Demonstration of Federated Learning in a Resource-Constrained Networked Environment

Dave Conway-Jones*, Tiffany Tuor†, Shiqiang Wang‡, Kin K. Leung†
*IBM Research, IBM UK, Hursley Park, UK. Email: conway@uk.ibm.com
†Imperial College London, UK. Email: {tiffany.tuor14, kin.leung}@imperial.ac.uk
‡IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. Email: wangshiq@us.ibm.com

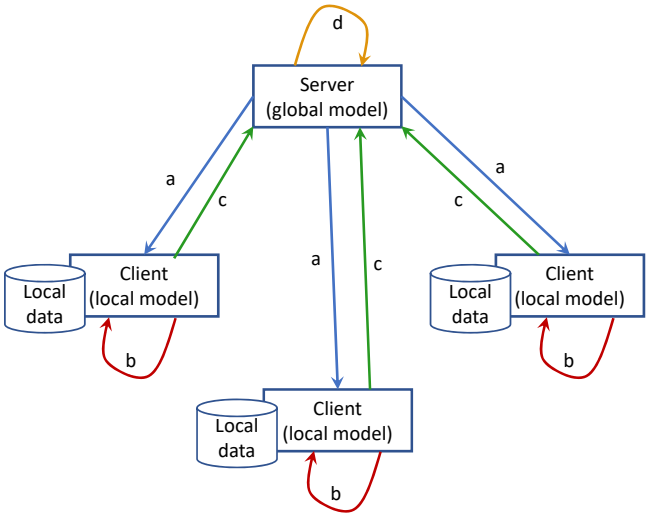
***Abstract*—Many modern applications in the area of smart computing are based on machine learning techniques. To train machine learning models, a large amount of data is usually required, which is often not readily available at a central location. Federated learning enables the training of machine learning models from distributed datasets at client devices without transmitting the data to a central place, which has benefits including preserving the privacy of user data and reducing communication bandwidth. In this demonstration, we show a federated learning system deployed in an emulated wide-area communications network with dynamic, heterogeneous, and intermittent resource availability, where the network is emulated using a CORE/EMANE emulator. In our system, the environment is decentralized and each client can ask for assistance by other clients. The availability of clients is intermittent so only those clients that are available can provide assistance. A graphical interface illustrates the network connections and the user can adjust these connections through the interface. A user interface displays the training progress and each client's contribution to training.**

***Index Terms*—Distributed machine learning, federated learning, model training, networking**

I. Introduction

Machine learning is a promising technology for many emerging smart applications nowadays. To apply machine learning, models need to be trained usually with a large amount of data. In a distributed system where data is collected by local clients and stored locally, it is difficult to send all the data to a central location due to bandwidth limitations and data privacy concerns. To address this problem, *federated learning* has been proposed [1], which allows the data to remain at local clients and only the model parameters are shared between clients and a central server. Since the size of model parameters is usually much smaller than the size of the dataset at local clients, federated learning saves the communication bandwidth and also preserves the privacy of each client's raw data. Note that for deep learning models, there exist ways to extract only a small amount of data to represent the entire set of model parameters [2], [3].

The basic system setup for federated learning includes a server and multiple clients, as shown in Fig. 1. In each training task, the server and all clients are configured to train a model with the same architecture. The server initiates the model and sends the parameter vector of the model to all clients. After receiving the model parameter, each client computes updates to the model parameter using (possibly stochastic) gradient descent on its local dataset. The local model update is performed for a specific number of steps, and this local update process at each client is similar to what is done in the centralized model training for deep neural networks [4]. After completing the pre-specified number of steps of local updates, the resulting local model parameter at each client is sent back to the server. The server aggregates all local parameters (usually by computing an average over the local parameter vectors) and sends the aggregated parameter back to all clients. Then, the same process continues until some



Steps:

a – Broadcast model parameter vector from server to all clients

b – Local model update by each client using its local dataset (τ steps of gradient descent)

c – Send updated local parameter vector from each client back to the server

d – Server computes the global model parameter by aggregating received local parameters

(The above steps continue until training completes)

Fig. 1. Federated learning system.

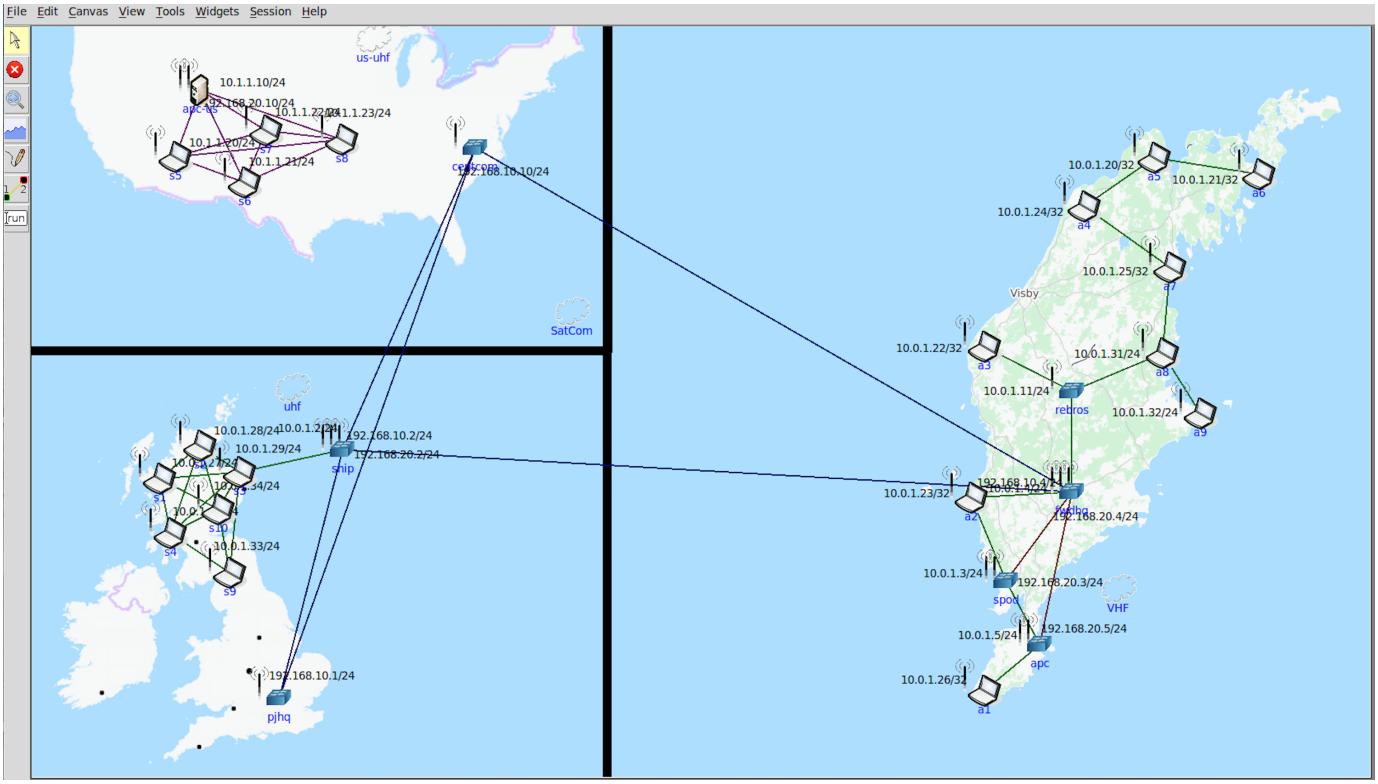


Fig. 2. Emulated network topology.

stopping condition has reached.

Several improvements to federated learning have been made after the initial proposal in [1]. For example, the adaptation of the number of local update steps (τ in Fig. 1) for efficient resource utilization is studied in [5], how to improve the security of global parameter aggregation is studied in [6], an approach of selecting the set of participating clients is proposed in [7], sharing a small subset of data to improve the performance in non-i.i.d. data distribution cases is considered in [8]. Recently, a design of large-scale federated learning systems is presented in [9], where it is mentioned that the connection between client and server may be intermittent due to wide-area network connection, and the availability of each client itself may be intermittent too due to the dynamics in the resource availability of each client.

In this work, we will show a demonstration of federated learning in an emulated networked system with dynamic, heterogeneous, and intermittent resource availability, where the control mechanism in [5] is applied. This is an extension to our previous demonstration [10] which illustrated federated learning in stable network environments. The system we show can be used as a research tool for developing efficient federated learning mechanisms that are robust against system variations.

II. Description of the Demonstration

We consider an emulated wide-area networked system with a variety of network types, as shown in Fig. 2. The system has both fixed and mobile nodes and the emulation is done using CORE/EMANE [11]. The graphical user interface shown in Fig. 2 is provided by the emulator, which allows the user to configure the network links as well as the locations of nodes. In our demonstrated system, each node runs a software stack that includes a Delay Tolerant Networking (DTN) layer, a local controller to respond to requests, and a federated learning stack that uses TensorFlow [12] with our own distributed protocol (see [10]).

The system allows any node to initiate a federated learning task that involves other participating nodes. The node that initiates the task serves as the server in federated learning. If the initiating node has data that is useful for its own training task, it also serves as a client at the same time. Note that a node can be a server and a client simultaneously in federated learning [10].

During task initiation, a set of specifications for the machine learning task (such as model architecture, data type, and expected prediction labels) is broadcast to all other nodes in the network. The DTN layer ensures that each node will receive the specifications, although they may not receive them instantaneously if the network connection is intermittent. After receiving the specifications, each node checks whether it has the data that can be useful for the machine learning task and whether it has enough computational resource to participate in the task. If both are true, the node will send a confirmation to the node that initiated the task (i.e., the federated learning server).

Federated learning starts when a waiting time has elapsed

after task initiation. The waiting time is determined by the task initiator, and can be related to aspects such as the minimum number of participating clients. During federated learning, clients can join or leave the network at any time. The system implements a mechanism that ignores updates from clients that are significantly outdated (i.e., the updated model parameter is computed based on an old global model parameter that is very different from the current global model parameter).

The user interface that displays the training progress is shown in Fig. 3. It allows the user to understand the effect of mobility and bandwidth in a visual way, and to rapidly model the effects of different network parameters on the overall machine learning model creation process.

After the model is trained, it can either be used locally (at the node that initiated the training task) or re-distributed to any nodes that require the model. In the demonstration, a separate application will query the trained model for image classification, and will show that the images can now be classified with a fairly high accuracy.

The system we present in this demonstration can be used as the basis for ongoing experimentation, with the aim of deploying this approach on next generation networking equipment and hardware.

Acknowledgment

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

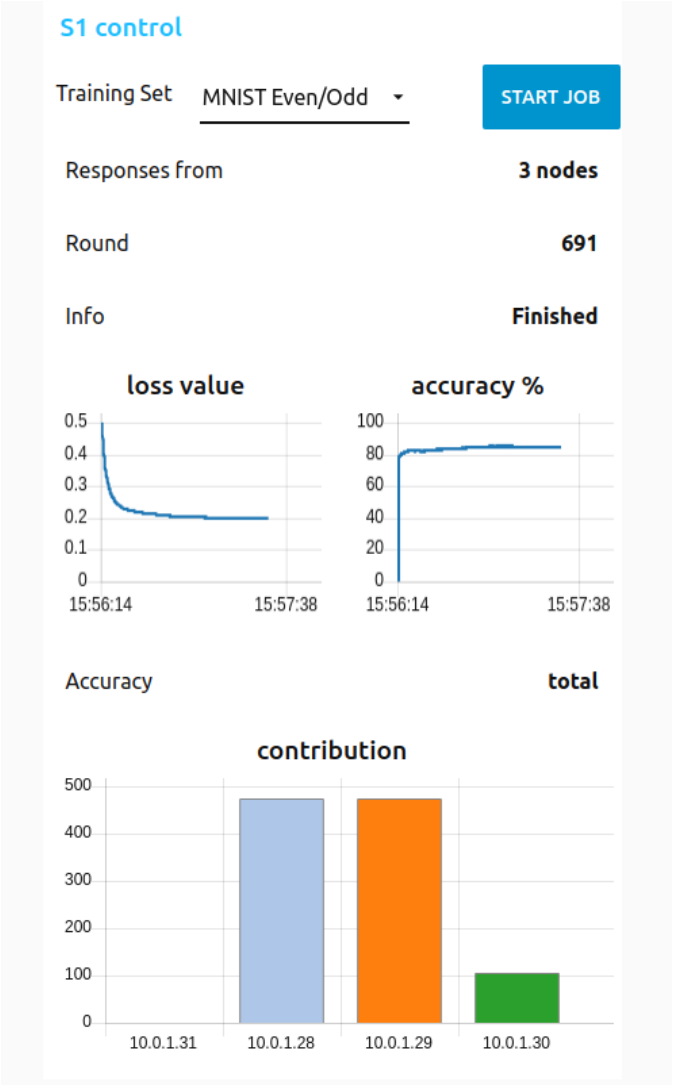


Fig. 3. User interface.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[2] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[3] C. Hardy, E. Le Merrer, and B. Sericola, "Distributed deep learning on edge-devices: feasibility via adaptive compression," in *Network Computing and Applications (NCA), 2017 IEEE 16th International Symposium on*. IEEE, 2017, pp. 1–8.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[5] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, pp. 1–1, 2019.

[6] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[7] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," *arXiv preprint arXiv:1804.08333*, 2018.

[8] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[9] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," *CoRR*, vol. abs/1902.01046, 2019. [Online]. Available: http://arxiv.org/abs/1902.01046

[10] T. Tuor, S. Wang, T. Salonidis, B. J. Ko, and K. K. Leung, "Demo abstract: Distributed machine learning at resource-limited edge nodes," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2018, pp. 1–2.

[11] J. Ahrenholz, "Comparison of core network emulation platforms," in *2010-Milcom 2010 Military Communications Conference*. IEEE, 2010, pp. 166–171.

[12] "Tensorflow." [Online]. Available: https://www.tensorflow.org/