow\frac{c_i+c_j}{2}$
  end if
  return c_i
end function
function UPDATE_TIMES(c_i, c_j)
  for $k\in K$ **do**
    $c_{i,k}\leftarrow$UPDATE_TIME($c_{i,k}, c_{j,k}$)
  end for
  return c_i
end function

TABLE I
EXPERIMENTAL ENVIRONMENT.

OS	Ubuntu 16.04.2 LTS
Kernel	Linux 4.4.0-79-generic
CPU	Intel Xeon E5-2698 v4
GPU	Tesla P100-PCIE-16GB

TABLE II
ACCURACY WHEN THE BATCH SIZE VARIES.

Batch size	Accuracy
64	87.1%
128	87.5%
256	87.4%
512	87.7%

(a) The number of wide-bandwidth nodes is 0.

Batch size	Accuracy
64	87.0%
128	87.1%
256	87.5%
512	87.5%

(b) The number of wide-bandwidth nodes is 8.

p_j is the reciprocal of the communication time $c_{i,j}$ with node j, and $p_i \leftarrow 0$ because a node does not communicate with its own node.

The communication time information c_i of node i uses the measured communication time c and received communication time information c_j of node j to update c_i. If $c_{j,k}$ is less than the threshold of $c_{i,k}$, $c_{i,k}$ is updated to $c_{j,k}$. Otherwise, if $c_{i,k}$ and $c_{j,k}$ are close, $c_{i,k}$ is updated to the average of $c_{i,k}$ and $c_{j,k}$. This update is $\Theta(K)$ for the number of nodes K, and it is sufficiently fast. c_j also uses c_i to update c_j.

V. EXPERIMENTS

In this section, we first confirm that increasing the communication frequency T resolves network congestion. Then, we present the evaluation of the warm-up technique proposed in Subsection IV-B.

A. Experimental Setup

We use the CIFAR-10 and CIFAR-100 datasets [20] and VGG16 [21]. Both datasets have 50000 training data points and 10000 testing data points. The number of nodes is 8 in all experiments. The accuracy is the average correct answer rate for the test data for each node. Table I shows the experimental environment.

B. Preparation

In the following experiments, we simulate communication but run the training on real GPUs. We use Chainer [22] to compute the gradients of the loss function. The computation time is the average of the values measured 10 times to account for the variation in the computation time.

We made a VGG16 model train using a single node and saved the model to a file in `npz` format every epoch. The `npz` format is often used to save numpy arrays. We saved each model for 10 epochs, and the size of all models was 54 MiB. The wide bandwidth is 10 Gbps, and the narrow bandwidth is 1.0 Gbps. The latency is fixed at 5.0 ms in all nodes.

C. CIFAR-10

The learning rate is fixed at 0.3 in the CIFAR-10 experiments. We first experiment with the batch size. Table II shows the highest accuracy, and Figure 2 shows the learning curves. "Wide" represents the number of wide-bandwidth nodes. Figure 2a shows that the learning efficiency with batch sizes of 256 and 512 is approximately the same, and Figure 2b shows that the learning efficiency with a batch size of 256 is the best. The accuracy with batch sizes of 256 and 512 is approximately the same, so the batch size is fixed at 256 in subsequent experiments.

TABLE III
ACCURACY WHEN THE COMMUNICATION FREQUENCY VARIES.

Communication frequency T	Accuracy
1	87.4%
4	87.5%
8	87.4%
16	86.9%
32	86.2%

TABLE IV
ACCURACY COMPARISON BETWEEN WARM-UP AND CONVENTIONAL GOSSIP.

Wide	Accuracy	
	Warm-up	Conventional
2 nodes	87.0%	87.3%
4 nodes	87.3%	87.0%
6 nodes	86.9%	86.7%

(a) CIFAR-10.

Wide	Accuracy	
	Warm-up	Conventional
2 nodes	60.1%	60.3%
4 nodes	60.3%	60.2%
6 nodes	59.7%	59.6%

(b) CIFAR-100.

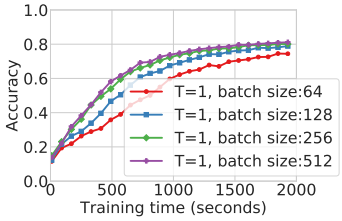
Next, we vary the communication frequency T to confirm that a large T resolves network congestion. Figure 3a shows the communication status of $T = 4$. Compared to Figure 1a, the communication time is short on the whole, and network congestion does not occur. Figure 3b shows the learning curves. In contrast to Figure 1a, the wider the bandwidth nodes are, the better the learning efficiency.

We adjust the communication frequency and investigate the best conditions for learning efficiency. In this paper, we regard the accuracy as low if it falls by 1% because each experiment is conducted only once. When the bandwidth of all nodes is narrow, Figure III shows the highest accuracy, and Figure 4 shows the learning curves. When $T = 32$, the accuracy is low. Moreover, the larger T is, the better the learning efficiency. Therefore, $T = 16$ is the optimal communication frequency. We also experiment with increasing numbers of wide-bandwidth nodes.

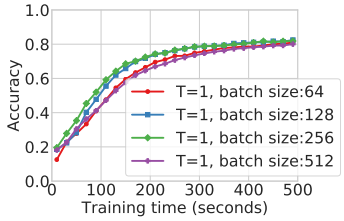
We now verify the effect of the warm-up technique. Warm-up is conducted in the initial 1500 iterations. Table IVa shows the highest accuracy. The accuracy of the warm-up and conventional methods is approximately the same. Figures 5a-7a show the learning curves. We see that the warm-up technique improves the learning efficiency in all three experiments. Furthermore, the effect of the warm-up technique is remarkable when many nodes have wide bandwidths.

D. CIFAR-100

The learning rate is fixed at 0.1 in the CIFAR-100 experiments. We adjust the batch size and communication frequency and then verify the effect of the warm-up technique. Warm-up is conducted in the initial 5000 iterations. The results are

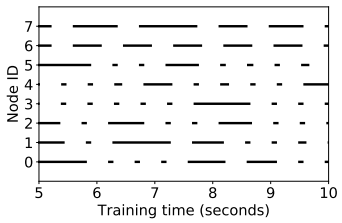


(a) The number of wide-bandwidth nodes is 0.

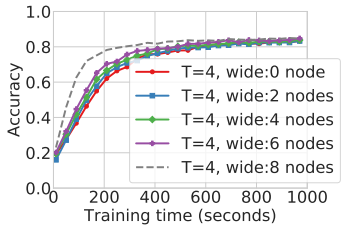


(b) The number of wide-bandwidth nodes is 8.

Fig. 2. Learning curves when the batch size varies.



(a) Communication status when the number of wide-bandwidth nodes is 6. Nodes with IDs less than 6 have wide bandwidths.



(b) Learning curves.

Fig. 3. Gossip SGD when the communication frequency $T = 4$.

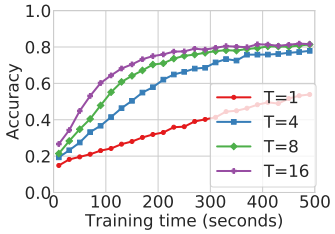


Fig. 4. Learning curves when the communication frequency varies.

similar to those of CIFAR-10. Table IVb shows the highest accuracy. The accuracy of the warm-up and conventional methods is approximately the same. Figures 5b-7b show the learning curves. We see that the warm-up technique improves the learning efficiency in all three experiments, and the effect of the warm-up technique is remarkable when many nodes have wide bandwidths.

VI. Conclusion

In this paper, we showed that when the network bandwidth is heterogeneous, conventional gossip SGD causes network congestion. Moreover, increasing the number of wide-bandwidth nodes does not improve the learning efficiency. We showed that the congestion problem can be solved by adjusting the communication frequency. Furthermore, we proposed a warm-up technique to improve the learning efficiency by adjusting the communication destination selection probability. Our experiments show that when network congestion is resolved by decreasing the communication frequency, increasing the number of wide-bandwidth nodes improves the learning efficiency. We verified that the warm-up technique improves the learning efficiency while maintaining accuracy.

A warm-up technique for cases in which the data distribution and computation capability are heterogeneous is left for future work. One solution is to share not only the communication time but also the computation time.

Tolerance to adversarial nodes is another topic for future work. Since we do not assume such nodes in this paper, the learning efficiency can be unsatisfactory. When there are adversarial nodes, the method of sharing and updating models may need to be changed.

Acknowledgment

This work was supported by New Energy and Industrial Technology Development Organization (NEDO).

References

[1] Ryan Poplin, Avinash V Varadarajan, Katy Blumer, Yun Liu, Michael V McConnell, Greg S Corrado, Lily Peng, and Dale R Webster. Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning. *Nature Biomedical Engineering*, 2(3):158, 2018.

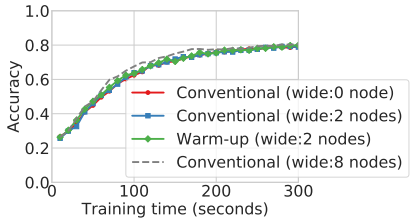
[2] Michael Blot, David Picard, Matthieu Cord, and Nicolas Thome. Gossip training for deep learning. *arXiv preprint arXiv:1611.09726*, 2016.

[3] Peter H Jin, Qiaochu Yuan, Forrest Iandola, and Kurt Keutzer. How to scale distributed deep learning? *arXiv preprint arXiv:1611.04581*, 2016.

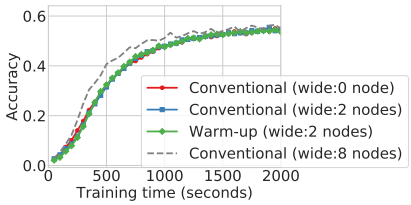
[4] Jeff Daily, Abhinav Vishnu, Charles Siegel, Thomas Warfel, and Vinay Amatya. Gossipgrad: Scalable deep learning using gossip communication based asynchronous gradient descent. *arXiv preprint arXiv:1803.05880*, 2018.

[5] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, pages 583–598, 2014.

[6] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, and Kurt Keutzer. Firecaffe: Near-linear acceleration of deep neural network training on compute clusters. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

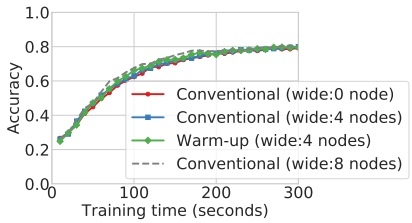


(a) CIFAR-10.

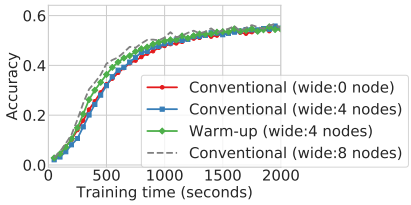


(b) CIFAR-100.

Fig. 5. Learning curve comparison between the warm-up and conventional gossip (wide: 2 nodes).

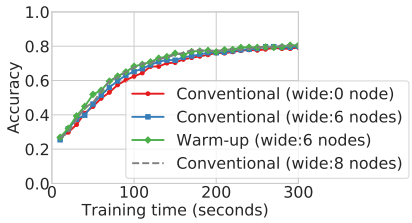


(a) CIFAR-10.

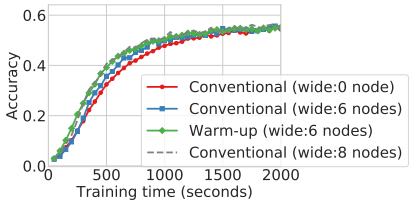


(b) CIFAR-100.

Fig. 6. Learning curve comparison between the warm-up and conventional gossip (wide: 4 nodes).



(a) CIFAR-10.



(b) CIFAR-100.

Fig. 7. Learning curve comparison between the warm-up and conventional gossip (wide: 6 nodes).

[7] Henggang Cui, Hao Zhang, Gregory R. Ganger, Phillip B. Gibbons, and Eric P. Xing. Geeps: scalable deep learning on distributed gpus with a gpu-specialized parameter server. In *Proceedings of the Eleventh European Conference on Computer Systems, EuroSys 2016, London, United Kingdom, April 18-21, 2016*, pages 4:1–4:16, 2016.

[8] Hanjoo Kim, Jaehong Park, Jaehee Jang, and Sungroh Yoon. Deepspark: Spark-based deep learning supporting asynchronous updates and caffe compatibility. *arXiv preprint arXiv:1602.08191*, 3, 2016.

[9] Chen Yu, Hanlin Tang, Cédric Renggli, Simon Kassing, Ankit Singla, Dan Alistarh, Ce Zhang, and Ji Liu. Distributed learning over unreliable networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 7202–7212, 2019.

[10] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[11] Takuya Akiba, Shuji Suzuki, and Keisuke Fukuda. Extremely large minibatch SGD: Training ResNet-50 on ImageNet in 15 minutes. *arXiv preprint arXiv:1711.04325*, 2017.

[12] Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. ImageNet training in minutes. In *Proceedings of the 47th International Conference on Parallel Processing*. ACM, 2018.

[13] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018.

[14] Shaohuai Shi, Xiaowen Chu, and Bo Li. MG-WFBP: Efficient data communication for distributed synchronous SGD algorithms. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 172–180. IEEE, 2019.

[15] Ammar Ahmad Awan, Khaled Hamidouche, Jahanzeb Maqbool Hashmi, and Dhabaleswar K Panda. S-caffe: Co-designing mpi runtimes and caffe for scalable deep learning on modern gpu clusters. In *Acm Sigplan Notices*, volume 52, pages 193–205. ACM, 2017.

[16] Róbert Ormándi, István Hegedűs, and Márk Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

[17] Qinyi Luo, Jinkun Lin, Youwei Zhuo, and Xuehai Qian. Hop: Heterogeneity-aware decentralized training. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 893–907, 2019.

[18] Qinyi Luo, Jiaao He, Youwei Zhuo, and Xuehai Qian. Prague: High-performance heterogeneity-aware asynchronous decentralized training. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 401–416, 2020.

[19] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pages 1273–1282, 2017.

[20] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[22] Seiya Tokui, Ryosuke Okuta, Takuya Akiba, Yusuke Niitani, Toru Ogawa, Shunta Saito, Shuji Suzuki, Kota Uenishi, Brian Vogel, and Hiroyuki Yamazaki Vincent. Chainer: A deep learning framework for accelerating the research cycle. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2002–2011. ACM, 2019.