SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

August 2020, Vol. 63 182102:1–182102:11
https://doi.org/10.1007/s11432-019-2705-2

Collaborative deep learning across multiple data centers

Haibo MI[1], Kele XU[1*], Dawei FENG[1], Huaimin WANG[1], Yiming ZHANG[1], Zibin ZHENG[2], Chuan CHEN[2] & Xu LAN[3]

[1] *College of Computer, National University of Defense Technology, Changsha* 410073*, China;*
[2] *School of Data and Computer Science, Sun Yat-sen University, Guangzhou* 510000*, China;*
[3] *School of Electronic Engineering and Computer Science, Queen Mary University of London, London* E1 4NS*, UK*

Received 21 March 2019/Revised 26 September 2019/Accepted 28 October 2019/Published online 14 July 2020

Abstract Valuable training data is often owned by independent organizations and located in multiple data centers. Most deep learning approaches require to centralize the multi-datacenter data for performance purpose. In practice, however, it is often infeasible to transfer all data of different organizations to a centralized data center owing to the constraints of privacy regulations. It is very challenging to conduct the geo-distributed deep learning among data centers without the privacy leaks. Model averaging is a conventional choice for data parallelized training and can reduce the risk of privacy leaks, but its ineffectiveness is claimed by previous studies as deep neural networks are often non-convex. In this paper, we argue that model averaging can be effective in the decentralized environment by using two strategies, namely, the cyclical learning rate (CLR) and the increased number of epochs for local model training. With the two strategies, we show that model averaging can provide competitive performance in the decentralized mode compared to the data-centralized one. In a practical environment with multiple data centers, we conduct extensive experiments using state-of-the-art deep network architectures on different types of data. Results demonstrate the effectiveness and robustness of the proposed method.

Keywords collaborative learning, multiple datacenters, distributed machine learning

Citation Mi H B, Xu K L, Feng D W, et al. Collaborative deep learning across multiple data centers. Sci China Inf Sci, 2020, 63(8): 182102, https://doi.org/10.1007/s11432-019-2705-2

1 Introduction

The sensitive data, such as medical imaging data, genetic sequences, financial records and other personal information, are often managed by independent organizations like hospitals and companies [1]. Many deep learning (DL) algorithms prefer to use as much data as possible distributed in different organizations for training, because the performance of these DL algorithms directly depends on the amount of high-quality data not only for rarely occurring patterns but also for the robustness to the outliers [2]. In practice, however, directly sharing data between different organizations are of great difficulties owing to many reasons including privacy protection, legal risk consideration and conflict of interests [3]. Therefore, it has become an important research topic for both academy and industry to fully employ the data of different organizations for training DL models without centralizing the data, while achieving similar performance compared to centralized training after moving all data together.

Recently, there has been a trend to use collaborative solvers to train a global model on geo-distributed, multi-datacenter data without directly sharing data between different data centers [4, 5]. Specifically,

[*] Corresponding author (email: kelele.xu@gmail.com)

several participants independently train the DL models for a while, and periodically aggregate their local updates to construct a shared model. Only parameters are exchanged and all the training data are kept in the original places [6]. However, there are several challenges for this approach.

- Large performance gap compared to the centralized mode. When training on the disjoint multi-party data, traditional deep models using stochastic gradient descent (SGD) are difficult to provide competitive performance compared to their centralized mode. Further, with limited data size, the local learner is vulnerable to fall into the local optima, as deep models are generally non-convex.
- High communication cost. Different datasets are stored on different data centers (on private cloud or public cloud). DL algorithms typically require frequent communication to exchange parameter updates such that the shared deep model is of superior performance. However, current parameter servers are designed for high-speed local area networks (LANs). Due to the limitation of network bandwidth of wide-area networks (WANs), parameters of the global model cannot be exchanged frequently in the multi-datacenter environment. Therefore, it is necessary to decrease the communication cost for parameter exchange between different data centers, while retaining the accuracy of the shared model.
- High model aggregation complexity. The update strategy to aggregate the local models is complicated. As the different participant has its own training setting, the approach to aggregate local learners should be simple. In addition, the aggregation method should support the learning procedure using different deep neural network architectures.

In this study, we propose a multi-datacenter based collaborative deep learning method (denoted as co-learning), which (1) minimizes the performance gap between the centralized and decentralized modes, (2) minimizes the inter-datacenter communication cost during the co-training procedure over WANs, (3) is applicable to a wide variety of deep network architectures without any change.

The co-learning approach proposes two strategies to improve the performance of a shared model in distributed learning, based on the conventional model averaging method. First, we adopt the modified cyclical learning rate (CLR) [7], so as to avoid falling into the local optima during the local training procedure. Second, we enlarge the number of local epochs when the difference between two consecutive shared models decreases to be less than a threshold, so as to increase the diversity between local models and reduce the inter-datacenter communication cost. The synchronization period is extended from milliseconds or seconds to ten of minutes or even hours.

Surprisingly, despite the claims from previous studies [6, 8], we find that model averaging in the decentralized mode can provide competitive performance compared to the traditional centralized mode. Extensive experiments are conducted on three different tasks: image classification, text classification and audio classification. Using the co-learning method, we have tested various state-of-the-art neural network architectures including VGGNet [9], ResNet [10], DenseNet [11] and Capsule architectures [12]. All the experiments reveal that the proposed co-learning approach can provide superior performance in the decentralized mode. In summary, the main contributions include.

- We propose a collaborative DL approach using model averaging. With two simple strategies (CLR and increased number of local training epochs), we show that model averaging can provide competitive performance compared to the centralized mode.
- Our approach enables the training of collaborative DL in the practical WAN environment.
- The proposed co-learning is flexible enough to be applied to a wide range of DL architectures without any change.

The remainder of this paper is organized as follows. Section 2 descries the related work, while Section 3 presents the details of our co-learning approach. Section 4 describes the experimental results, the discussion and conclusion are given in Section 5.

2 Related work

Like [13–15], DL models are traditionally trained on a single cluster by centralizing data from distributed sources. However, with the increase of privacy constraints, data size and model complexity, it is more and more difficult for organizations to centralize data. An increasing trend to conduct DL is to partition the training dataset in the geo-distributed environment, concurrently train separate models on the disjoint subset. By aggregating the updates of local model's parameters via a parameter server, a shared model can be constructed. In this paper, we define this method as collaborative DL, which can be applied in the practical situation where each participant wants to hide their own training data from each other.

2.1 Parallelized SGD

Many recent attempts have been made to parallelized SGD based learning schemes across multiple data centers [5,16]. Nevertheless, the geo-distributed nature of data prevents its widespread utilization between organizations, owing to the aforementioned reasons like limitations in cross data center connectivity, or data sovereignty regulations restriction. To break through these restrictions, increasing effort has been made. Ref. [17] used the parallel SGD algorithm to train the model for the consideration of privacy preservation. The communication cost between the client and the server is prohibitively high, thereby can seldom be deployed in WAN scenarios owing to the bandwidth limit. Ref. [1] proposed a secure multi-party computation (MPC) approach for simple and effective computations, yet its overhead for complex computations and the model training is nontrivial. Consequently, this approach is more suitable for shallow machine learning (ML) models, while it is difficult to be applied to deep learning (DL) models [18].

It is worth mentioning that this paper targets to solve the collaborative DL across multiple geo-distributed datacenters and the optimization methods (e.g., synchronized batch normalization, data augmentation) of model training within one data center cloud be used to improve the local model performance. Furthermore, to reduce the communication cost, many compression approaches have been explored, such as, gradient quantization [19] and network pruning [20], knowledge distillation [21, 22].

2.2 Model averaging

For collaborative DL, model averaging is an alternative method for parallelized SGD [8,23], and can reduce the risk of privacy leak [6]. However, most of the previous literature [24,25] claimed that traditional model averaging cannot provide satisfied performance in the distributed setting, as a deep neural network is a highly non-convex model. For example, Ref. [8] claimed that the model averaging algorithm did not work well for speech recognition models. The main reason to support these claims was that, when the size of the data is limited for the training of a local model, the local models may fall into different local-optima. The shared model obtained by averaging the local model's parameters, might even perform worse than any local model. Moreover, in the follow-up step, the shared model would be used as a new starting point of the successive iterations of local training, and the poor performance of the shared model would drastically slow down the convergence of the training process and further decreas the performance of the shared model [24]. To avoid falling into local optima, many regularization methods were proposed [26,27]. In [7], it was found that using a CLR could lead to better generalization than the conventional training.

A federated learning approach [6] was proposed for a data parallelization in the context of DL. It targeted to solve the model training on massive mobile devices, and a fixed number of epochs for local model training was employed for the devices. However, we utilize a modified CLR and an increasing number of epochs for local model training to get competitive performance in the decentralized mode with comparison to the centralized one.

3 Methodology

3.1 Notation and problem formulation

A typical process of parallel training for deep models is illustrated in Figure 1. Participants train their local models with the individual DL platform in their private data centers (in private clouds or trusted

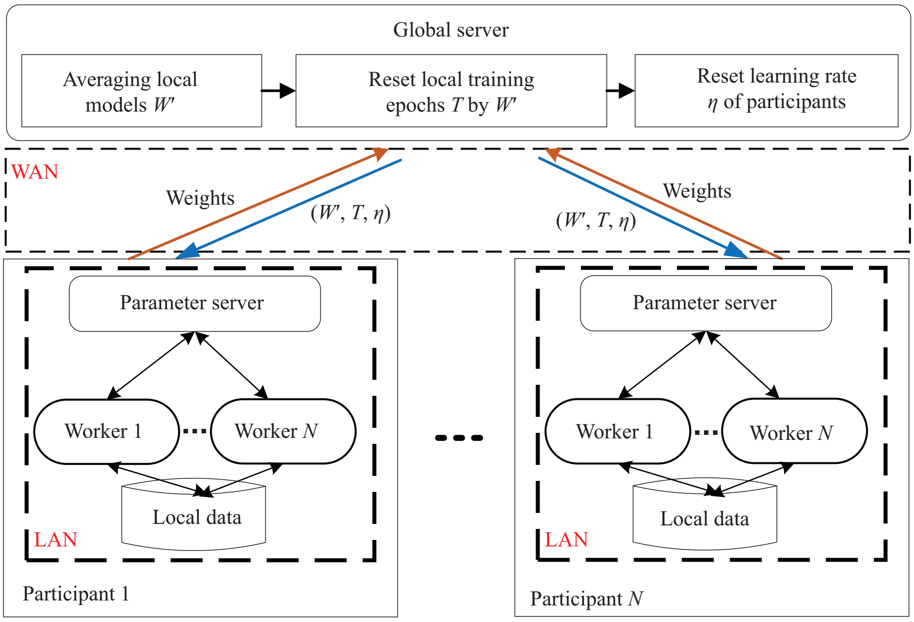

Figure 1 (Color online) Workflow of co-learning. Assume that the participants are different data centers. Each participant holds an amount of private data and uses the disjoint data to train a local classifier. The local model parameters will be averaged by the global server to formulate the new shared model, which in turn are used for as the starting point for the next round of local training. Besides the new shared model, the global server also updates the number of local training epochs and the learning rate.

public clouds). These local data centers communicate over WANs. In the piratical situation, owing to the limitation of WAN bandwidth, participants cannot exchange updates frequently.

In the following, we denote a deep neural network as $f(\boldsymbol{w})$, where $\boldsymbol{w}$ represents the parameters of this neural network model. In addition, we denote the outputs of the model $f(\boldsymbol{w})$ on the input x as $f(\boldsymbol{w}, x)$. In the parallel training of deep models, suppose there are K participants and each of them holds a local dataset $D_k = \{(x_{k,1}, y_{k,1}), \ldots, (x_{k,m_k}, y_{k,m_k})\}$ with size m_k, $k \in \{1, \ldots, K\}$. Denote the weight of the neural network model at the iteration t of i-th round (with T_i epochs been performed) on the participant k as $\boldsymbol{w}_k^{i,t}$. Then a typical parallel training procedure for neural network implements the following two steps.

- Local training for the participants. At the t-th iteration of round i, participant k updates its local model by using SGD. We refer to one full iteration over all local training data as an epoch. The local model is communicated and aggregated to formulate a shared model after T_i epochs, which is decided dynamically by the global server. Then each participant can initialize its local parameters for the following local training by downloading latest values of the shared model from the global server. During the local training, the participant does not need to exchange the data with other participants. At the iteration t of i-th round, the empirical loss of the k-th local model is defined as

$$\mathcal{L}(f(\boldsymbol{w}_k^{i,t}, x_k), y_k) = \sum_{m=1}^{m_k} \mathcal{L}(f(\boldsymbol{w}_k^{i,t}, x_{k,m}), y_{k,m}). \tag{1}$$

Specifically, participant k updates its local model from $\boldsymbol{w}_k^{i,t}$ to $\boldsymbol{w}_k^{i,t+1}$ by minimizing the training loss using SGD.

- Model aggregation for the global server. Firstly, the global server initializes the shared model parameters and pushes them to all participants. The local training of each participant follows the aforementioned procedures. If one participant k fails to upload its parameters owing to network errors or other failures, the global server will restart the local training process of participant k. After all K participants finish their updates in the i-th round and obtain the parameter $\boldsymbol{w}_k^i$, the global deep neural network model is updated by taking the average of the K sets of parameters, i.e.,

$$\bar{\boldsymbol{w}}^i = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{w}_k^i, \tag{2}$$

which is further sent back to the local participants, and set as the initial parameters for the following training. Further, the number of epochs T_i is reset according to the conditions defined in (4). The parameters of the shared model, as well as T_i and η^i, are sent back to local participants, and used as the starting point for the next round of local training (as can be seen in Figure 1).

3.2 Cyclical learning rate and increasing local epochs (ILE)

The learning rate and local epoch are two important hyper-parameters that directly affect the convergence speed and accuracy of the training models. The previous research failed to obtain high model performance by the model averaging method. However, we find that by adjusting these two parameters, the performance of the model can be greatly improved. Specifically, we set the following heuristic rules for these two parameters.

To avoid falling into local optima, we employ the CLR schedule in the training phase of the local participants. Specifically, within the i-th communication round, we decay the learning rate with an exponential annealing for each epoch j as follows:

$$\eta_j^i = \eta^i \times r^{\frac{j}{T_i}}, \tag{3}$$

where r is the decay rate (in our experiment, r is set as 1/4), η^i is the shared learning rate in the i-th round, used as an initial value to update each participant's local learning rate. It can be updated as i grows. For simplicity, we set η^i as a constant value (i.e., 0.01) in this paper. As mentioned above, the global server has to decide the number of epochs for local participants dynamically, since these values have a significant impact on the accuracy of the shared model. The number of local epochs in the i-th round (T_i) is updated based on the following rules:

$$T_i = \begin{cases} T_0, & \text{if } i = 0, \\ 2 \cdot T_{i-1}, & \text{if } i > 0 \ \& \ \frac{|\bar{\boldsymbol{w}}^i - \bar{\boldsymbol{w}}^{i-1}|}{|\bar{\boldsymbol{w}}^{i-1}|} \leqslant \epsilon, \\ T_{i-1}, & \text{if } i > 0 \ \& \ \frac{|\bar{\boldsymbol{w}}^i - \bar{\boldsymbol{w}}^{i-1}|}{|\bar{\boldsymbol{w}}^{i-1}|} > \epsilon, \end{cases} \tag{4}$$

where ϵ is used to control the convergence precision of the shared model parameters. In other words, the number of epochs in each round is increased by a factor of 2 at every communication round once the change of the shared model parameters is lower than ϵ. The pseudocode of the proposed co-learning is given in Algorithm 1.

Algorithm 1 Co-learning

```
Initialize $w^0$, $\eta^0$ and $T_0$.
for each round $i = 0, 1, 2, \ldots, N$ do
  Reset $T_i$ according to the (4);
  Send $w^i$, $\eta^i$ and $T_i$ to participants;
  for each participant $k \in K$ parallel do
    for local epoch $j$ from 1 to $T_i$ do
      Update $\eta_j^i$ according to the (3);
      $w_k^i \leftarrow$ localSGD($w^i$, $\eta_j^i$);
    end for
    upload $w_k^i$ to server;
  end for
  $w^{i+1} \leftarrow \frac{1}{K}\sum_{k=1}^{K} w_k^i$;
end for
```

4 Experiments

4.1 Experimental settings

To demonstrate the effectiveness of co-learning, empirical experiments were conducted on three different tasks: image classification, text classification and audio classification. For image classification, both CIFAR-10 and ImageNet-2014 [28] were used for the experiments; for text classification, Toxic comment classification dataset was used in the classification tasks; for audio classification, Google speech command data [29] and Audio Set [30] were employed. Using the proposed co-learning method, different neural network architectures were tested, including state of the art neural networks architectures. We conducted experiments across five geo-distributed data centers in a public cloud, each equipped with a GPU server with four Tesla P40. All kinds of datasets were randomly allocated to 5 participants in an equally distributed manner. All our experiments were implemented in TensoFlow slim. Also, it is worthwhile to notice that all the results were obtained using the average of five repetitive trials of the experiments. The following three groups of experiments were conducted.

- We performed a thorough ablation study to highlight the benefits of CLR and ILE on model averaging. We also employed the exponential learning rate (ELR, i.e., non-CLR) and fixed local epochs (FLE) for the quantitative comparison.
- It is a common strategy to integrate the training results of each participant by using ensemble learning. In more detail, each participant independently trains its own model, without interacting with other participants during the training process. The average output of each participants model is used as the final prediction. With the CIFAR-10 dataset, accuracy comparison between ensemble-learning and co-learning were carried out on different kinds of network architectures. Besides, training a deep model using the entire dataset in a single data center (denoted as vanilla-learning below) is introduced as a reference for comparison. Except for the two proposed strategies for co-learning, other configuration settings for vanilla-learning are kept the same as the settings of co-learning.
- Moreover, to make a quantitative comparison between the data centralized training method and de-centralized one, we conducted comprehensive experiments using vanilla-learning and the proposed co-learning on different kinds of deep network architectures and various types of datasets.

4.2 Ablation study on CLR and ILE

We run experiments on the CIFAR-10 dataset, which consists of 10 classes 32×32 images with three channels. 50000 training images are partitioned into five disjoint subsets, which are stored in five different data centers, and each containing 10000 samples. The 10000 test images are used for the evaluation. The initial values of T_0 for the DenseNet-40, ResNet-152, Inception-V4, and Inception-ResNet-V2 models are 5, 5, 20, 5 respectively. The batch size of the experiments was set to 32.

Using the pairwise combination of (CLR, ELR) and (ILE, FLE), Figure 2 shows the accuracy of model averaging method for training DenseNet-40, ResNet-152, Inception-V4 and Inception-ResNet-V2. As can be seen from the Figure 2.

- The combination of CLR and ILE achieves the highest accuracy on four different network architectures. The results demonstrate that co-learning (CLR+ILE, the red line) tends to generalize better, which indicates the benefits of both CLR and ILE. The reason behind might be that co-learning could converge to flat local optima rather than sharp, isolated optima. Such flat regions are robust to data perturbations as well as perturbations of the parameters, all of which are crucial factors to achieve good generalization.
- Similar to previous studies using model averaging, the combination of ELR and FLE (the green line) cannot effectively improve the performance of the collaborative learning, and tends to be over-fitting in the training phase. In other words, the performance of the shared model cannot be improved by using model averaging alone without any optimization strategy.
- Further, ELR+ILE leads to a converged result, however, the CLR+FLE prones to be over-fitting. This indicates the ILE may bring more performance gains than the CLR on the CIFAR-10 dataset, and

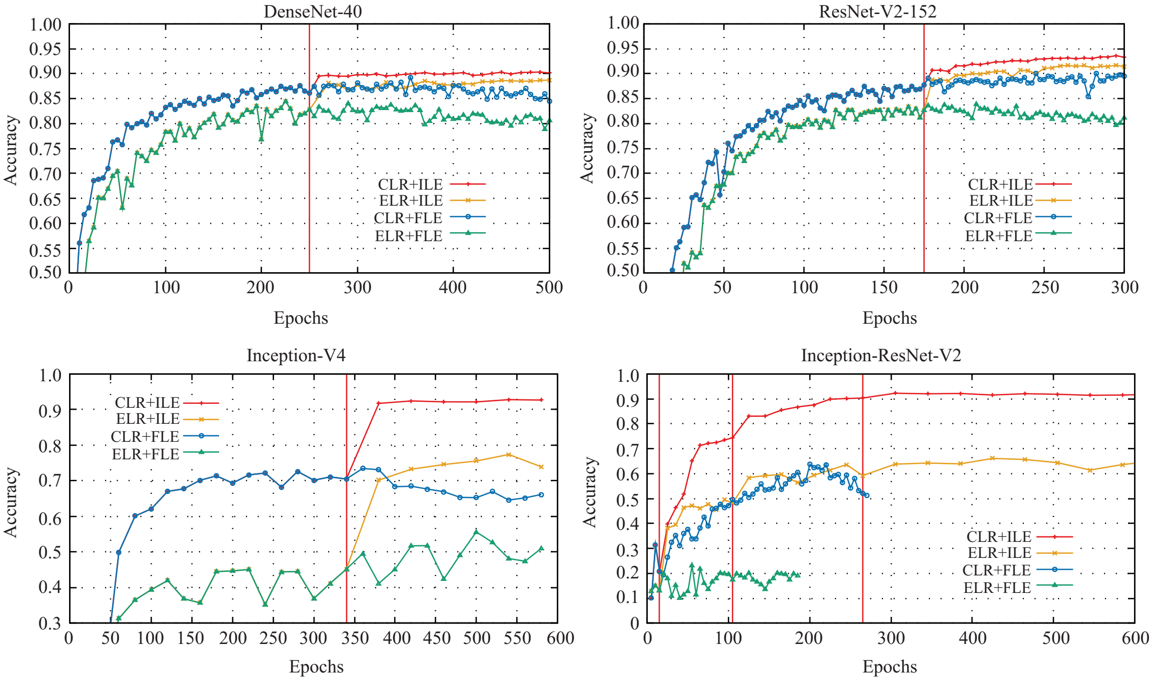

Figure 2 (Color online) Accuracy on the CIFAR-10 dataset by using different strategies. The employed neural network architectures include: Inception-V4, ResNet, Inception-ResNet, DenseNet. Using the proposed ILE strategy, DenseNet-40, ResNet-152, Inception-V4 enlarges T_0 at the 250th, 175th and 340th epoch respectively, while Inception-ResNet-V2 increases T_0, T_1, T_2 at the 15th, 105th and 265th epoch, respectively. After the adjustment, the performance of each shared model sees a significant improvement in the following rounds. The FLE strategy in the bottom-right figure (the blue line and green line) experiences an early stop, as it does not boost the performance in the previous rounds.

ILE can increase the diversities between different local models, which consequently derives a better shared model.

The experiments indicate the benefit of both CLR and enlarging local training epoch strategies. On one hand, by restarting the optimization with a large learning rate, the intrinsic random motion across gradient direction prevents the model from reaching any of the sharp basins along its optimization path, which allows the model to find a better local optima. In this way, although the performance temporarily suffers when the learning rate cycle is restarted, the performance eventually surpasses the previous cycle after annealing the learning rate. On the other hand, by increasing the number of local epoch in the iterations, each local model could do large steps in the parameter space to get diverse networks. Thus, it is expected to achieve better possible accuracy on its local datasets. Moreover, the ILE leads to add the diversities between different local models, which can be averaged to get a better shared model.

4.3 Communication cost

We briefly summarize the communication cost for the proposed co-learning approach. Table 1 exhibits the communication interval and the transferred volume of one model in a round. The 2nd column reveals the communication interval between a local participant and the global server in a communication round before T_0 is increased (i.e., time elapsed between two consecutive model-synchronizations). Specifically, using the CLR+ILE strategy, the communication intervals for different models range from minutes to hours, e.g., 60 min for the Inception-V4 and 27.5 min for the Inception-ResNet-152. Moreover, if T is enlarged in the following training, the communication interval will be further extended. Take the Inception-V4 as an example, in the 340th epoch, the number of local epochs T is increased from 20 to 40. Consequently, the communication interval is enlarged from 60 min to 120 min, which can greatly alleviate the dependence on the WAN bandwidth.

In short, comparing with the other three strategies, the CLR+ILE strategy can improve the performance of the shared model as well as reducing the communication cost. It is also worthwhile to notice that

Table 1 Stats for using CLR+ILE on different models in a communication round

Models	Communication interval (min/T_0)	Communication volume (MB)
DenseNet-40	4.5 / 5	13
ResNet-152	30 / 5	223
Inception-V4	60 / 20	168
Inception-ResNet-V2	27.5 / 5	218

Table 2 CIFAR-10 accuracy comparison between ensemble-learning, vanilla-learning and co-learning

Model	Accuracy (%)		
	Vanilla	Ensemble	Co-learning
VGG-19	89.44	80.39	**89.64**
ResNet-152	92.64	85.4	**93.51**
Inception-V4	91.34	83.83	**92.07**
Inception-ResNet-V2	**92.86**	84.7	92.83
DenseNet-40	91.35	81.24	**91.43**

we do not employ the compression technique by which the communication cost can be further decreased.

4.4 Ensemble-learning, vanilla-learning and co-learning

In the following experiment, using the CIFAR-10 dataset, we show the comparison between ensemble-learning, vanilla-learning and co-learning, on five kinds of models (i.e., VGG-19, ResNet-152, Inception-V4, Inception-ResNet-V2, and DenseNet-40). For the vanilla-learning, the ELR is employed. Table 2 illustrates the results. Bold fonts indicate the best results. It can be observed that using ensemble-learning, the model accuracy is significantly declined, i.e., nearly 10% reduction compared with the vanilla-learning. As each participant has only 1/5 disjoint training data, the accuracy of the local model is poor. Consequently, by averaging the outputs of each model after independent local training, it is infeasible to obtain a competitive performance with the one using vanilla-learning. On the contrary, the accuracy obtained by the co-learning achieves competitive results with comparison to the vanilla-learning. Surprisingly, co-learning on four models (i.e., VGG-19, ResNet-152, Incpeiton-V4 and DenseNet-40) even achieves better performance than the vanilla-learning. These results exhibit again the effectiveness of the CLR and ILE on model averaging.

4.5 Comparison between co-learning and vanilla-learning

4.5.1 *Image classification*

We conduct another image classification experiments on the ImageNet-2014 to further evaluate the generalization accuracy of co-learning, as the classification error on ImageNet is particularly important because many state-of-the-art computer vision problems derive image features or architectures from ImageNet classification models.

In the training phase, we follow standard data augmentation practices: scale and aspect ratio distortions, random crops, and horizontal flips. The batch size is set to 256. Three different state-of-the-art models (VGG, Inception-V4, ResNet-V2-101) are trained, by using both of the co-learning and vanilla-learning approach. Top-1 and Top-5 accuracy rates are reported in Table 3. We find that the co-learning leads to improved accuracy over vanilla-learning using the same network architecture settings, which illustrates the promising potential of co-learning. This indicates that the co-learning approach can be generically applied to large-scale image classification settings.

4.5.2 *Text classification*

We also run experiments on a large-scale toxic comments classification task to demonstrate the effectiveness of co-learning on a natural language processing problem. In more detail, the training dataset consists of 159571 Wikipedia comments, which have been labeled by human raters for toxic behavior,

Table 3 Test accuracy of ImageNet-2014 using different models

Model		Accuracy(%)	
		Top-1	Top-5
VGG-19	Vanilla	70.41	88.12
	Co-learning	**70.62**	**88.7**
Inception-V4	Vanilla	79.16	93.82
	Co-learning	**79.35**	**94.28**
ResNet-V2-101	Vanilla	75.66	92.28
	Co-learning	**75.85**	**92.39**

Table 4 Multi-class AUC on toxic comment classification challenge dataset

Model	Multi-class AUC (%)	
	Vanilla	Co-learning
LSTM	98.52	**98.79**
Capsule	98.32	**98.75**

Table 5 TensorFlow speech commands recognition

Method	Validation accuracy (%)	Test accuracy (%)
Vanilla	93.1	**93.3**
Co-learning	**93.3**	93.2

while 153164 records are used for the evaluation. The types of toxicity are: toxic, severe toxic, obscene, threat, insult, identity hate. In the training stage, the training dataset is randomly partitioned into 5 participants. Each contains equal-size disjoint examples, which are stored in the different data center.

For the classification, the employed models include long short-term memory (LSTM) [31] and Capsule [32]. The input embeddings for each word are of dimension 300 (for the pre-trained word vectors, fastText [33] is employed). For LSTM model, we use a bidirectional GRU and the batch size is set to 128 here. For Capsule model, the input is the reshaped embedding vectors, while the second layer is a primary capsule layer with strides of 1. This layer consists of 32 "Component Capsules" with a dimension of 8. Final capsule layer includes 6 capsules, refereed to as "Class Capsules", one for each type of toxicity. The dimension of these capsules is 16.

For the evaluation, the mean column-wise receiver operating characteristic curve (ROC) area under curve (AUC) is used. As can be ceen from the Table 4, the co-learning improves the accuracy with comparison to the vanilla-learning. The experimental results suggest that our method is practically applicable to the large-scale text classification task.

4.5.3 *Audio classification*

Next, we conduct experiments on the audio classification task. Two different datasets are used: Google commands dataset and Audio Set.

- Google command recognition. Google commands dataset contains 65000 utterance, in which each audio is about one-second long and belongs to one out of 30 classes. The voice commands include classes, such as left, right, yes, no. To process the utterances, we first calculate the log Mel spectrograms from the original raw audio signal at a sample rate of 16 kHz. The model architecture consists of two convolutional layers followed by two fully connected layers and then a softmax layer for classification. While this model is not the state-of-the-art, it is sufficient for our needs, as our goal is to the quantitative study, not achieve the best possible accuracy on this task. Table 5 gives the recognition accuracy of the co-learning, and vanilla-learning. As seen from the Table 5, nearly the same accuracy can be achieved using the co-learning.

- Audio event classification using audio set. To make a quantitative comparison between the co-learning and the vanilla-learning, large-scale audio event classification experiments are conducted. Audio set consists of a large ontology of 632 sound event classes and a collection of 2 million human-labeled

Table 6 Audio set classification task using a single/multi data center(s)[a)]

	Vanilla / co-learning		
Model	MAP[b)]	AUC	d-prime
AP	**0.300** / 0.299	**0.964** / 0.962	**2.536** / 2.506
MP	0.292 / 0.292	**0.960** / 0.959	**2.471** / 2.456
SA	0.337 / 0.337	**0.968** / 0.966	**2.612** / 2.574
MA	**0.357** / 0.352	0.968 / 0.968	**2.621** / 2.618

a) AP represents result of CRNN with average pooling, MP for CRNN with max pooling, SA for CRNN with single attention and MA for CRNN with multi-attention.

b) MAP: mean average precision.

sound clips (mostly 10-second length) drawn from 2 million YouTube videos.

Each audio recording feature has 240 frames by 64 mel frequency channels, which are employed as the input for different architectures. The convolutional recurrent neural networks (CRNN) are adopted for the classification task. Specifically, one bi-directional gated recurrent neural network with 128 units is used. Instead of applying a single-level attention model after the fully connected neural network, multiple attention modules [34] can be applied after intermediate layers as well. The batch size is set to 128 for different network architectures. Table 6 summarizes the results of different network architectures. Overall, the accuracy is similar by using the co-learning and the vanilla-learning. The result demonstrates the general applicability of our method on audio datasets.

5 Discussion and conclusion

In this paper, we present co-learning, a novel collaborative DL approach, for training deep models on disjoint multi-party datasets. Extensive experiments are conducted on different types of data, including image, text, and audio, with the goal to demonstrate the effectiveness of co-learning both quantitatively and qualitatively. All the experiments demonstrate that co-learning method can provide competitive (sometimes, even better) performance, with comparison to the data centralized learning.

The paper justifies the benefit of both CLR and enlarging local training epoch strategies. The reason behind might be that co-learning could converge to flat local optima rather than sharp, isolated local optima. Such flat regions are robust to data perturbations as well as perturbations of the parameters, all of which are crucial factors to achieve good generalization.

In brief, our co-learning method offers a solution for collaborative DL in the context of multi-parties data. Future work includes the practical privacy mechanism, secured MPC in the co-learning framework.

Acknowledgements This work was supported by National Grand R&D Plan (Grant No. 2016YFB1000101).

References

1 Tian L, Jayaraman B, Gu Q, et al. Aggregating private sparse learning models using multi-party computation. In: Proceedings of NIPS Workshop on Private Multi-Party Machine Learning, Barcelona, 2016

2 Amir-Khalili A, Kianzad S, Abugharbieh R, et al. Scalable and fault tolerant platform for distributed learning on private medical data. In: Proceedings of International Workshop on Machine Learning in Medical Imaging, 2017. 176–184

3 Yang Q, Liu Y, Chen T, et al. Federated machine learning: concept and applications. ACM Trans Intell Syst Tech, 2019, 10: 12

4 Cano I, Weimer M, Mahajan D, et al. Towards geo-distributed machine learning. 2016. ArXiv: 1603.09035

5 Hsieh K, Harlap A, Vijaykumar N, et al. Gaia: geo-distributed machine learning approaching lan speeds. In: Proceedings of USENIX Symposium on Networked Systems Design and Implementation, 2017. 629–647

6 McMahan H B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data. 2016. ArXiv: 1602.05629

7 Izmailov P, Podoprikhin D, Garipov T, et al. Averaging weights leads to wider optima and better generalization. 2018. ArXiv: 1803.05407

8 Povey D, Zhang X, Khudanpur S. Parallel training of dnns with natural gradient and parameter averaging. 2014. ArXiv: 1410.7455

9 Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014. ArXiv: 1409.1556

10 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2016. 770–778

11 Huang G, Liu Z, van Der Maaten L, et al. Densely connected convolutional networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2017

12 Sabour S, Frosst N, Hinton G E. Dynamic routing between capsules. In: Proceedings of Advances in Neural Information Processing Systems, 2017. 3856–3866

13 Tong Y, Chen Y, Zhou Z, et al. The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017. 1653–1662

14 Guo L H, Guo C G, Li L, et al. Two-stage local constrained sparse coding for fine-grained visual categorization. Sci China Inf Sci, 2018, 61: 018104

15 Chen C, Peng X, Sun J, et al. Generative API usage code recommendation with parameter concretization. Sci China Inf Sci, 2019, 62: 192103

16 Zhang H, Zheng Z, Xu S, et al. Poseidon: an efficient communication architecture for distributed deep learning on GPU clusters. 2017. ArXiv: 1706.03292

17 Shokri R, Shmatikov V. Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015. 1310–1321

18 Zinkevich M, Weimer M, Li L, et al. Parallelized stochastic gradient descent. In: Proceedings of Advances in Neural Information Processing Systems. 2010. 2595–2603

19 Alistarh D, Grubic D, Li J, et al. Qsgd: communication-efficient sgd via gradient quantization and encoding. In: Proceedings of Advances in Neural Information Processing Systems. 2017. 1709–1720

20 Lin Y, Han S, Mao H, et al. Deep gradient compression: reducing the communication bandwidth for distributed training. 2017. ArXiv: 1712.01887

21 Anil R, Pereyra G, Passos A, et al. Large scale distributed neural network training through online distillation. 2018. ArXiv: 1804.03235

22 Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. 2015. ArXiv: 1503.02531

23 Su H, Chen H. Experiments on parallel training of deep neural network using model averaging. 2015. ArXiv: 1507.01239

24 Sun S, Chen W, Bian J, et al. Ensemble-compression: a new method for parallel training of deep neural networks. In: Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2017. 187–202

25 Goodfellow I J, Vinyals O, Saxe A M. Qualitatively characterizing neural network optimization problems. 2014. ArXiv: 1412.6544

26 Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res, 2014, 15: 1929–1958

27 Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. 2015. ArXiv: 1502.03167

28 Russakovsky O, Deng J, Su H, et al. Imagenet large scale visual recognition challenge. Int J Comput Vis, 2015, 115: 211–252

29 Sainath T N, Parada C. Convolutional neural networks for small-footprint keyword spotting. In: Prceedings of the 16th Annual Conference of the International Speech Communication Association, 2015

30 Gemmeke J F, Ellis D P, Freedman D, et al. Audio set: an ontology and human-labeled dataset for audio events. In: Proceedings of 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017. 776–780

31 Greff K, Srivastava R K, Koutnik J, et al. LSTM: a search space odyssey. IEEE Trans Neural Netw Learn Syst, 2017, 28: 2222–2232

32 Hinton G, Frosst N, Sabour S. Matrix capsules with em routing. In: Proceedings of the 6th International Conference On Learning Representations, 2018

33 Bojanowski P, Grave E, Joulin A, et al. Enriching word vectors with subword information. Trans Assoc Comput Linguist, 2017, 5: 135–146

34 Yu C, Barsim K S, Kong Q, et al. Multi-level attention model for weakly supervised audio classification. 2018. ArXiv: 1803.02353