Poster: Combining Split and Federated Architectures for Efficiency and Privacy in Deep Learning

Valeria Turina
Saint Louis University
valeria.turina@slu.edu

Zongshun Zhang
Boston University
zhangzs@bu.edu

Flavio Esposito
Saint Louis University
flavio.esposito@slu.edu

Ibrahim Matta
Boston University
matta@bu.edu

ABSTRACT

Distributed learning systems are increasingly being adopted for a variety of applications as centralized training becomes unfeasible. A few architectures have emerged to divide and conquer the computational load, or to run privacy-aware deep learning models, using split or federated learning. Each architecture has benefits and drawbacks. In this work, we compare the efficiency and privacy performance of two distributed learning architectures that combine the principles of split and federated learning, trying to get the best of both. In particular, our design goal is to reduce the computational power required by each client in *Federated Learning* and to parallelize *Split Learning*. We share some initial lessons learned from our implementation that leverages the *PySyft* and *PyGrid* libraries.

CCS CONCEPTS

• **Security and privacy** → **Privacy protections**; • **Computer systems organization** → **Distributed architectures**; • **Computing methodologies** → *Machine learning*.

KEYWORDS

Split Learning, Federated Learning, Privacy.

ACM Reference Format:
Valeria Turina, Zongshun Zhang, Flavio Esposito, and Ibrahim Matta. 2020. Poster: Combining Split and Federated Architectures for Efficiency and Privacy in Deep Learning. In *The 16th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '20), December 1–4, 2020, Barcelona, Spain.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3386367.3431678

1 INTRODUCTION

Federated learning was introduced by McMahan et al. [2] in 2016. It allows the training of a neural network without sharing the raw training data. Thus, the data can be privately owned by each process participating in the training. The design principle of this architecture assumes that the machine learning model is sent to the training processes with a data shard each, and then the different (decomposed) neural networks are combined via an exchange of the neural network weights during the training phase. While federated learning is becoming a necessity for many applications that benefit from regular retraining [5], in other applications this approach faces a challenge: clients required to run the sub-training process may not have enough computational power to train the full neural network model, and the delay of the aggregation phase is dictated by the slowest client. To solve this and other problems, another type of distributed architecture has recently emerged: *Split Learning* [8]. In Split Learning, multiple agents systematically split the deep neural network (composed of many computational layers) and run a subset of these layers, sharing over the network only the intermediate result(s). The seminal paper *SplitNN* [8] generally lowered the computing power required at each device as compared to the federated learning model. But Split Learning processes need to run sequentially by design and are hard to parallelize. Furthermore, the training process may not reach convergence sharply, or may not reach convergence at all if the dataset is unbalanced. In this work, we study how to improve the efficiency and privacy of Split and Federated learning, by combining them into a new architecture, that we call *Federated Split Learning*. We then compare this architecture with *Parallel Split Learning* [9], a recent attempt to combining the two classical architectures. We tested our algorithms on real datasets and observe that *Federated Split Learning* is similar in terms of accuracy to *Parallel Split Learning* but the former one is better in terms of privacy provided the datasets at the clients are big enough. Furthermore, we are working on a real distributed implementation of the privacy-aware versions. Also, to extend the comparison, we plan to measure other aspects of the performance, e.g. in terms of cost, and to conduct further experiments with other neural networks and datasets.

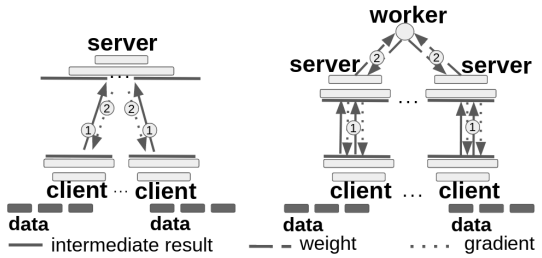

Figure 1: Parallel (left) and Federated (right) Split Learning

2 FEDERATED SPLIT LEARNING

Our *Federated Split Learning* architecture is composed of a number of clients that is equal to the number of servers, as illustrated in Figure 1. The neural network is divided into two parts as in the *SplitNN* algorithm and then each client-server pair computes the training of the model simultaneously. When all data in each client has been used to update the parameters, the neural network weights of the servers' portion of the neural network are averaged and updated.

CoNEXT '20, December 1–4, 2020, Barcelona, Spain

ACM ISBN 978-1-4503-7948-9/20/12…$15.00
https://doi.org/10.1145/3386367.3431678

Note that at the end of the training each of the clients will have a different set of parameters based on the input data. We use *Parallel Split Learning* [9] as the state of art to compare with our new architecture. It splits the neural network in to two parts: the first is sent to I different clients and the second to a server. The forward propagation phase of a neural network learning process begins inside each client simultaneously, and then the intermediate activation output is sent to the single central server that continues the forward pass and computes the loss function using a batch size equal to $I \times (client's\ batch\ size)$. After this step, from the server-side, the gradient of the loss is computed and starts to backpropagate. Finally, each client receives a gradient and finishes the backpropagation.

2.1 Performance and Privacy analysis

Performance analysis. In Figure 2, considering *privacy-oblivious* graphs, we can observe that the performance in terms of accuracy of *Federated Split Learning* is similar to *Parallel Split Learning* if each client has enough data; but if the data owned by each client is small, the latter can achieve a higher level of accuracy in less time. For example, with a fixed training dataset of 60,000 of images when we move from 20 to 500 clients to compute the training, we can see a significant drop in the test-set level of accuracy, from 98% to 40%, under *Federated Split Learning*.

Privacy analysis. We implement an attacker neural network, that can reconstruct the original data starting from the intermediate result that a client sends to the server during the training phase. We notice that for both *Federated Split Learning* and *Parallel Split learning*, the attacker is able to easily reconstruct the original data. To overcome this problem, we changed the loss function used during the training of the two architectures; in particular, as in [6], we add a privacy increment (called *Distance Correlation* [6]) to the loss function *Cross Entropy* that describes how the initial data derives from the intermediate result. Using the attacker neural network on these new intermediate results we note that *Federated Split Learning* is able to provide better privacy for the data than using *Parallel Split Learning*; at the same time, *Federated Split Learning* is able to obtain good results in terms of accuracy (compare the privacy-aware curves in Figure 2). We noted that a drop of 10% of the distance correlation value in *Federated Split Learning* is enough to preserve the privacy of the input data. For example, in our experiments using the MNIST dataset [11] the distance correlation value changes from 0.9958 to 0.9085 when we move from the *privacy-oblivious* to the *privacy-aware* training in *Federated Split learning*; the value of this quantity remains almost identical for *Parallel Split Learning*: from 0.9964 to 0.9944.

3 SYSTEM IMPLEMENTATION

To implement the two architectures, we use the PySyft library [7] for a local setup of the processes on the same machine and, then, the PyGrid [1] library for a distributed setup of the architectures on different machines. In particular, the distributed implementation is deployed on several virtual machines hosted on the Chameleon Cloud [3]. Each instance reserved has 48 vCPUs and 128 GB memory. We experiment with multiple datasets, including MNIST and a dataset of chest X-ray for COVID-19 recognition [4], using different neural networks, respectively, LeNet [10] and DarkCovidNet [4].

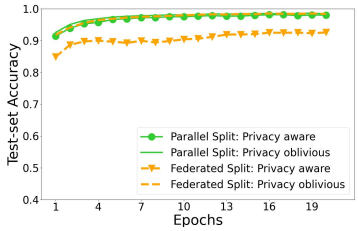

Figure 2: Accuracy: with and without privacy-awareness

4 INITIAL RESULTS AND LESSONS LEARNED

Using the distributed setup on different virtual machines, we are able to reproduce the same values of accuracy and loss function obtained in the local setup of the *privacy-oblivious* approach. However, running the code on *PyGrid* workers, we obtain an average run-time for each epoch of 6 minutes in *Parallel Split Learning* and 8 minutes in *Federated Split Learning*. This result differs from the experiment run over the same local simulated versions, which have corresponding average run-time of 2.1 and 2.8 minutes, respectively. This run-time discrepancy suggests a need for performant split and federated learning libraries as an interested networking (for machine learning) research direction.

We are studying the backward propagation process, where we add the privacy increment to the standard *Cross Entropy*. To keep the privacy increment private inside each client, we are implementing the *Distance Correlation* increment so that it does not depend on the server neural network,during the backpropagation phase when the server's gradients are sent to the client neural networks.

In conclusion, we found that *Federated Split Learning* performs well in terms of time of convergence as it is able to reach performance (accuracy) similar to the state of art, but it is better in terms of privacy, provided that each of the clients has a reasonable amount of data for its training operations.

5 ACKNOWLEDGMENT

This work has been partially supported by Comcast and by NSF awards CNS-1647084, CNS-1836906, CNS-1908574, CNS-1908677.

REFERENCES

[1] 2020. *PyGrid*. Retrieved Oct 8, 2020 from https://github.com/OpenMined/PyGrid
[2] H. McMahan et. al. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.
[3] Kate Keahey et. al. 2020. Lessons Learned from the Chameleon Testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*.
[4] Ozturk et. al. 2020. Automated Detection of COVID-19 Cases Using Deep Neural Networks with X-ray Images. *Computers in Biology and Medicine* 121 (04 2020). https://doi.org/10.1016/j.compbiomed.2020.103792
[5] Philipp Moritz et. al. 2018. Ray: A Distributed Framework for Emerging AI Applications. In *13th USENIX OSDI 18*. Carlsbad, CA, 561–577.
[6] Praneeth Vepakomma et. al. 2020. NoPeek: Information leakage reduction to share activations in distributed deep learning. *CoRR* abs/2008.09161 (2020). arXiv:2008.09161 https://arxiv.org/abs/2008.09161
[7] Theo Ryffel et. al. 2018. A generic framework for privacy preserving deep learning. arXiv:1811.04017 [cs.LG]
[8] Otkrist Gupta and Ramesh Raskar. 2018. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications* 116 (August 2018), 1–8. https://doi.org/10.1016/j.jnca.2018.05.003
[9] Jeon Joohyung and Joongheon Kim. 2020. Privacy-Sensitive Parallel Split Learning. 7–9. https://doi.org/10.1109/ICOIN48656.2020.9016486
[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[11] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. (2010). http://yann.lecun.com/exdb/mnist/