doi:10.3772/j.issn.1006-6748.2022.02.007

Pseudo-label based semi-supervised learning in the distributed machine learning framework①

WANG Xiaoxi (王晓曦)*, WU Wenjun②*, YANG Feng*, SI Pengbo*, ZHANG Xuanyi**, ZHANG Yanhua*
(*Faculty of Information Technology, Beijing University of Technology, Beijing 100124, P. R. China)
(**Beijing Capital International Airport Co., Ltd., Beijing 101317, P. R. China)

Abstract

With the emergence of various intelligent applications, machine learning technologies face lots of challenges including large-scale models, application oriented real-time dataset and limited capabilities of nodes in practice. Therefore, distributed machine learning (DML) and semi-supervised learning methods which help solve these problems have been addressed in both academia and industry. In this paper, the semi-supervised learning method and the data parallelism DML framework are combined. The pseudo-label based local loss function for each distributed node is studied, and the stochastic gradient descent (SGD) based distributed parameter update principle is derived. A demo that implements the pseudo-label based semi-supervised learning in the DML framework is conducted, and the CIFAR-10 dataset for target classification is used to evaluate the performance. Experimental results confirm the convergence and the accuracy of the model using the pseudo-label based semi-supervised learning in the DML framework. Given the proportion of the pseudo-label dataset is 20%, the accuracy of the model is over 90% when the value of local parameter update steps between two global aggregations is less than 5. Besides, fixing the global aggregations interval to 3, the model converges with acceptable performance degradation when the proportion of the pseudo-label dataset varies from 20% to 80%.

Key words: distributed machine learning (DML), semi-supervised, deep neural network (DNN)

0 Introduction

Recently, the rapid growth of emergent applications including unmanned driving, face recognition and automatic navigation has greatly promoted the development of artificial intelligent (AI) technologies. However, in some of the scenarios, such as unmanned aerial vehicle (UAV) networks[1] and Internet of vehicles (IoV)[2], the implementation of AI technologies faces challenges aroused by the limited batteries, low computing capability as well as data privacy. Moreover, in most of the practical cases, the performance of AI technologies is limited by the size of training sample set and the accuracy of the labels. To address the above problems, distributed machine learning (DML)[3] and semi-supervised learning[4] has attracted enormous attentions.

DML is a distributed collaboration architecture for multiple worker nodes to train a machine learning (ML) model together[3]. Generally, there are two basic parallelism modes for DML, which are model parallelism and data parallelism. In the model parallelism mode, the ML model is partitioned among workers and each worker updates part of the parameters using the entire dataset[5]. In the data parallelism mode, each worker has a local copy of the complete ML model and updates model parameters based on their local data[3,6]. Nowadays, the data parallelism has been more widely adopted than the model parallelism, given that most ML models can be entirely stored in the memory of modern GPUs. Since workers do not need to send the raw data to a central node, the data privacy issue is well solved.

As for the aggregation process of DML, both the synchronous mode and the asynchronous mode can be used in the communication between workers and the aggregator[7]. In the synchronous communication, all the workers should stop at the overall ‘barrier synchronous’ and wait for other workers to finish the local

① Supported by the National Key R&D Program of China (No. 2020YFC1807904), the National Natural Science Foundation of Beijing Municipality (No. L192002) and the National Natural Science Foundation of China (No. U1633115).

② To whom correspondence should be addressed. E-mail: wenjunwu@bjut.edu.cn.

Received on Mar. 18, 2021

training before the barrier. While using the asynchronous communication, such as HogWild![8] and Cyclades[9], all the workers can send their parameters or models to the aggregator when they accomplish several local training. Obviously, the synchronous mode wastes the waiting time, but the aggregation algorithm is simple. Meanwhile, the asynchronous mode can fully utilize the time. However, due to the different computation capabilities of different workers, the poor workers may slow down the convergence rate and become a drag of the whole model.

Semi-supervised learning is a class of methods which can train deep neural networks (DNN) by using both labeled and unlabeled data. With these methods, the problem of lacking labeled data in some of the real-time application scenarios can be overcome. One of the early methods of training the DNN based on labeled and unlabeled data was studied [10]. Ref. [11] proposed a semi-supervised deep learning method for hyperspectral image classification which uses limited labeled data and unlabeled data to train a DNN. In the research area of modulation classification, combining handcrafted feature with deep learning, Ref. [12] proposed a few-shot modulation classification method based on feature dimension reduction and pseudo-label training.

Although DML and semi-supervised learning are widely used in the areas such as image classification[13], face recognition[14], natural language processing[15] etc., the implementation combining these two technologies together has not been well studied. For some of the scenarios, especially some AI applications with real-time collected unlabeled data using devices with limited capability such as the UAV based emergency rescue and real time high definition mapping in IoV, the combination of DML with semi-supervised learning is of great necessary.

In this paper, a data parallelism architecture enabling the pseudo-label based semi-supervised learning in DML is proposed. Then the cross entropy based local loss function with pseudo-label at each worker is given and the learning problem is formulated. The stochastic gradient descent (SGD) is adopted in the training process and the corresponding local parameter updating equation is derived. A demo that implements the pseudo-label based semi-supervised learning in the DML framework is conducted and the CIFAR-10 data set for target classification is used to evaluate the performance. Given the proportion of the pseudo-label dataset is 20%, results show that the model converges when the local update steps between every two global aggregation is less than 5. Results also confirm that the proportion of the pseudo-label dataset affects the convergence rate and the accuracy.

1 Distributed semi-supervised method

1.1 Architecture

A typical data parallelism architecture enabling the pseudo-label based semi-supervised learning in DML is considered and shown in Fig. 1. The raw data is locally stored at N worker nodes, each of which trains a complete machine learning model (i.e. DNN) by using the local data. The local dataset of each worker consists of two parts, i.e., the labeled dataset and unlabeled dataset. Moreover, it is necessary to assume that all the workers collaborate in a synchronous way and a parameter server implements the parameter aggregation process.

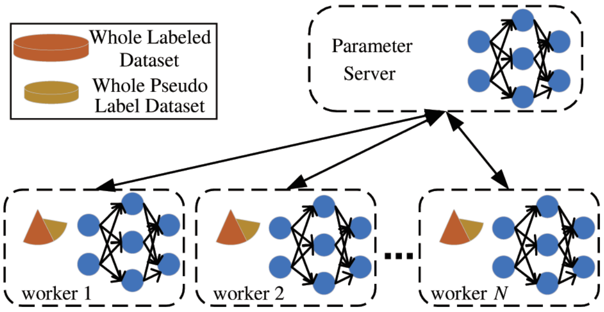


Fig. 1 Architecture

1.2 Loss function of the distributed semi-supervised learning

In the data parallelism distributed framework, the local loss function of the i-th worker is given as

$$L_i(\boldsymbol{w}) \triangleq \frac{1}{|D_i|}\sum_{j\in D_i}\sum_{k=1}^{C} l(\boldsymbol{w}, y_k^j, f_k^j) \qquad (1)$$

where D_i is the local data set of the i-th worker, C is the number of labels, $l(\boldsymbol{w}, y_k^j, f_k^j)$ is the loss function of the k-th label of the j-th data sample, $\boldsymbol{w}$ is the parameter vector of the model, $\boldsymbol{f}_k^j$ is the output units of the k-th label of the j-th sample and $\boldsymbol{y}_k^j$ is the label of it. And the global DML loss function is given as

$$L_{\mathrm{DML}}(\boldsymbol{w}) = \sum_{i=1}^{N}\frac{|D_i|\, L_i(\boldsymbol{w})}{|D|} \qquad (2)$$

where $D \triangleq \cup_{i=1}^{N} D_i$[3].

According to the pseudo-label based semi-supervised learning proposed in Ref. [10], the overall loss function with unlabeled dataset is

$$L(\boldsymbol{w}) = \frac{1}{n}\sum_{j=1}^{n}\sum_{k=1}^{C} l(\boldsymbol{w}, y_k^j, f_k^j) + \alpha\frac{1}{n'}\sum_{j=1}^{n'}\sum_{k=1}^{C} l(\boldsymbol{w}, y_k'^j, f_k'^j) \qquad (3)$$

where n is the number of the labeled data, n' is the number of the unlabeled data, α is a balance coefficient, f'^{j}_{k} is the output units of the k-th pseudo label of the j-th unlabeled sample, and the class which has maximum predicted probability is picked up as the pseudo-label y'^{j}_{k} for j-th unlabeled sample, and y'^{j}_{k} is given by

$$y'^{j}_{k} = \begin{cases} 1 & k = \mathrm{argmax}_{k'} f_{k'}(\boldsymbol{w}, \boldsymbol{x}^{j}) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\boldsymbol{x}^{j}$ is the input vector according to the j-th unlabeled sample.

The labeled and unlabeled dataset at worker i are denoted by D_{i1} and D_{i2} respectively. So the local dataset at node i is $D_i = D_{i1} \cup D_{i2}$. Combining Eqs(1) and (3), the local loss function of the i-th worker with pseudo label dataset can be expressed as

$$L_i(\boldsymbol{w}) = \frac{1}{|D_i|}\Big[\sum_{j\in D_{i1}}\sum_{k=1}^{C} l(\boldsymbol{w}, y^{j}_{k}, f^{j}_{k}) + \sum_{j\in D_{i2}}\sum_{k=1}^{C} l(\boldsymbol{w}, y'^{j}_{k}, f'^{j}_{k})\Big] \tag{5}$$

Taking Eq. (5) into Eq. (2), the overall loss function of the distributed semi-supervised learning can be expressed as

$$L(\boldsymbol{w}) = \frac{\sum_{i=1}^{N}\Big[\sum_{j\in D_{i1}}\sum_{k=1}^{C} l(\boldsymbol{w}, y^{j}_{k}, f^{j}_{k}) + \sum_{j\in D_{i2}}\sum_{k=1}^{C} l(\boldsymbol{w}, y'^{j}_{k}, f'^{j}_{k})\Big]}{|D|} \tag{6}$$

More specifically, the cross entropy is chosen as the loss function, so the item of $l(\boldsymbol{w}, y^{j}_{k}, f^{j}_{k})$ in Eq. (6) can be expressed as

$$l(\boldsymbol{w}, y^{j}_{k}, f^{j}_{k}) = -y^{j}_{k}\log f^{j}_{k}(\boldsymbol{w}) - (1 - y^{j}_{k})\log(1 - f^{j}_{k}(\boldsymbol{w})) \tag{7}$$

Besides, the soft-max function is adopted as the output unit in the neural network, and for the j-th sample,

$$f^{j}_{k} = P(y^{j}_{k} \mid \boldsymbol{z} \mid^{j}) = \frac{\exp(z^{j}_{k})}{\sum_{c=1}^{C}\exp(z^{j}_{c})} \tag{8}$$

where $\boldsymbol{z}^{j} = [z^{j}_{1}, z^{j}_{2}, \cdots, z^{j}_{C}]$ is the input vector of the output unit layer of the neural network.

2 SGD-based training process

The learning problem is to find

$$\boldsymbol{w}^{*} = \mathrm{argmin} L(\boldsymbol{w}) \tag{9}$$

When the gradient-descent method is used, the process of the solving Eq. (9) can be expressed as

$$\boldsymbol{w}(t) = \boldsymbol{w}(t-1) - \eta\nabla L(\boldsymbol{w}(t-1)) \tag{10}$$

where η is the learning rate.

Taking Eq. (2) into the derivative of $L(\boldsymbol{w})$, Eq. (11) can be obtained.

$$\nabla L(\boldsymbol{w}) = \nabla\frac{\sum_{i=1}^{N}|D_i| L_i(\boldsymbol{w})}{|D|} = \frac{\sum_{i=1}^{N}|D_i| \nabla L_i(\boldsymbol{w})}{|D|} \tag{11}$$

thus

$$\boldsymbol{w}(t) = \boldsymbol{w}(t-1) - \eta\frac{\sum_{i=1}^{N}|D_i| \nabla L_i(\boldsymbol{w}(t-1))}{|D|} = \frac{\sum_{i=1}^{N}|D_i| [\boldsymbol{w}(t-1) - \eta\nabla L_i(\boldsymbol{w}(t-1))]}{|D|} \tag{12}$$

As for the local learning problem

$$\boldsymbol{w}_i^{*} = \mathrm{argmin} L_i(\boldsymbol{w}_i) \tag{13}$$

According to the gradient-descent-based method, the local parameter vector $w_i(t)$ is updated as

$$\boldsymbol{w}_i(t) = \boldsymbol{w}'_i(t-1) - \eta\nabla L_i(\boldsymbol{w}'_i(t-1)) \tag{14}$$

It is necessary to note that since the local data set $D_i \neq D_j$ if $i \neq j$, $\boldsymbol{w}_i(t) \neq \boldsymbol{w}_j(t) \neq \boldsymbol{w}(t)$ and generally $\boldsymbol{w}'_i(t) \neq \boldsymbol{w}_i(t)$ if the aggregation is performed at time step t. However, it is an acceptable error of the DML framework compared with the centralize training since the convergence of DML with this error is proofed in Ref. [6]. Therefore, taking Eq. (13) into Eq. (12), the global parameter vector can be updated as

$$\boldsymbol{w}(t) \triangleq \frac{\sum_{i=1}^{N}|D_i| [\boldsymbol{w}'_i(t-1) - \eta\nabla L_i(\boldsymbol{w}'_i(t-1))]}{|D|} = \frac{\sum_{i=1}^{N}|D_i| \boldsymbol{w}_i(t)}{|D|} \tag{15}$$

Obviously, how to use the local data to calculate Eq. (14) and to update the local parameter vector in practice is crucial.

In this work, the stochastic gradient descent (SGD)[16] method is adopted for the local training in each worker. In each local update step of the SGD method, the gradient of the loss function is computed based on a randomly selected subset of the samples, which is referred to as a mini-batch, rather than the whole sample dataset. So the mini-batch chosen at local training step t at the i-th worker can be defined as $S_{it} \subset D_i$, which includes labeled and unlabeled samples. Then $S_{i1t} \subset D_{i1}$ and $S_{i2t} \subset D_{i2}$ are the sets of labeled and unlabeled samples in S_{it} respectively. What's more, the proportion of the pseudo-label samples in the whole mini-batch can be defined as $\mu = |S_{i2t}| / |S_{it}|$.

Based on the above definitions and the local loss function given in Eq. (5), the local loss function of the i-th worker at time step t in the SDG based training process can be written as

$$L_i(\boldsymbol{w}_i(t)) = \frac{1}{|S_{i1t}|}\sum_{j\in S_{i1t}}\sum_{k=1}^{C} l(\boldsymbol{w}_i(t), y_k^j, f_k^j) + \frac{1}{|S_{i2t}|}\sum_{j\in S_{i2t}}\sum_{k=1}^{C} l(\boldsymbol{w}_i(t), y'^j_k, f'^j_k) \tag{16}$$

Combining Eq. (7) and Eq. (16) and considering that f_k^j is the function of $\boldsymbol{w}_i(t)$, the gradient of $L_i(\boldsymbol{w}_i(t))$ is derived as

$$\begin{aligned}
&\nabla L_i(\boldsymbol{w}_i(t)) \\
&= \frac{1}{|S_{i1t}|}\sum_{j\in S_{i1t}}\sum_{k=1}^{C} \nabla l(\boldsymbol{w}_i(t), y_k^j, f_k^j) \\
&\quad + \frac{1}{|S_{i2t}|}\sum_{j\in S_{i2t}}\sum_{k=1}^{C} \nabla l(\boldsymbol{w}_i(t), y'^j_k, f'^j_k) \\
&= \frac{1}{|S_{i1t}|}\sum_{j\in S_{i1t}}\sum_{k=1}^{C} \frac{\partial l(\boldsymbol{w}_i(t), y_k^j, f_k^j)}{\partial f_k^j}\frac{\partial f_k^j}{\partial \boldsymbol{w}_i(t)} \\
&\quad + \frac{1}{|S_{i2t}|}\sum_{j\in S_{i2t}}\sum_{k=1}^{C} \frac{\partial l(\boldsymbol{w}_i(t), y'^j_k, f'^j_k)}{\partial f'^j_k}\frac{\partial f'^j_k}{\partial w_i(t)} \\
&= \frac{1}{|S_{i1t}|}\sum_{j\in S_{i1t}}\sum_{k=1}^{C} \left[\left(-\frac{y_k^j}{f_k^j} + \frac{1-y_k^j}{1-f_k^j}\right)\frac{\partial f_k^j}{\partial \boldsymbol{w}_i(t)}\right] \\
&\quad + \frac{1}{|S_{i2t}|}\sum_{j\in S_{i2t}}\sum_{k=1}^{C} \left[\left(-\frac{y'^j_k}{f'^j_k} + \frac{1-y'^j_k}{1-f'^j_k}\right)\frac{\partial f'^j_k}{\partial \boldsymbol{w}_i(t)}\right]
\end{aligned} \tag{17}$$

Taking Eq. (17) into Eq. (14), the whole local training process can be conducted.

The distributed training process of this model can be concluded to three parts: (1) the labeled data set and pseudo-label based data set are prepared in each worker; (2) each worker trains their local model based on Eqs(13) and (16), and this process is referred to as the local update; (3) after τ local update steps, a global aggregation is performed at the aggregator to update the global parameter according to Eq. (14).

The complete procedure at the aggregator and each edge node are presented in Algorithm 1 and 2, respectively. To make the description clearly, the index of the global aggregation is denoted by *global _ step*.

Algorithm 1 Training procedure at the aggregator

```
Initialize $t \leftarrow 0$, global _ step $\leftarrow 0$
Initialize $w(0)$ as a random vector and send it to all workers
Repeat:
  If $t \% \tau = 0$
    Receive $w_i(t)$ from each worker $i$
    Compute $w(t) = \frac{\sum_{i=1}^{N} |D_i| w_i(t)}{|D|}$
    Broadcast $w(t)$ to all the workers
    global _ step $\leftarrow$ global _ step + 1
  Until the model converges
Set STOP flag and send it o all the workers
```

Algorithm 2 Training procedure at the *i*-th worker

```
Initialize $t \rightarrow 0$
Repeat:
  Select a mini-batch (including labeled data and unlabeled data) from the local dataset of the $i$-th worker
  Obtain the pseudo label following
    $y'^j_k = \begin{cases} 1 & k = \mathrm{argmax}_{k'} f_{k'}(w, x^j) \\ 0 & \text{otherwise} \end{cases}$
  Receive $w(t)$ from the aggregator, set $w'_i(t) \leftarrow w(t)$
  for $\mu = 1, 2, \cdots, \tau$ do
    $t \leftarrow t + 1$
    Compute:
    $w_i(t) = w'_i(t-1)$
      $- \eta \frac{1}{|S_{i1t}|}\sum_{j\in S_{i1t}}\sum_{k=1}^{C}\left[\left(-\frac{y_k^j}{f_k^j} + \frac{1-y_k^j}{1-f_k^j}\right)\frac{\partial f_k^j}{\partial w'_i(t-1)}\right]$
      $- \eta \frac{1}{|S_{i2t}|}\sum_{j\in S_{i2t}}\sum_{k=1}^{C}\left[\left(-\frac{y'^j_k}{f'^j_k} + \frac{1-y'^j_k}{1-f'^j_k}\right)\frac{\partial f'^j_k}{\partial w'_i(t-1)}\right]$
    if $\mu < \tau$ then
      $w'_i(t) \leftarrow w_i(t)$
    else
      Send $w_i(t)$ to the aggregator
    end if
  end for
Until STOP flag is received.
```

3 Experimentation

3.1 Hardware environment

A DML framework that consists of three workers (worker0, worker1 and worker2) and a parameter server (ps0) is adopted in the experiment. The experiment demo is conducted on three laptops, which are in a local area network and connected by a router, and ps0 and worker0 are deployed on the same laptop. The configuration parameters of the three PCs are as follows.

(1) worker0 and ps0 PC. Processor: AMD Ryzen 74 800 H with Radeon Graphics 2.90 GHz. Memory: RAM 32.00 GB. System type: 64-bit operating system, based on the X64 processor. Operating system version: Windows 10.

(2) worker1 PC. Processor: Intel (R) Core (TM) i7-9750H CPU @2.60 GHz 2.59 GHz. Memory: RAM 16.00 GB. System type: 64-bit operating system, based on the X64 processor. Operating system version: Windows 10.

(3) worker2 PC. Processor: Intel (R) Core (TM) i7-9750H CPU @2.60 GHz 2.59 GHz. Memory: RAM 32.00 GB. System type: 64-bit operating system, based on the X64 processor. Operating system version: Windows 10.

3.2 Software environment

The deep neural network of the target classification model is GoogLeNet[17], which is a convolution neural network considering the local sparsity of the model. It consists of 27 layers including convolution layers, max pool layers, inception structure, dropout layers, linear layers and softmax layers. At the end of the neural network, the fully connection layer is replaced by an average pooling layer, but the use of dropout remained essential.

The DML model is conducted based on the framework of distributed TensorFlow[18], and Python 3.7 and TensorFlow-gpu 1.13.1 with CUDA 10.0 and cuDNN 7.5 are used. Worker0, which is in charge of initializing and restoring the model, is the chief worker in the TensorFlow cluster, and others should wait for the chief worker finishing his initialization and then start their training. Since the value of local update steps τ between every two global aggregations affects the convergence of the training process, the results of different values of τ are evaluated under the condition of $\mu = 20\%$.

3.3 Dataset

The dataset of CIFAR-10 is used, which includes 60 000 color images (50 000 for training and 10 000 for testing) of 10 different types of classifications[19]. In this experiment, the whole dataset is randomly divided into three parts of 10 000 samples, 20 000 samples and 20 000 samples which are the local datasets of worker0, worker1 and worker2, respectively. Thus each worker has the uniform (but not full) information. In the training process, the size of mini-batch is set as 20 for all the workers.

The proportion of the pseudo-label dataset is another factor that affects the convergence performance, thus different value of μ are also evaluated when the value of τ is 3. Since all the samples in CIFAR-10 are labeled originally, some samples are regarded as the unlabeled data and their labels are ignored, and pseudo labels are sought for them following Eq. (4) in the training process.

4 Results and analysis

4.1 Performance evaluation of different values of τ

In the first part of the experiments, the value of μ is fixed as 20% and the value of τ is varied from 3 to 7.

The training results in Fig. 2 and Fig. 3 show that when the value of τ is not greater than 5, the target classification model is converged. But for different value of τ, the convergence performance is different. When $\tau = 3$, the value of loss is close to 0.25 and the accuracy is 97.7% after 12 000 *global_step* (36 000 local update steps). When $\tau = 5$, the model needs much more time to get an acceptable performance. It converges after about 220 000 *global_step* (1 100 000 local update steps), and the final values of loss and accuracy are about 0.48 and 96%, respectively. However, when the value of τ is greater than 5, taking $\tau = 7$ as the example, the target classification model cannot converge. The value of loss drops down at the first, and then rises again after several *global_step*. Meanwhile, the accuracy rises to about 56% and then falls. This is because that when τ is too large, the local gradient may deviate too much from the global gradient, resulting in the poor convergence.

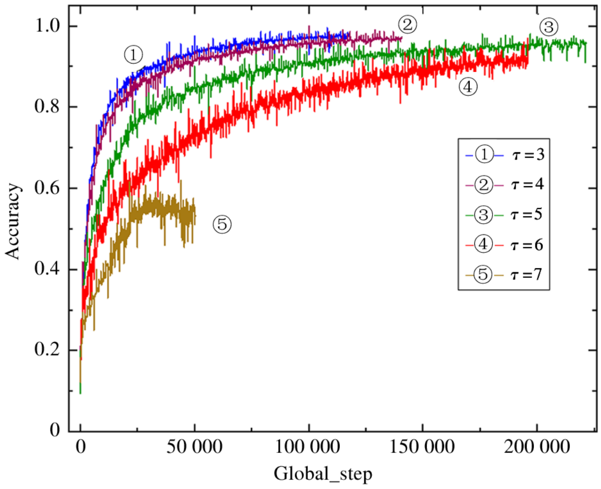


Fig. 2 Training results of accuracy ($\mu = 20\%$)

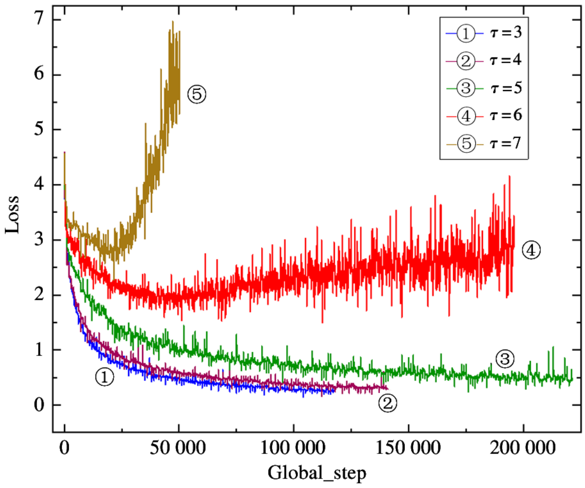


Fig. 3 Training results of loss ($\mu = 20\%$)

The test results in Fig. 4 and Fig. 5 are consistent with the training results in most cases, and the performance of accuracy and loss are slightly worse than that of the training performance due to the difference in training and test samples. When $\tau = 3$, the loss of the test result is about 0.6 and accuracy is about 92%. It is worth noting that when the value of τ is equal to 5, the loss performance cannot converge on the test dataset. That is to say, the acceptable value of τ is less than 5 in practice.

Further, the performance of the well trained model is also tested. The models with the best training performance in Fig. 2 for different value of τ are selected, and the test results are given in Fig. 6. The performance of $\tau = 3$ and $\tau = 4$ is as good as that given in Fig. 2 and Fig. 5. However, when $\tau = 5$ and $\tau = 6$, the performance is not good, which is consistent with the phenomenon occuring in Fig. 5. Since $\tau = 5$ is the critical value and the performance is unstable, the distributions of the loss for the test samples using models obtained from different *global _ step* are counted in Fig. 7. When the average loss is 5, which is acceptable, the loss values of most of the samples are less than 5. But when the average loss is around 70, only 18.8% of the samples can gain the low loss which is less than 5, and 25.9% of the samples have extremely high loss which is larger than 100.

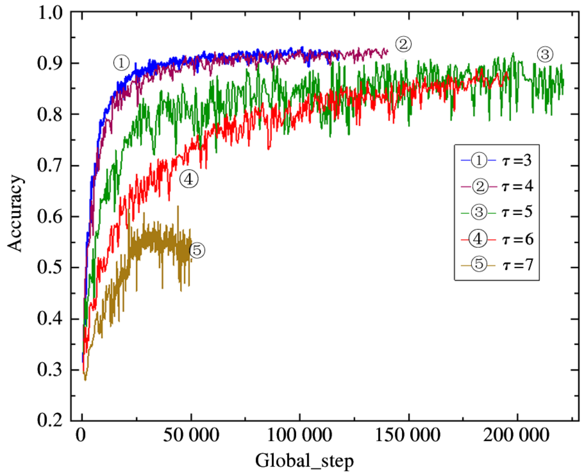
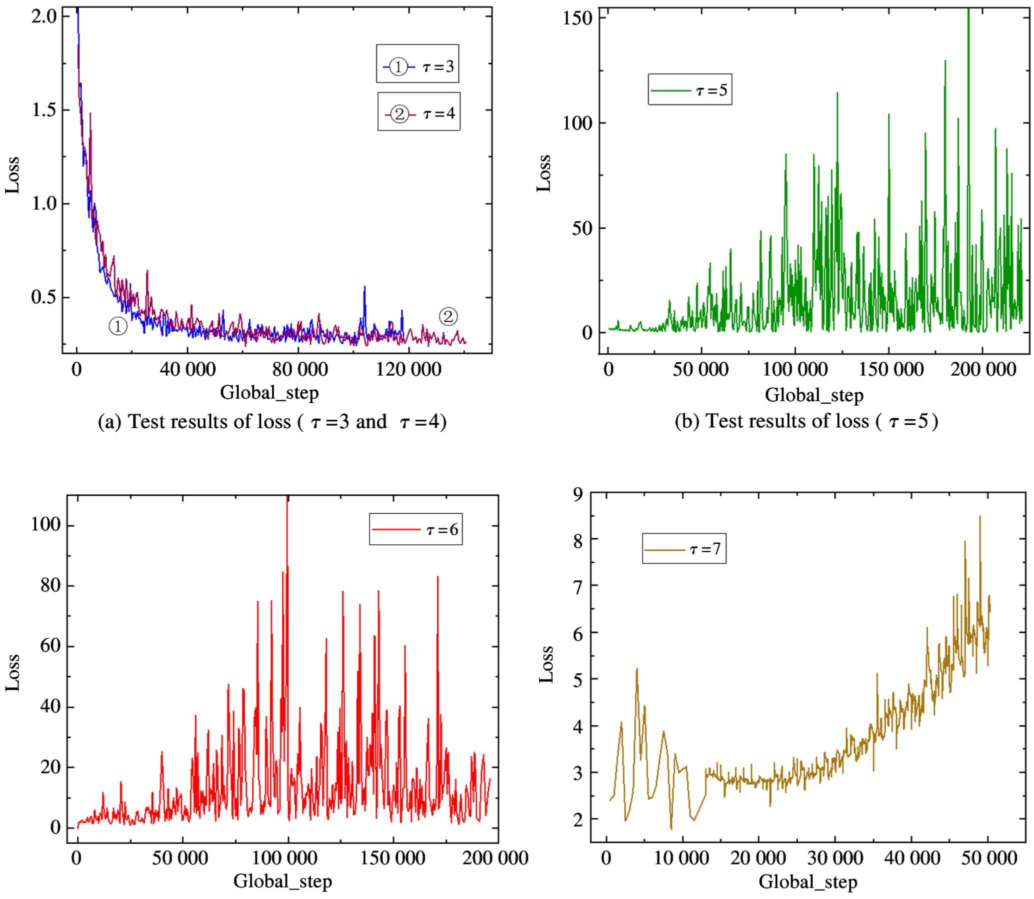


Fig. 4 Test results of accuracy ($\mu = 20\%$)



(a) Test results of loss ($\tau = 3$ and $\tau = 4$)

(b) Test results of loss ($\tau = 5$)

(c) Test results of loss ($\tau = 6$)

(d) Test results of loss ($\tau = 7$)

Fig. 5 Test results of loss ($\mu = 20\%$)

4.2 Performance evaluation of different values of μ

The impact of different proportion of pseudo-label dataset on the performance is evaluated. Since the critical value for $\mu = 20\%$ is $\tau = 5$, the value of τ is set as 3 in the following experiments to ensure a certain margin, and μ is set to 20%, 50% and 80%.

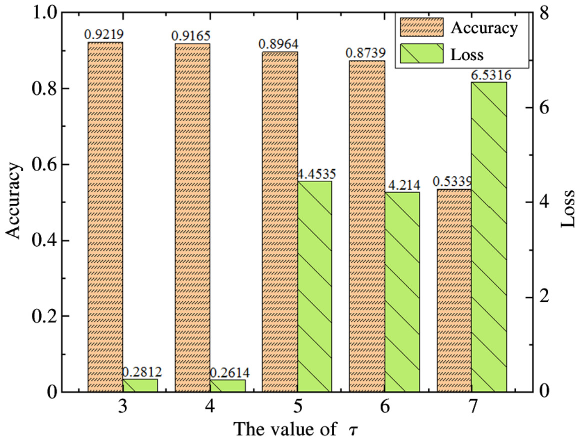


Fig. 6 The best test performance of different value of τ

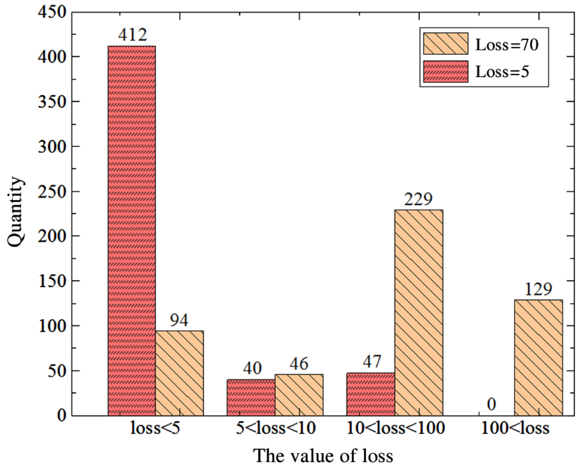


Fig. 7 Proportion of samples with the different test loss results

As shown in Fig. 8 and Fig. 9, the target classification model converges. When $\mu = 20\%$, the accuracy can increase to 98%, and the loss value can descent to 0.17. When $\mu = 50\%$, the model can also achieve an acceptable accuracy of about 95%, and the value of loss is about 0.3. However, when $\mu = 80\%$, the curve is different from those when $\mu = 20\%$ and $\mu = 50\%$. The convergence rate decreases obviously, and three steps can be observed in the ascending stage. At the beginning, the accuracy is kept in an extremely low level, which is because the credible pseudo label for the unlabeled data cannot be obtained based on the initial model. But after about 80 000 *global _ step*, it raises up gradually, and the final accuracy can reach 94%.

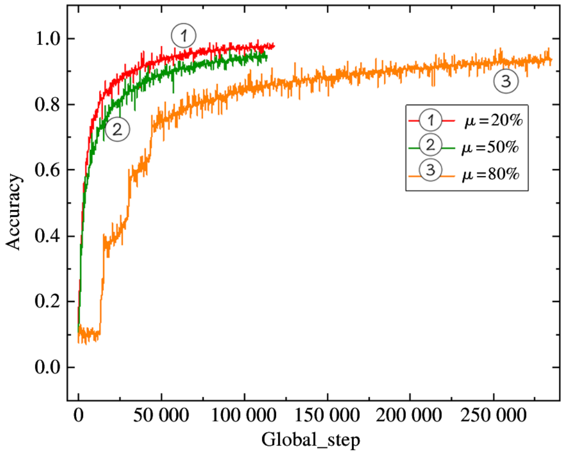


Fig. 8 Training results of accuracy ($\tau = 3$)

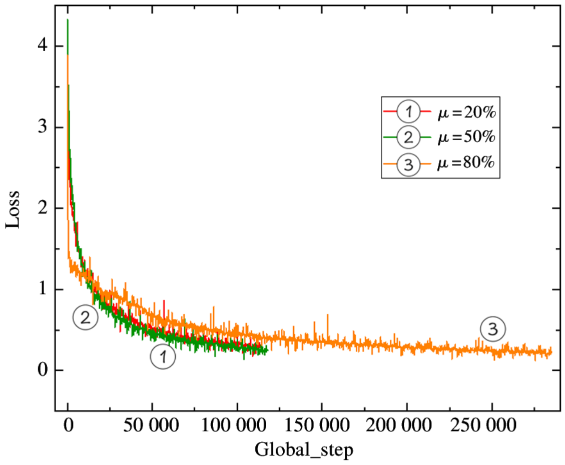


Fig. 9 Training results of loss ($\tau = 3$)

As for the test results given in Fig. 10 and Fig. 11, the performance is slightly degraded, but still acceptable. When $\mu = 20\%$ and $\mu = 50\%$, the accuracy of the model can reach 92%. But when $\mu = 80\%$, which is very large, the accuracy on the test dataset is only 87%.

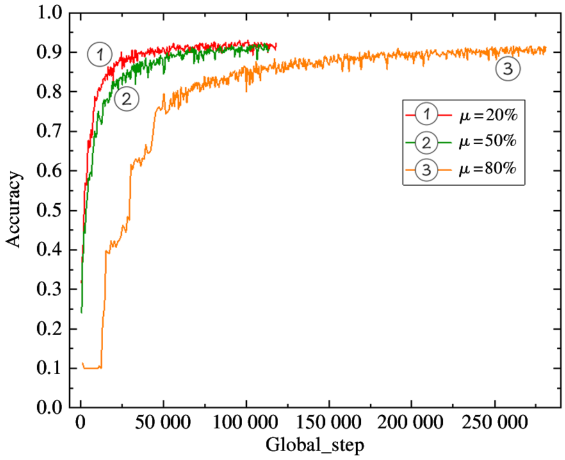


Fig. 10 Test results of accuracy ($\tau = 3$)

4.3 Calculation and communication cost analysis

The cost of the pseudo-label based semi-supervised learning algorithm is composed of two parts, i. e., the computational cost of the local training process and the communication cost of the parameter transmission.

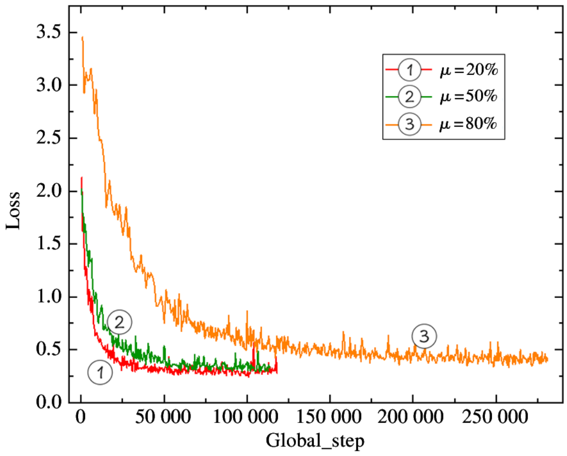


Fig. 11 Test results of loss ($\tau = 3$)

The measurement of computational cost mainly focuses on the number of floating-point operations (FLOPs). As mentioned in Section 3, the convolution neural network GoogLeNet is adopted, and the time complexity of all convolution layers can be expressed as

$$\Omega_{\text{Time}} \sim O\left(\sum_{l=1}^{d} n_{l-1} \times s_l^2 \times n_l \times m_l^2\right) \qquad (18)$$

where l is the index of a convolution layer, and d is the depth (number of convolution layers), s_l is the spatial size (length) of the filter, m_l is the spatial size of the output feature map, n_l is the number of filters in the l-th layer, and n_{l-1} is also known as the number of input channels of the l-th layer[20]. For the local training between two adjacent global updates in the experiment, the time complexity of one local worker is $\Omega_{\text{Time}}\tau \mid S_{it} \mid (1+\mu)$ FLOPs. Taking the specific hyper parameter of GoogLeNet into Eq. (14), the time complexity for one sample is $\Omega_{\text{Time}} \approx 1500$ M FLOPs according to the calculation in Ref. [17]. When $\tau = 3$, $\mid S_{it} \mid = 20$ and $\mu = 20\%$, the time complexity for one local worker between two adjacent global updates is 108 000 M FLOPs.

The communication cost of the parameter transmission is proportional to the number of parameters. For the convolution neural networks, the number of parameters can be expressed as

$$\Omega_{\text{para}} \sim O\left(\sum_{l=1}^{d} K_l^2 \times C_{l-1} \times C_l + \sum_{l=1}^{d} M^2 \times C_l\right) \qquad (19)$$

This is the sum quantity of parameters and feature maps of all layers, K is the size of convolution kernel, C_l is the number of output channels of the l-th layer, and M is the length of the output feature map. Since the worker needs to upload the local parameters to the aggregator and get the updated global parameters from the aggregator, the communication cost of one local worker can be measured as $2\Omega_{\text{para}}$. As for GoogLeNet, the number of the parameters of the whole model is $\Omega_{\text{para}} \approx 6.8$ M according to the calculation in Ref. [17]. Thus, the communication cost of one local worker is about 13.6 M.

Since all the workers train their local models in parallel and communicate with the aggregator at the same time, the cost of one global update can be estimated as $\Omega_{\text{Time}}\tau \mid S_{it} \mid (1+\mu) + 2\Omega_{\text{para}}$ from the aspect of time efficiency. Therefore, the cost of the whole distributed training process is $T_M[\Omega_{\text{Time}}\tau \mid S_{it} \mid (1+\mu) + 2\Omega_{\text{para}}]$, where T_M is the number of global updates.

5 Conclusions

In this paper, the main work is the implementation of the semi-supervised DML based on the pseudo-label method in a distributed framework. The local loss function of each distributed learning worker is studied, and the SGD-based parameter update equation is derived. The GoogLeNet based target classification model is evaluated by using the dataset of CIFAR-10. Results show that the model converges when the local update steps between every two global aggregation is less than 5 and the proportion of the pseudo-label dataset is 20%. Further evaluation results under the condition of $\tau = 3$ show that the increasing of the proportion of the pseudo-label dataset slows down the convergence rate and reduces the accuracy. But even when $\mu = 80\%$, the model still converges.

References

[1] CHEN W, LIU B, HUANG H, et al. When UAV swarm meets edge-cloud computing: the QoS perspective[J]. *IEEE Network*, 2019, 33(2): 36-43

[2] YANG F, WANG S, LI J, et al. An overview of Internet of vehicles[J]. *China Communications*, 2014, 11(10): 1-15

[3] WANG S, TUOR T, SALONIDIS T, et al. When edge meets learning: adaptive control for resource-constrained distributed machine learning [C] // IEEE INFOCOM 2018-IEEE Conference on Computer Communications, Honolulu, USA, 2018: 63-71

[4] ZHU X. Semi-supervised Learning Literature Survey[D]. Madison: University of Wisconsin Department of Computer Sciences, 2012: 33-35

[5] DEAN J, CORRADO G S, MONGA R, et al. Large scale distributed deep networks[J]. *Advances in Neural Information Processing Systems*, 2012, 25: 1223-1231

[6] WANG S, TUOR T, SALONIDIS T, et al. Adaptive federated learning in resource constrained edge computing systems[J]. *IEEE Journal on Selected Areas in Communications*, 2019, 37(6): 1205-1221

[7] SEIL L, HANJOO K , JAEHONG P , et al. Tensor lightning: a traffic-efficient distributed deep learning on commodity spark clusters[J]. *IEEE Access*, 2018, 6: 27671-27680

[8] NIU F, RECHT B, RE C, et al. HOGWILD!: a lock-

free approach to parallelizing stochastic gradient descent [J]. *Advances in Neural Information Processing Systems*, 2011, 24: 693-701

[9] PAN X, LAM M, TU S, et al. CYCLADES: conflict-free asynchronous machine learning[C]//Advances in Neural Information Processing Systems, Barcelona, Spain, 2016: 2568-2576

[10] LEE D H. Pseudo-label: the simple and efficient semi-supervised learning method for deep neural networks[C]//Challenges in Representation Learning, Atlanta, USA, 2013: 2-6

[11] WU H, PRASAD S. Semi-supervised deep learning using pseudo labels for hyperspectral image classification[J]. *IEEE Transactions on Image Processing*, 2018, 27(3): 1259-1270

[12] SHI Y, XU H, JIANG L, et al. Few-shot modulation classification method based on feature dimension reduction and pseudo-label training[J]. *IEEE Access*, 2020, 8: 140411-140425

[13] ENGUEHARD J, HALLORAN P, GHOLIPOUR A. Semi-supervised learning with deep embedded clustering for image classification and segmentation[J]. *IEEE Access*,2019, 7: 11093-11104

[14] WANG Y, NAKACHI T. A privacy-preserving learning framework for face recognition in edge and cloud networks [J]. *IEEE Access*, 2020, 8: 136056-136070

[15] ZHU S, CAO R, YU K. Dual Learning for semi-supervised natural language understanding[J]. *IEEE Transactions on Audio*,*Speech*, *and Language Processing*, 2020, 28: 1936-1947

[16] SHWARTZ S, DAVID S. Understanding machine learning: from theory to algorithms[M]. London: Cambridge University Press, 2004: 188-191

[17] SZEGEDY C, LIU W, JIA Y Q, et al. Going deeper with convolutions[C]//IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, USA, 2015: 1-9

[18] ABADI M, AGARWAL A, BARHAM P, et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems[EB/OL]. https://arxiv.org/pdf/1603.04467.pdf:arXiv, (2016-03-16),[2021-03-18]

[19] KRIZHEVSKY A. Learning multiple layers of features from tiny images[J]. *Handbook of Systemic Autoimmune Diseases*, 2009: 54-57

[20] HE K M, SUN J. Convolutional neural networks at constrained time cost[C]//IEEE Conference on Computer Vision and Pattern Recognition (CVPR) IEEE, Boston, USA, 2015: 5353-5360

WANG Xiaoxi, born in 1997. She received the B. S. degree from Faculty of Information and Electrical Engineering of Hebei University of Engineering in 2019. She is currently pursuing the M. S. degree in Faculty of Information Technology from Beijing University of Technology. Her current research interests include distributed machine learning and wireless communication networks.