Communication Efficient Distributed Learning Over Wireless Channels

Idan Achituve, Wenbo Wang, Senior Member, IEEE, Ethan Fetaya, and Amir Leshem, Fellow, IEEE

Abstract—Vertically distributed learning exploits the local features collected by multiple learning workers to form a better global model. However, data exchange between the workers and the model aggregator for parameter training incurs a heavy communication burden, especially when the learning system is built upon capacity-constrained wireless networks. In this letter, we propose a novel hierarchical distributed learning framework, where each worker separately learns a low-dimensional embedding of their local observed data. Then, they perform communication-efficient distributed max-pooling to efficiently transmit the synthesized input to the aggregator. For data exchange over a shared wireless channel, we propose an opportunistic carrier sensing-based protocol to implement the max-pooling of the output of all the workers. Our simulation experiments show that the proposed learning framework is able to achieve almost the same model accuracy as the learning model using the concatenation of all the raw outputs from the learning workers while significantly reducing the communication load.

Index Terms—Learning over wireless channels, max-pooling, opportunistic carrier sensing, vertically distributed learning.

I. INTRODUCTION

IN CONTRAST to horizontal distributed or federated learning [1], where a dataset is partitioned among local learning workers, in a typical vertical learning system (see the examples in [2]), multiple learning parties hold either correlated views or disjoint parts of the same global input data for all samples. For example, in a wireless sensor network for smart surveillance, a number of security cameras with overlapping fields of the same region of interest may work together with several acoustic sensors to identify the behavior pattern of an intruder based on the fusion of both visual and acoustic features. Unlike the traditional centralized approaches of multi-view/multi-modal learning (e.g., [3]), a vertically distributed learning system is expected to perform most of the computations locally at each individual worker, while only sending the minimal necessary information for data/decision fusion to the leader node or the data fusion center. By doing so, the efficiency of data communication between the learning workers and the server can be significantly improved, and better data privacy can be preserved on the worker side.

Nowadays, techniques ranging from the simple fusion of local learning decisions by the distributed workers [3] to model and loss function partition [4], [5] are all considered possible options in the investigation of an efficient vertical distributed learning framework. In practice, it has been recognized that for general distributed computing tasks, the computation load and the communication load are inversely proportional to each other [6]. As a result, in our context, the goals of short local computation/training time (i.e., efficiency), low inter-node communication load (i.e., scalability), and high learning accuracy cannot be simultaneously achieved. Under the real-world constraint on the inter-node communication capacity, this tradeoff requires the size of the local models at each worker to be properly selected, and the data exchange pattern in the server-worker hierarchy to be carefully designed. For this reason, we propose a novel framework of vertical distributed learning based on max-pooling. More specifically, instead of applying the traditional encoding techniques, such as compression at the raw local data level [7] and sparsification at the gradient level [8], we apply the max-pooling operation to combine the intermediate data-representation output by each worker [9] and employ the opportunistic carrier sensing protocol to reduce the wireless transmission load between the learning workers and the fusion center. Our experiments show that for a learning system of \( N \) workers, the proposed learning scheme achieves almost the same inference performance as the learning algorithm that uses the concatenated data from all the workers while consuming only \( O(\log N/N) \) of the total communication load required by collecting the full data from all workers. Moreover, typically, the collision is detected during the first RTS transmission which is part of most CSMA protocols, so only a preamble is transmitted and payload data is not transmitted, so the actual saving is closer to \( 1/N \), since the RTS packets are shorter.

II. DESIGN OF THE LEARNING FRAMEWORK

A. System Model

Consider a distributed learning system that is constructed upon an IoT network (as illustrated in Fig. 1). The system consists of \( N \) wireless devices (e.g., smart sensors) that provide multiple views or sliced observations/features of the same phenomenon, relying on a shared wireless channel for inter-node communication. A single fusion center is responsible for synchronizing \( N \) distributed learning workers and aligning their locally observed data to identical labels. The concatenation of
the local raw data from the $N$ workers after data alignment creates a global raw feature $x = [x_1^T, \ldots, x_N^T]^T$. Rather than directly transmitting the raw data to the fusion center, worker $n$ trains its local model to produce a low-dimensional embedding $h_n = f_n(x_n; \theta_n)$, which is parameterized by $\theta_n$. Without loss of generality, we assume that worker $n$ deploys a Neural Network (NN) of $L$ layers with a parameter set $\theta_n = \{\omega_n^l\}_{l=1}^L$, where $\omega_n^l$ represents either a convolutional operator or the connection weights (including the biases) to be learned for the $l$-th layer. Then, $h_n$ can be obtained using the following transformation:

$$h_n(x_n; \theta_n) = (\omega_n^L)^T \sigma_L \left( \left( \omega_n^{L-1} \right)^T \sigma_{L-1} \left( \ldots \sigma_1 \left( \left( \omega_n^1 \right)^T x_n \right) \right) \right),$$

where $\sigma_l$ is a non-linear operation such as a Rectified Linear Unit (ReLU) activation or a hyperbolic tangent function.

The shared model at the fusion center can be expressed in a similar way to (1) with a parameter set $\theta$ and the concatenated worker outputs $h = [h_1^T, \ldots, h_N^T]^T$ as $\hat{y} = f_0(h; \theta_0)$. Let $y$ denote the label that is corresponding to $x$, held by the fusion center. For a given set of training samples and their corresponding labels $\mathcal{X} = \{[x^{(m)}, y^{(m)}] \}_{m=1}^{|\mathcal{X}|}$, the designing goal of the hierarchical framework for distributed learning is to minimize the expected difference between the inference by the fusion center and the labels as follows:

$$\left(\theta_0^*, \ldots, \theta_N^*\right) = \arg \min_{\theta_0, \ldots, \theta_N} \mathbb{E}_{x,y} \left[ \sum_{m=1}^{|\mathcal{X}|} \mathcal{L}(y^{(m)}, \hat{y}^{(m)}) \right],$$

where $\mathcal{L}(\cdot)$ is the loss function for the specific learning task at the fusion center.\(^2\) For regression problems and classification tasks, we can use

$$\mathcal{L}(y, \hat{y}) = ||y - \hat{y}(h)||^2,$$

$$\mathcal{L}(y, \hat{y}) = - \sum_{c \in C} 1_{y = c} \log \Pr(\hat{y} = c|h; \theta_0),$$

respectively, where $1_{a=b}$ denote the 0/1 indicator function.

\(^1\)Similarly [10], [11] use a linear transform.

\(^2\)For simplicity we ignore the index $m$ of the samples in what follows.

B. Feature Aggregation Based on Max Pooling

Adhering to the system model, we utilize max-pooling for the purpose of reducing features communicated to the fusion center. We assume that each worker observes local $d$-dimensional raw features $x_n \in \mathbb{R}^d$, while the output of each worker is a $K$-dimensional feature vector representing a low-dimensional description of the local raw feature. Sending all these features over the air would result in the transmission of $O(NK)$ messages. To reduce the communication even further, the max-pooling operation is employed at the fusion center to aggregate the intermediate outputs of all the workers, $h = [h_1^T, \ldots, h_N^T]^T$, as the input of the shared model $f_0(\cdot; \theta_0)$. Specifically, max-pooling produces a single $K$-dimensional feature, denoted by $v = [v_1, \ldots, v_K]^T$, by selecting the maximum value in each dimension out of the $N$ local features fed in by the workers, with

$$v_k = \max_{n \in \{1, \ldots, N\}} \{h_{n,k} \}, \forall k = 1, \ldots, K. \quad (4)$$

Naively implementing (4) would require $O(NK)$ transmission, but as we will show we can design a protocol to reduce it to $O(K)$. With (4), it is possible for a worker $n$ to send only its winning elements in $h_n$ for max-pooling to the fusion center, instead of sending the entire output feature vector. Namely, for each feature dimension $k$, only one worker is required to transmit its corresponding feature element $k$. By doing so, the forward communication load from the workers to the server can be reduced from $O(NK)$ to $O(K)$ in terms of the total dimension of the output features. In Section III, we will present a distributed mechanism that implements the pooling operation by providing access to the shared channel only to the worker with the highest value of the same feature element.

Max-pooling can also save the communication cost from the fusion center to the workers during the process of gradient backpropagation (i.e., backward transmission in Fig. 1). Following the standard backpropagation procedure, for the $n$-th worker ($n = 1, \ldots, N$) we have (see also (2) and (3))

$$\frac{\partial L}{\partial \theta_n} = \frac{\partial L}{\partial v} \frac{\partial v}{\partial h_n} \frac{\partial h_n}{\partial \theta_n},$$

By (4), $\forall k = 1, \ldots, K$ we have $\frac{\partial v}{\partial h_n} \in \mathbb{R}^{K \times K}$ and $\forall j = 1, \ldots, K,$

$$\left[ \frac{\partial v}{\partial h_n} \right]_{j,k} = \begin{cases} 1, & \text{if } n = \arg \max \{h_{1,k}, \ldots, h_{N,k}\}, j = k \\ 0, & \text{otherwise}. \end{cases}$$

Therefore, the fusion center needs only to broadcast the vector $\frac{\partial L}{\partial h_n}$ once to all the workers or send only the elements to the corresponding winning workers according to (4).

III. MAX-POOLING USING OPPORTUNISTIC CARRIER SENSING

To effectively reduce the communication loads required for obtaining (4), we apply Opportunistic Carrier Sensing (OCS) [12] over the shared wireless channel for max-pooling over the $N$ local embedding features. For multiple channels, such as in the case of Orthogonal Frequency-Division Multiple Access (OFDMA), multi-channel OCS can be applied similarly to the approach outlined in [13], [14]. The crux of OCS is to map the $k$-th element of each feature into a worker’s backoff time...
Algorithm 1: Max-Pooling Through OCS by Worker \( n \) (in Parallel) at Sub-Frame \( k = 1, \ldots, K \).

1: Sense after backoff in \( \text{BIN}_1(\mathbf{h}_{n,k}) \in \{0, 1\} \) sub-slot
2: if a blocking signal is detected then
3: \( \text{exit} \) \{negative values quit the contention as a loser\}
4: end if
5: for all \( d = 2, \ldots, D \) do
6: Sense after backoff in \( \text{BIN}_d(g(\mathbf{h}_{n,k})) \in \{0, 1, \ldots, D - 1\} \) sub-slot
7: if a blocking signal is detected then
8: \( \text{exit} \) \{quit the contention as a loser\}
9: else
10: \( \text{if} \) ACK = 0 then \{No ACK is received\}
11: \( \text{continue} \) \{continue the contention in sub-slot \( d + 1 \}\)
12: \( \text{else if} \) ACK = \( n \) then
13: Transmits \( \mathbf{h}_{n,k} \) to the server \{winning the contention\}
14: else \{some other worker receives the ACK\}
15: \( \text{exit} \) \{quit the contention as a loser\}
16: end if
17: end if
18: end for
19: while a blocking signal is not detected do
20: Randomly choose the back-off decision from \( \{0, 1\} \)
21: end while

in the corresponding sub-frame of contention resolution. Then, only the first node (winner) completing the backoff needs to send the corresponding feature element to the server in the subsequent data transmission sub-frame. Based on (4), we consider that worker \( n \) chooses a backoff period \( t_{n,k} = g(\mathbf{h}_{n,k}) \) for the \( k \)-th element of its output feature. Here, \( g(z) \) refers to a predetermined, strictly decreasing function in \( z \). Without loss of generality, we consider a non-negative feature element encoded in a binary floating-point format with a length of \( \log_2(D) \) bits.\(^3\) Thus, the inequality between two encoded values can be determined through bit-wise comparison. We define the backoff function for \( \mathbf{h}_{n,k} \) as \( g(\mathbf{h}_{n,k}) = D - \text{INT}(\mathbf{h}_{n,k}) \), where the operation \( \text{INT}(\cdot) \) forces the integer-based view of a given floating-point field.

Algorithm 1 describes the OCS-based max-pooling protocol from a worker’s perspective. It comprises of \( K \) sub-frames, each corresponding to an element in the features to be aggregated. Each sub-frame contains \( D \) sub-slots for backoff contention. The operation \( \text{BIN}_d(g(\mathbf{h}_{n,k})) \) (see Line 6, \( d = 1, \ldots, D \)) outputs the binary view of \( g(\mathbf{h}_{n,k}) \) at the \( d \)-th digit. \( \text{BIN}_d(g(\mathbf{h}_{n,k})) = 1 \) indicates that worker \( n \) backoffs at sub-slot \( d \) of sub-frame \( k \). A worker stops in the future sub-slots when detecting other transmissions (blocking signals). The workers who are able to transmit the blocking signal but receive no ACK signal identify collisions in contention and continue their backoff in the next sub-slot. For each sub-frame \( k \), it is very likely that a winner is determined after \( D \) sub-slots. However, to resolve potential ties after \( D \) sub-slots, the remaining contending workers continue in a random number of sub-slots by randomly generating backoff decisions in \( \{0, 1, \ldots, D - 1\} \) (see Lines 19-21 in Algorithm 1). The expected number of random back-offs per sub-frame is bounded by \( O(\log_D(N)) \) (see e.g., [16] for \( D = 2 \)). Consequently, the time complexity of the OCS-based max-pooling process is bounded in the worst case by \( O(K \log_D(N)) \).

IV. SIMULATION RESULTS

We evaluate the performance of our proposed distributed learning framework in two different scenarios. The first is a noisy reconstruction task on the MNIST dataset [17], and the second is a discriminative task on CUB-200-2011 [18], CIFAR-10 and CIFAR-100 datasets [19]. The latter ones are widely used in federated learning (e.g., [20] that is built on [21]). We use the default train-test splits of these datasets. We also use a validation set for hyper-parameter tuning and early stopping-based samples allocated from the training set of each dataset. For the MNIST and CIFAR datasets, we assign 10% from the training set to the validation set, and for the CUB dataset, we use 5 images per class. Since the OCS method results in a single payload transmission per sub-frame, we assume that optimal coding is applied and error-free transmission is achieved.

A. Handwritten Digit Reconstruction

One common scenario of vertical distributed learning with shared labels can be found in data reconstruction from multiple wireless sensors. It involves different lossy sampling processes, each causing distortion of the same signals at each sensor. To assess our method, we generated different local views of sensors based on synthetic data from the MNIST dataset. In our experiment (see Fig. 2), we assumed that \( N = 4 \) sensors independently sample a source image. Each sensor’s local sample is scaled to the range \([0,1]\) and corrupted with independent noise drawn from a Gaussian distribution with a fixed standard deviation of \( \sigma = 2 \). The model aggregator reconstructed the clean image from the local image representation learned by the sensors. We refer to the NN on each worker as an “encoder” and the NN on the model aggregator as the “decoder”. Each encoder samples a flattened view of a noisy MNIST image of size 784 and then transforms it to an embedding vector of size 64 using

\(^3\)Following the IEEE 745 standard, with a fixed number of exponent and significand bits [15] packed from left to right with sign included.
the local NN. The decoder reconstructs the image of the original dimension from the low-dimensional embedding vectors output by the workers. We deploy small-size networks for both the encoders and decoder and adopt the OCS-based max-pooling protocol between them. Each encoder adopts an NN of three hidden layers with sizes \{512, 256, 128\}. The decoder adopts an NN of three hidden layers with sizes \{128, 256, 512\}. Fig. 2 compares the reconstruction quality of 64 images in a system with one worker compared to a system with 4 workers. From the figure, the quality of reconstruction in the latter is significantly better even though the two systems communicate the same number of parameters over the wireless channel. Quantitatively, the average negative log-likelihood of the system with 4 workers is 0.13, while that with one worker is 0.19.

B. Prediction Using Patches of a Global Image

Our second experiment focuses on a scenario where the wireless sensors sample disjoint parts of the same global data. To simulate this scenario, we partitioned images from CUB-200-2011, CIFAR-10, and CIFAR-100 to a grid of 4 and 9 cells, respectively, and assigned to each worker a fixed unique cell for sensing. Each worker uses an NN to generate a low-dimensional embedding of its locally observed cell. The fusion center then aggregates the embeddings of all workers using a different NN ("classification head") to learn a distribution over the classes. We used MobileNetV2 [22] as a feature extractor on each worker since it balances well between good model performance and low computational load. The fusion center adopts a fully connected NN with 3 hidden layers of sizes \{512, 512, 512\}.

For efficiency evaluation, we compared the performance of the following methods:

1) **Avg. Workers Preds:** Each worker generates a separate class distribution. The fusion center then averages over all the local probability vectors to obtain a global distribution over the classes.

2) **Best Worker Pred:** The model of the best performance among the independent local models in (1).

3) **Avg. Workers Embed:** An alternative pooling method that averages over each dimension of the embedding output across all the workers. The new representation is used as the input to the classification head at the server. This method requires each worker to send its full embedding feature to the server.

4) **Concat Workers Embed:** The outputs of all the workers are concatenated to form an input to the classification head. This is reminiscent of the vertical federated learning methods [23] with excessive communication loads.

5) **FedOCS:** Our proposed method using max-pooling for feature aggregation.

The results are presented in Table I, which reports the accuracy and the standard deviation based on three runs with random seeds. Additionally, Fig. 3 reports the test accuracy on the CUB-200-2011 dataset at each training step for methods (3)–(5). From Table I, we notice that aggregating information in the feature space is considerably more effective compared to the individual baselines in which each client makes a separate decision. Aside from the communication efficiency of our method compared to the baseline that performs mean-pooling, FedOCS manages to maintain the same accuracy on CIFAR-10 and even surpass it on the harder classification tasks of CIFAR-100 and CUB-200-2011 images. This is further pronounced in Fig. 3 which shows that FedOCS converges faster than this baseline. Finally, and quite remarkably FedOCS is often comparable to the baseline which concatenates the embedding representation. This baseline can be considered as an upper-bound for FedOCS performance, yet incurring a much larger communication overhead.

| TABLE I |
| --- |
| **TEST ACCURACY (±STD) WITH 4/9 WORKERS, RESPECTIVELY** |
| No. of clients | CIFAR-10 | CIFAR-100 | CUB-200-2011 |
| 4 | 9 | 4 | 9 | 4 | 9 |
| Concat Workers Embed | 84.35 ± 0.5 | 81.02 ± 0.1 | 53.06 ± 0.2 | 51.05 ± 0.5 | 35.73 ± 1.2 | 30.89 ± 0.5 |
| Best Worker Pred | 79.31 ± 0.6 | 44.87 ± 1.3 | 98.76 ± 1.4 | 79.87 ± 1.6 | 88.59 ± 0.3 | 6.96 ± 0.3 |
| Avg. Workers Preds | 76.03 ± 0.6 | 61.60 ± 0.6 | 31.14 ± 0.7 | 16.89 ± 1.5 | 18.39 ± 0.6 | 7.93 ± 0.1 |
| Avg. Workers Embed | 84.28 ± 0.2 | 80.58 ± 0.5 | 50.80 ± 0.3 | 45.78 ± 0.3 | 34.84 ± 0.7 | 29.99 ± 0.3 |
| FedOCS | 84.48 ± 0.7 | 80.52 ± 0.3 | 53.98 ± 0.7 | 49.79 ± 0.2 | 36.34 ± 0.6 | 31.39 ± 0.5 |

The numbers in bold are the statistically significant best results.

Fig. 3. Test accuracy per step on CUB-200-2011 dataset.
REFERENCES

[1] C. Xu, S. Liu, Z. Yang, Y. Huang, and K.-K. Wong, “Learning rate optimization for federated learning exploiting over-the-air computation,” *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3742–3756, Dec. 2021.

[2] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, “A survey on distributed machine learning,” *ACM Comput. Surv.*, vol. 53, no. 2, Mar. 2020, Art. no. 30.

[3] S. Poria, E. Cambria, and A. Gelbukh, “Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis,” in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 2539–2544.

[4] K. Cheng et al., “SecureBoost: A lossless federated learning framework,” *IEEE Intell. Syst.*, vol. 36, no. 6, pp. 87–98, Nov./Dec. 2021.

[5] M. M. Amiri and D. Gündüz, “Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air,” *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, 2020.

[6] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, “A fundamental tradeoff between computation and communication in distributed computing,” *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan. 2018.

[7] Z. Li, D. Kovalev, X. Qian, and P. Richtarik, “Acceleration for compressed gradient descent in distributed and federated optimization,” in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 5895–5904.

[8] L. Li, D. Shi, R. Hou, H. Li, M. Pan, and Z. Han, “To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices,” in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[10] M. Gastpar, P. L. Dragotti, and M. Vetterli, “The distributed Karhunen–Loève transform,” *IEEE Trans. Inf. Theory*, vol. 52, no. 12, pp. 5177–5196, Dec. 2006.

[11] L. Gispan, A. Leshem, and Y. Be’ery, “Decentralized estimation of regression coefficients in sensor networks,” *Digit. Signal Process.*, vol. 68, pp. 16–23, 2017.

[12] Q. Zhao and L. Tong, “Opportunistic carrier sensing for energy-efficient information retrieval in sensor networks,” *EURASIP J. Wireless Commun. Netw.*, vol. 2005, no. 2, pp. 1–11, 2005.

[13] A. Leshem, E. Zehavi, and Y. Yaffe, “Multichannel opportunistic carrier sensing for stable channel access control in cognitive radio systems,” *IEEE J. Sel. Areas Commun.*, vol. 30, no. 1, pp. 82–95, Jan. 2012.

[14] S. Zafaruddin, I. Bistritz, A. Leshem, and D. Niyato, “Distributed learning for channel allocation over a shared spectrum,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2357–2349, Oct. 2019.

[15] N. Brisebarre, C. Lauter, M. Mezzarobba, and J.-M. Muller, “Comparison between binary and decimal floating-point numbers,” *IEEE Trans. Comput.*, vol. 65, no. 7, pp. 2032–2044, Jul. 2016.

[16] T. Boyarski, W. Wang, and A. Leshem, “Distributed learning for optimal spectrum access in dense device-to-device ad-hoc networks,” *IEEE Trans. Signal Process.*, vol. 71, pp. 3149–3163, 2023.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[18] P. Welinder et al., “Caltech-UCSD birds 200,” California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CNS-TR-2011-001, 2010.

[19] A. Krizhevsky et al., “Learning multiple layers of features from tiny images,” Univ. Toronto, Toronto, Canada, Tech. Rep., 2009. [Online]. Available: http://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[20] I. Achituve, A. Shamsian, A. Navon, G. Chechik, and E. Fetaya, “Personalized federated learning with Gaussian processes,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 8392–8406.

[21] I. Achituve, A. Navon, Y. Yemini, G. Chechik, and E. Fetaya, “GP-Tree: A Gaussian process classifier for few-shot incremental learning,” in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 54–65.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

[23] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, Jan. 2019, Art. no. 12.