Learning from Peers at the Wireless Edge

Shuvam Chakraborty*, Hesham Mohammed* and Dola Saha
Department of Electrical & Computer Engineering
University at Albany, SUNY, Albany, NY 12222 USA
{schakraborty, hhussien, dsaha} @albany.edu
*co-First Authors

***Abstract*—The last mile connection is dominated by wireless links where heterogeneous nodes share the limited and already crowded electromagnetic spectrum. Current contention based decentralized wireless access system is reactive in nature to mitigate the interference. In this paper, we propose to use neural networks to learn and predict spectrum availability in a collaborative manner such that its availability can be predicted with a high accuracy to maximize wireless access and minimize interference between simultaneous links. Edge nodes have a wide range of sensing and computation capabilities, while often using different operator networks, who might be reluctant to share their models. Hence, we introduce a peer to peer Federated Learning model, where a local model is trained based on the sensing results of each node and shared among its peers to create a global model. The need for a base station or access point to act as centralized parameter server is replaced by empowering the edge nodes as aggregators of the local models and minimizing the communication overhead for model transmission. We generate wireless channel access data, which is used to train the local models. Simulation results for both local and global models show over 95% accuracy in predicting channel opportunities in various network topology.**

I. Introduction

Exponential increase [1] in data capacity requirement for emerging applications can only be sustained by efficient usage of electromagnetic spectrum by a variety of heterogeneous devices. Efforts have been made to open up new spectrum, while several unlicensed and semi-licensed models have been proposed for a shared usage. Although, multiple operators will prevail for licensed access, large swaths of frequencies will be available for unlicensed use for different protocols. We envision that future intelligent wireless networks will be able to make distributed decisions on wireless channel access without any aid from the centralized base station.

Distributed wireless channel access is performed using carrier-sense and backoff mechanisms as in Wi-Fi [2]. As the system only reacts to collisions, much of the time is wasted in sensing, backoff and collisions as the number of nodes in the system increases [3]. If an accurate collaborative prediction system is appointed, we will notice a better usage of the available spectrum. Machine learning based wireless systems have received attraction in recent years to learn hidden parameters in a system, which are difficult to model. In this scenario, traditional machine learning approaches require centralizing the training data and inference processes on a single data center. Due to the propagation characteristics of radio frequencies, wireless channel is inherently distributed, and has to be measured and learned at each node for optimum performance. A base station's view of wireless channel could be completely different from a mobile terminal's view, specially when they are spatially separated. Hidden terminal problems cannot be solved by a centralized entity when multiple parties share the radio frequency spectrum. At the same time, many channel properties may overlap, which can be similar in the vicinity and learned from neighbors. Sensing at the mobile terminals and sending the data to the base station is infeasible because of communication costs. Moreover, there are trust and privacy issues, which deters the operators to share their data. With these challenges, we design our protocol to predict wireless channel availability in a distributed wireless network.

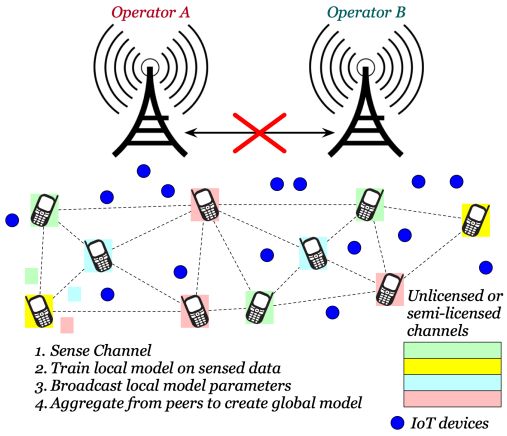


Fig. 1: Future Intelligent Wireless Network.

Figure 1 shows the last mile future network, which mainly will constitute 1) Base stations, operated by different operators providing Internet access to mobile terminals and IoT devices, 2) IoT devices, densely deployed and often with limited sensing and computation capabilities, 3) Mobile terminals, including smartphones, tablets etc., which are capable of sensing and transmitting in a wide variety of frequency bands. Also, they may not have the capability to monitor all the available channels all the time, but should be able to use any of the channels when the transmission opportunity exists. One of the major issues of decentralized wireless access is also the hidden terminal problem, which cannot be mitigated by sensing at the transmitter. This requires learning channel availability at the intended receiver (one hop neighbor) and cannot be a localized decision. This is precisely the reason where we deploy Federated Learning [4], [5] to predict the

channel availability in each node. The first step of our system is to sense the channel, which the mobile terminals choose depending on any specified criteria. Based on the sensing data, it trains a local neural network model to predict the channel availability. Then, it broadcasts the trained local model parameters to its neighbors using a shared control channel. These neighbors can be connected to different operators, but can form an overlay network with peers to share learned models over a common unlicensed channel. Once a node receives local models from its neighbors, it 1) concatenates the models for which it does not have the data and 2) aggregates the model by averaging the model parameters from its neighbors. The first case helps a node to learn channel availability quickly from neighbor, which it has not sensed and thus does not have a local model. The second case addresses the hidden terminal problem by considering channel prediction models of it's one-hop neighbors.

Our protocol does not require a centralized parameter server, since we deploy global model aggregation at each node. Channel availability prediction does not require to propagate multiple hops as it depends on the interference that a transmitter can create at another intended receiver. Hence, the global model is also small enough to be implemented at the edge. It is to be noted here that current smartphones already deploy neural network models in GPUs or neural processors for efficient image processing. Hence, our assumption of deploying a local neural network model on these smartphones at the edge of the network is quite realistic.

II. Related work

Federated learning [4] was proposed to increase communication efficiency where the entire data-set is not readily available to the central server and mobile nodes have a small fraction of that data available to them. They use the local data to learn the local model and share only the model parameters with parameter server (PS). The model parameters are aggregated in the centralized PS to generate the global model, which is shared with the mobile nodes. There has been numerous applications of federated learning to model various aspects of wireless systems, none of those have attempted to make the system completely decentralized removing the need of any parameter server. Authors in [6] proposed a model segment level decentralized federated learning to pull the models from participating nodes. Authors have taken a segmented update approach in [7], [8], which in spite of being a fully decentralized approach, needs number of nodes for each gossip segment to be precisely defined for most efficient model update. This is not needed in our system design. [9] explores the effect of varying number of nodes updating simultaneously to the parameter server. A federated learning approach for packet classification has been discussed in [10], which also requires parameter server to aggregate the model. [11]–[17] incorporates various applications and model updates for Federated learning, which uses base stations as the parameter server. A peer to peer model of federated learning is proposed in [18] where the authors assumed the data is available to each mobile nodes. Also, there is an assumption that the data is fully orthogonal or uncorrelated. On the contrary, in our system, the neighbors in close proximity will have highly correlated data based on channel sensing. Hence, none of the above mentioned models or solutions can be applied directly in our system.

III. Background

In this section we describe the concept of federated learning and why it is so well suited for our problem of channel sensing and prediction. Federated Learning enables distributed devices to collaboratively learn a shared prediction model while keeping all the training data on the device. Once trained, the updated parameters are aggregated in a centralized parameter server to create a global model. Assuming n nodes are present in a network, and θ_i is the local model parameter matrix of the node i, then the aggregator creates a global model Θ as shown in equation 1.

$$\frac{1}{n}\sum_{i=1}^{n}\theta_i = \Theta \tag{1}$$

Any wireless communication system is inherently a distributed system. Conventional ML systems work on the assumption of having the entire data-set and processing capability available in a central server. It is not feasible in our case not only due to privacy reasons, but due to high volume of data that needs to be shared for training purposes yielding high communication costs. Consequently, decentralized approach is a lucrative solution that incurs minimum communication overhead and computation costs.

IV. Problem Formulation

In this section we demonstrate peer to peer based federated learning system for wireless networks for predicting channel availability. The notion of distributed learning regime lies in two possible scenarios: data parallelisation and model parallelisation. While federated learning predominantly exploits the data parallelisation by using the same training model with orthogonal or non-overlapping data-set. In wireless systems, the data might overlap, thus providing higher priority for the overlapping data, as this is informed by frequent appearance of relative parameter. We assume N number mobile nodes in the network, where i^{th} node is denoted by N_i. All the nodes are acting as wireless sensors denoted by set $\mathcal{N}$ and are participating in the distributed learning. Each node has their model generated from channel sensing results of that node itself, thus guaranteeing only a local view of the wireless system owing to limited visibility of the mobile nodes. Each local data-set is denoted by X_i and is accompanied by a label set Y_i, $i \in N$. Following our assumption that local data-sets have overlap in wireless networks, $\bigcap_{i=0}^{n} X_i \neq \emptyset$, where $\emptyset$ denotes the empty set.

Each node N_i generates a local parameter set θ_i, where i denotes the node identity. These parameters are shared among the neighboring mobile nodes. Only the model parameters from the local model are shared with other nodes in a broadcast signal, as it does not include any raw data from the

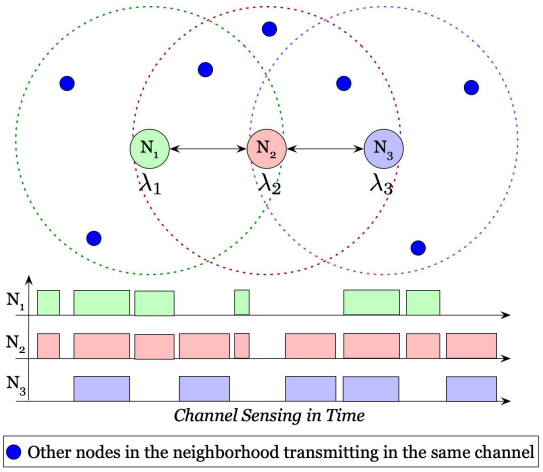


Fig. 2: Channel Sensing and Local Model Exchange in Hidden Terminal Scenario.

primary node leveraging the inherent data preserving nature of federated learning. We assume the local models implemented in the mobile nodes has access to the local likelihood functions that generate each local weight matrix or parameter matrix θ_i.

Based on our assumptions, the global parameter generated at the node N_i can be denoted as Θ_i, where

$$\|Y_i - X_i\theta_i\| = \eta_1 \quad \texttt{and} \quad \|Y_i - X_i\Theta_i\| = \eta_2$$

where $\eta_2 \leqslant \eta_1$ and $\|.\|$ is the L_2 norm. Since η_2 denotes the error rate in predicting channel availability while using updated global parameter, it should be equal or less than the error rate using local models η_1, because of limited channel information shared in local models.

V. Channel availability prediction Protocol

In traditional CSMA-CA system, channel availability is sensed by a mobile node for a short duration and if it senses the channel busy, it backs-off for a duration w randomly chosen from the contention window, which grows exponentially in every iteration if the node senses the channel to be busy. Furthermore low power mobile nodes can sense only one channel giving rise to uneven distribution of channel resource usage for each node. Figure 2 shows a hidden terminal scenario, where nodes N_1 and N_3 are hidden to each other when they sense the channel and transmit at the same time to create interference at the receiver, node N_2. Hence, sensing locally and learning on only local sensing data will not address the hidden terminal issue. It is important to capture node N_2's sensing information in the learning parameters of both N_1 and N_3. Thus, when local model θ_2 of node N_2 gets propagated to both its neighbors, channel availability at receiver N_2 is also incorporated in the aggregated model. It is to be noted here that sensing the channel creates a prediction for all the transmissions near that node. Hence, even when IoT devices or other nodes are not sensing the channel, their transmission characteristics are captured by one-hop neighbors who are sensing the channel.

A. Channel sensing

Multiple traffic arrival rates (multiple varying λs) are incorporated in the channel traffic model, where individual traffic arrival follows Poisson distribution and different possible arrivals are uniformly distributed in time. These traffic flows may be generated by one node or multiple nodes, but, when transmitted in a channel, is sensed by all neighbors which are sensing or receiving in that channel. For example, multiple IoT devices may generate different traffic rates, and might not be sensing the channel due to power constraints. However, a mobile node, if participating in sensing and collaboration, will sense the channel and observe it to be busy during the transmission period. For example, if multiple nodes around node N_1 in figure 2 generates traffic at different rates and transmits them, then sensing at N_1 will capture all those times as channel being busy. Thus, λ_1 is a combination of multiple traffic patterns. Also, there are multiple nodes that are common in one-hop neighborhood, thus there is a significant overlap of data among one-hop neighbors. The mobile nodes sense the channel for a small time period δ, where $\delta \ll L_{pkt}$, and L_{pkt} is the minimum packet transmission duration in the network. In each δ, if the mobile terminal senses the channel busy for any duration, it indicates the channel to be busy (denoted as 1) for that time, thus discretizing the channel output and generating a sequence of bits that encodes channel sensing result as a binary time-series.

B. Training Local Model

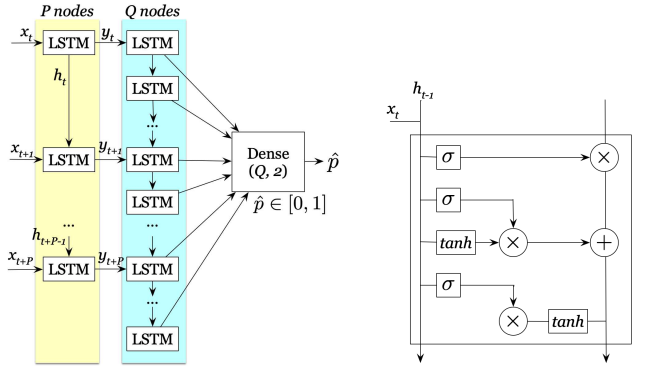
Each node N_i generates the time-series as channel sensing result, which includes channel activity sensed within the coverage area of this node. This time-series is mapped is mapped into one-hot code and fed to a two layer LSTM network, which generates the local parameter set θ_i for channel prediction depending on sensing data from node N_i only.

C. Model Sharing

Every node shares their locally learned parameter matrix θ_i, for node i as a broadcast packet, thus sharing its local model only to all one hop neighbors. These local models contain parameters learned from only the local limited view of the source node. The parameters learned from the neighbors are not shared, thus limiting the model propagation to one hop only.

D. Global Model Generation

All nodes $N_j, j \neq i, j \in (1, M)$ are sensing the same channel thus seeing a part of the same network traffic as the primary node N_i denoted by λ_i along with other network traffic denoted by $\lambda_1, \lambda_2, ..., \lambda_M$, where each of them are association of set of different arrival rates. In the figure 2 we can only see three nodes, but we will generalize our discussion here. In this scenario all local parameter set θ_l generated from $N_l, l \in M$ will carry the information of network traffic λ_i. Thus averaging, of the parameters will generate very high accuracy of the global model of node N_i. Averaging will reduce weights of network traffic contributed by λ_ks. Thus



(a) Neural Network Architecture. (b) LSTM cell.

Fig. 3: Neural Network architecture in each edge node.

setting $w_i = 1$ generates 98% accuracy for channel prediction of node N_i So the global model becomes,

$$\theta_i + \frac{1}{M}\sum_{l=1}^{M} w_l(\theta_l + \eta_l) = \Theta_i \tag{2}$$

where w_l=1, $l \in M$

E. Learning parameters for orthogonal channels

There might appear another scenario where nodes sense different channels even in a similar network as shown in Figure 2 consequently generating local model parameters that are entirely uncorrelated, thus aggregating parameters following any algebraic operation is not feasible. So to predict other channel availability we have to store the model parameters and generate a concatenated global model for channel prediction. Here as we are storing different models and since it is a multi-node update process, there needs to be an optimum number of shared models to be stored while dealing with memory constrained edge devices.

VI. Neural Network Architecture

Traditional neural networks (NN) (i.e., feed-forward networks and CNN) are not capable of learning data sequences such as text prediction since the output of the traditional NN depends on the current input and is given by:

$$Y_i = f(w, b, X_i) \tag{3}$$

where f is the NN activation function, w and b are the weights and biases of the NN, and X_i and Y_i are the input and the output respectively.

Recurrent NN (RNN) [19] solves this problem by making the output of the RNN depends on the current input and the input state. The input state relies on the history of the previous inputs and the outputs of the RNN. The output of the RNN is given by :

$$(Y_t, h_t) = f(w, b, X_t, h_{t-1}) \tag{4}$$

where h_t is the state of the RNN at time t. However the RNN can not capture the long term dependencies, since the output depends on only one cell state ash shown in (4). Long Short Term Memory networks (LSTMs) [20] are a special kind of RNNs, which are capable of capturing the sequence dependencies both long and short term.

A. Neural Network Structure

We use LSTMs in the local model at each edge node, the structure of which is shown in figure 3. As depicted in figure 3a the LSTM network has 2 layers including P nodes in the first layer and Q nodes in the second layer followed by a dense layer that generates the trained parameters. We have chosen a two layer LSTM network along with a dense layer before output, with neurons $\{P,Q\}$ pairs, where outputs from P neurons of input layer is mapped in an one to many fashion to neurons of second layer with Q elements, which ultimately generates a parameter set of dimension (Q,2).

Each cell of LSTM network is based on two main components. The first part is the conveyor belt, which shares the network history among all the LSTM cells. The second part is the component of the LSTM cell. The LSTM cell consists of three sigmoid functions, which act as three main gates as shown in figure 3b. The first gate allows the cell input update the conveyor belt. The second one decides whether the cell state is affected by both the conveyor belt (i.e the Network history) and the cell input or depends on the conveyor belt. The last one controls the cell output such that the cell output depends on both the input and the cell state or results from the cell state only.

B. Data mapping

In this work, we consider the channel state prediction which has two states, either idle or busy channel. This representation is not suitable for neural networks (NN) since they deal with the real numbers. So, we transform the data from binary representation to a supported data representation to construct the data set used to train the LSTM.

Let m be the set of all possible events. We use one-of-m representations for the channel state, given by

$$p_k = [1(x_k = Z_1), 1(x_k = Z_2), 1(x_k = Z_3), .., 1(x_k = Z_m)]^T \tag{5}$$

Therefore, the element corresponding to the event equals to 1 while the others are 0 (i.e one-hot encoding). In this work, we only consider one channel measurement at a time (i.e $m = 2$). However this representation is also valid if the node can sense more than one channel at a single time slot. Note that P_k can be considered as the probability mass function (PMF) of the event to happen.

The output layer of the LSTM is a softmax layer. The softmax layer output $\hat{p_k} \in [0, 1]$ is the probability vector of the transmitted message. The softmax function is given by:

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}} \tag{6}$$

where x_j is the component j in the vector x.

In the training phase, The LSTM updates its parameters in each training epoch to achieve the optimal parameter

$$w^* = \min L(\hat{p} = p/p) \tag{7}$$

where L is the loss function between the predicted and the actual states, which is similar to maximize log likelihood that acts as the best estimator.

C. Loss Function

The Loss function is used to adjust the weights and biases to map the LSTM prediction to the actual targets included in the training set. The optimization problem in (7) is solved by applying stochastic gradient descent (SGD) using a cross-entropy loss function which is given by:

$$L_{cross} = H(\hat{p}, p) = H(p) + D_{KL}(p||\hat{p}) \quad (8)$$

where H is the entropy, and $D_{KL}(||)$ is the Kullback- Leibler divergence [21]. Note that, minimizing L_{cross} or $D_{KL}(||)$ is equivalent to maximizing the likelihood between the predict probability and the actual occurrence of an event.

VII. Experiment and Evaluation

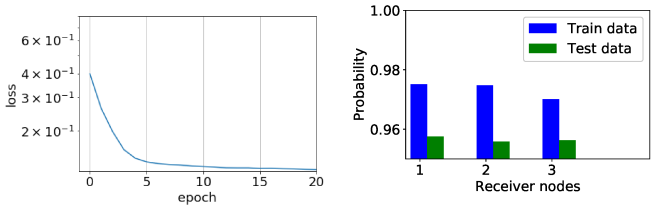
A. Experimentation Setup

We have generated the network topology in MATLAB assuming varying Poisson arrival rates with different packet sizes. For experimentation, we choose two different set-ups.The first set of experiment includes one primary node and three different neighbors, sensing a part of common network traffic denoted by average arrival rate λ_i that is sensed by primary node N_i as well, where as in the second set there are five neighbors introducing further variation to channel traffic. Motivation behind choosing such a setup was to demonstrate the local model and global model update of that primary node for any real network scenario with varying number of neighbors, where there will be some overlap of sensing data among the neighbors. The neighbouring nodes sense additional network traffic on the same channel denoted by λ_1, λ_2, etc. where each of these terms are associated with multiple different packet arrival rates. We have tested this case for five neighbouring nodes as well, incurring further variance and consequently higher mutual information between the nodes to learn.

We have chosen the maximum packet size to be equal to standard wireless TCP packet size 2312 bytes along with total 52 bytes of MAC and IP headers. The packet length varies between 2000 to 2364 bytes to emulate a real network traffic. Any edge sensor senses the channel for a duration of $20\mu s$, which is one of parameters of the DIFS (DCF interframe spacing) times in IEEE 802.11n standard. The sensing data is generated as a time-series from a seed with channel traffic distributed as Poisson distribution and effectively the channel idle time as an exponential distribution with labels 0 (denoting channel idle) and 1 (denoting channel busy) for a total channel sensing duration of *5 seconds*. All our experiments require only 5 seconds worth of data for the local model to be trained. No training is required after aggregation, essentially making it practical for deployment. This generates $250,000$ instances of data in the time-series for training the LSTM network, 10% of training data-set, generated using a different seed is used for validation and the entire training is implemented on an Intel NUC (NUC7i7BNH) with i7-7567U processor and 16GB DDR4 memory, without using any acceleration units. The different packet arrival rates used for experimental set-up has been shown in table I for all different experimental setup.

TABLE I: Packet arrival rates in different network topology

Network Topology	Arrival Rate of Primary Node	Arrival Rate of Neighbors
3 neighbors	$\lambda_i = 5.0$	$\lambda_1 = \{5.0, 9.5, 12.0\}$ $\lambda_2 = \{5.0, 8.6, 10.5\}$ $\lambda_3 = \{5.0, 16.0, 6.0\}$
5 neighbors	$\lambda_i = 5.0$	$\lambda_1 = \{5.0, 9.5, 12.0\}$ $\lambda_2 = \{5.0, 8.6, 10.5\}$ $\lambda_3 = \{5.0, 16.0, 6.0\}$ $\lambda_4 = \{5.0, 15.8, 21.0\}$ $\lambda_5 = \{5.0, 2.8, 13.0\}$



(a) Local model loss function. (b) Prediction accuracy.

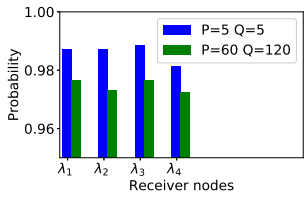
Fig. 4: Performance of local model (Node 1).

According to the figure 3a we have chosen two different sets of $\{P, Q\}$ pairs, {60, 120} and {5, 5}, which will be denoted in the following evaluation section as T_b and T_s.

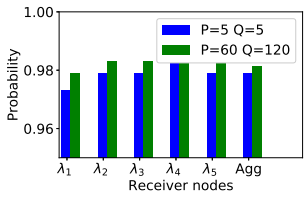
B. Evaluation

1) Performance of Local Model: Figure 4a shows the loss curve during the training of a local model for edge node 1. The other nodes' loss curves for local models overlap with it and hence is not shown in the graph. During training, network T_b reaches about 97.8% training accuracy in 20 epochs, which equals to approximate computation time of 250 seconds. It achieves 96% validation accuracy in predicting channel occupancy only with local model and this value stays same as demonstrated in figure 4b for three different nodes with their local models.

2) Performance of Global Model: We have tested accuracy of global models with three neighbors and five neighbors. Figure 5a shows the channel prediction accuracy for local models of each neighbors as well as that of the aggregated global model of Node 1 using both, network T_b and T_s. Leveraging accuracy of LSTM networks in prediction of sequential data, both the networks T_b and T_s are able to achieve about 98% validation accuracy. T_b required 40 epochs to generate weights that helps global model to predict the channel with an accuracy of 98.12%, which is equivalent to computation time of 250 seconds. With the same input data, T_s required 400 epochs, though total number of up-gradable neuron weights being much less it incurred a computation time of 240 seconds, which is comparable to T_b. We intend to use the smaller neural network, T_s, which achieves similar accuracy as the larger network, T_b, but requires much smaller footprint to be implemented in hardware and a smaller model update packet to be transmitted over the air to the neighbors, thus reducing communication overhead. These times reported in seconds do not use any acceleration units, like Neural Processing

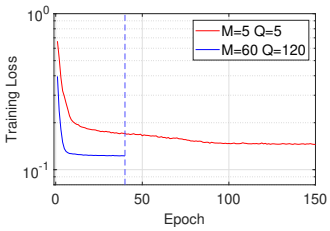


(a) Three Nodes

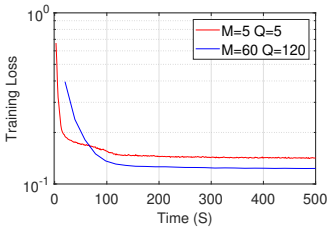


(b) Five nodes.

Fig. 5: Performance of global model for multiple nodes and different size model.



(a) Loss function with increasing number of epochs.



(b) Loss function with computation time.

Fig. 6: Loss function of two different model sizes for the same node (Node 1).

Units (NPUs) and Graphics Processing Units (GPUs), which are prevalent in current smartphones. In other words, when deployed in edge nodes, the computation time will be even less, thus making it a practical choice for channel availability prediction.

3) Effect of Size of The Neural Network: The two LSTM networks used T_b and T_s have similar structure, thus performing in the same way, but T_s is preferrable for the following properties:

- T_s has 392 parameters (in floating point) in local model yielding 1512 bytes, which needs to be transmitted over the air to share with neighbors. On the other hand, T_b has 102,252 parameters in local model yielding a size of 0.4 Megabytes. Since the local model has to be shared among peers, T_b incurs higher computation and communication overhead than T_s.
- As shown in figure 6a, T_s requires 400 epochs to reach the similar accuracy as of T_b, which it achieves in 40 epochs. But if we notice the computation time requirement, both networks converge at the same time as shown in figure 6, thus reaching equivalent accuracy in same computation time.
- Since our primary implementation is for edge nodes with power and hardware resource constraints, the smaller network is a natural preference due to lower footprint and computation requirement.

Thus even though in higher variance channel traffic T_b generates slightly better results there are better trade-offs to opt for T_s for deploying in edge nodes.

VIII. Conclusion and Future Work

In this paper, we have proposed a distributed framework for peer-to-peer based federated learning to reduce the need for centralized parameter server. Our evaluation shows it is highly effective in predicting channel availability in a wireless ad hoc network. Future work will require evaluation of the system in a larger network with a variety of channel access mechanism. Future exploration may include edge nodes to be able to sense disjoint channel properties and aggregate them for transfer learning.

References

[1] "Cisco visual networking index: Global mobile data traffic forecast update, 2014 –2019," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf.

[2] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Computer Society : LAN/MAN Standards Committee. [Online]. Available: http://standards.ieee.org/getieee802/download/802.11-2007.pdf

[3] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, March 2000.

[4] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. [Online]. Available: https://arxiv.org/abs/1610.05492

[5] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," *CoRR*, vol. abs/1902.00146, 2019. [Online]. Available: http://arxiv.org/abs/1902.00146

[6] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," 2019.

[7] D. Liu, G. Zhu, J. Zhang, and K. Huang, "Wireless data acquisition for edge learning: Importance-aware retransmission," in *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, July 2019, pp. 1–5.

[8] G. J. Mendis, M. Sabounchi, J. Wei, and R. Roche', "Blockchain as a service: An autonomous, privacy preserving, decentralized architecture for deep learning," *CoRR*, vol. abs/1807.02515, 2018. [Online]. Available: http://arxiv.org/abs/1807.02515

[9] N. H. Tran, W. Bao, A. Zomaya, N. Minh N.H., and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, April 2019, pp. 1387–1395.

[10] E. Bakopoulou, B. Tillman, and A. Markopoulou, "A federated learning approach for mobile packet classification," 2019.

[11] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," 2019.

[12] Z. Chai, H. Fayyaz, Z. Fayyaz, A. Anwar, Y. Zhou, N. Baracaldo, H. Ludwig, and Y. Cheng, "Towards taming the resource and data heterogeneity in federated learning," in *2019 USENIX Conference on Operational Machine Learning (OpML 19)*. Santa Clara, CA: USENIX Association, May 2019, pp. 19–21. [Online]. Available: https://www.usenix.org/conference/opml19/presentation/chai

[13] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," 2018.

[14] H. H. Yang, Z. Liu, T. Q. S. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," 2019.

[15] J.-H. Ahn, O. Simeone, and J. Kang, "Wireless federated distillation for distributed edge learning with heterogeneous data," 2019.

[16] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning (extended version)," 2018.

[17] T. T. Vu, D. T. Ngo, N. H. Tran, H. Q. Ngo, M. N. Dao, and R. H. Middleton, "Cell-free massive mimo for wireless federated learning," 2019.

[18] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," *CoRR*, vol. abs/1901.11173, 2019. [Online]. Available: http://arxiv.org/abs/1901.11173

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[20] colah's blog. (2015) Understanding lstm networks. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[21] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing, 2nd Edition)*. Wiley-Interscience, July 2006.