SPLIT LEARNING FOR HEALTH: DISTRIBUTED DEEP LEARNING WITHOUT SHARING RAW PATIENT DATA

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, Ramesh Raskar
Massachusetts Institute of Technology
Cambridge, MA 15213, USA
{vepakom,otkrist,tswedish,raskar}@mit.edu

ABSTRACT

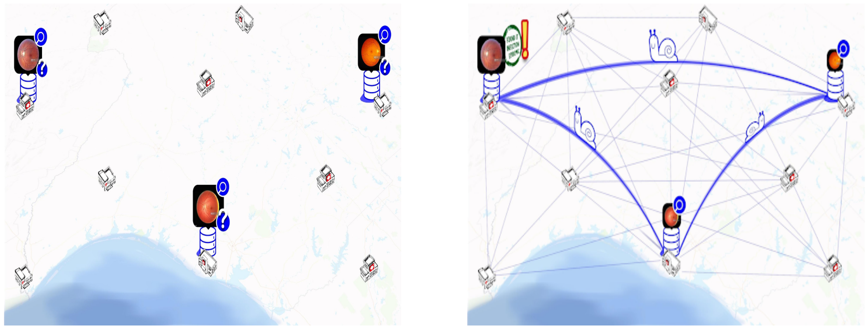
Can health entities collaboratively train deep learning models without sharing sensitive raw data? This paper proposes several configurations of a distributed deep learning method called SplitNN to facilitate such collaborations. SplitNN does not share raw data or model details with collaborating institutions. The proposed configurations of splitNN cater to practical settings of i) entities holding different modalities of patient data, ii) centralized and local health entities collaborating on multiple tasks and iii) learning without sharing labels. We compare performance and resource efficiency trade-offs of splitNN and other distributed deep learning methods like federated learning, large batch synchronous stochastic gradient descent and show highly encouraging results for splitNN.

1 INTRODUCTION

Collaboration in health is heavily impeded by lack of trust, data sharing regulations such as HIPAA (Annas et al., 2003; CDC, 2003; Mercuri, 2004; Gostin et al., 2009; Luxton et al., 2012) and limited consent of patients. In settings where different institutions hold different modalities of patient data in the form of electronic health records (EHR), picture archiving and communication systems (PACS) for radiology and other imaging data, pathology test results, or other sensitive data such as genetic markers for disease, collaborative training of distributed machine learning models without any data sharing is desired. Deep learning methods in general have found a pervasive suite of applications in biology, clinical medicine, genomics and public health as surveyed in Ching et al. (2018); Shickel et al. (2018); Miotto et al. (2017); Ravı et al. (2017); Alipanahi et al. (2015); Litjens et al. (2017). Training of distributed deep learning models without sharing model architectures and parameters in addition to not sharing raw data is needed to prevent undesirable scrutiny by other entities. As a concrete health example, consider the use case of training a deep learning model for patient diagnosis via collaboration of two entities holding pathology test results and radiology data respectively. These entities are unable to share their raw data with each other due to the concerns noted above. That said, diagnostic performance of the distributed deep learning model is highly contingent on being able to use data from both the institutions for its training. In addition to such multi-modal settings, this problem also manifests in settings with entities holding data of the same modality as shown in Fig 1 below. As illustrated, local hospitals or tele-health screening centers do not acquire an enormous number of diagnostic images on their own. These entitites may also be limited by diagnostic manpower. A distributed machine learning method for diagnosis in this setting would enable each individual center to contribute data to an aggregate model without sharing any raw data. This configuration can achieve high accuracy while using significantly lower computational resources and communication bandwidth than previously proposed approaches. This enables smaller hospitals to effectively serve those in need while also benefiting the distributed training network as a whole. In this paper, we build upon splitNN introduced in Gupta & Raskar (2018) to propose specific configurations that cater to practical health settings such as these and furthermore as described in the sections below.

1.1 RELATED WORK:

In addition to splitNN Gupta & Raskar (2018), techniques of federated deep learning McMahan et al. (2016) and large batch synchronous stochastic gradient descent (SGD)Chen et al. (2016); Konečny et al. (2015) are currently available approaches for distributed deep learning. There has been no



(a) Non-cooperating health units

(b) Distributed learning without raw data sharing

Figure 1: Distributed learning over retinopathy images (or undetected fast moving threats) over slow bit-rate ('snail-pace'), to detect the emerging threat by pooling their images but without exchanging raw patient data.

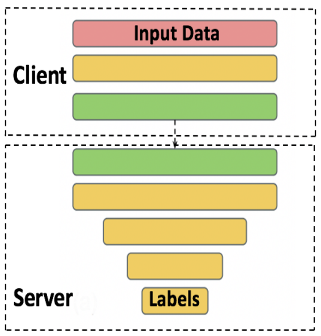
work as yet on federated deep learning and large batch synchronous SGD methods with regards to their applicability to useful non-vanilla settings of distributed deep learning studied in rest of this paper such as a) distributed deep learning with vertically partitioned data, b) distributed deep learning without label sharing, c) distributed semi-supervised learning and d) distributed multi-task learning. That said, with regards to 'non-neural network' based federated learning techniques, the work in Hardy et al. (2017) shows their applicability to vertically partitioned distributed data Navathe et al. (1984); Agrawal et al. (2004); Smith et al. (2017); Abadi et al. (2007) shows applicability to multi-task learning in distributed settings. We now propose configurations of splitNN for all these useful settings in the rest of this paper.

2 SplitNN Algorithm:

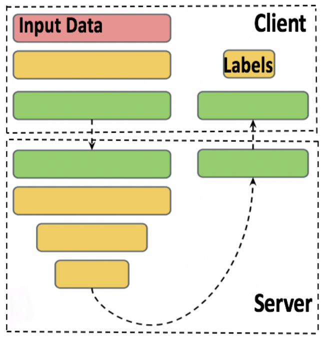
Algorithm 1 `SplitNN`. The K clients are indexed by k; B is the local minibatch size, and η is the learning rate.

```
EnsureServer executes at round $t \geq 0$:
  for each client $k \in S_t$ in parallel do
    $\mathbf{A}_t^k \leftarrow \text{ClientUpdate}(k, t)$
    Compute $\mathbf{W}_t \leftarrow \mathbf{W}_t - \eta \nabla \ell(\mathbf{W}_t; \mathbf{A}_t)$
    Send $\nabla \ell(\mathbf{A}_t; \mathbf{W}_t)$ to client $k$ for ClientBackprop$(k, t)$

EnsureClientUpdate$(k, t)$:   // Run on client $k$
  $\mathbf{A}_t^k = \phi$
  for each local epoch $i$ from $1$ to $E$ do
    for batch $b \in \mathcal{B}$ do
      Concatenate $f(b, \mathbf{H}_t^k)$ to $\mathbf{A}_t^k$
  return $\mathbf{A}_t^k$ to server

EnsureClientBackprop$(k, t, \nabla \ell(\mathbf{A}_t; \mathbf{W}_t))$:   // Run on client $k$
  for batch $b \in \mathcal{B}$ do
    $\mathbf{H}_t^k = \mathbf{H}_t^k - \eta \nabla \ell(\mathbf{A}_t; \mathbf{W}_t; b)$
```

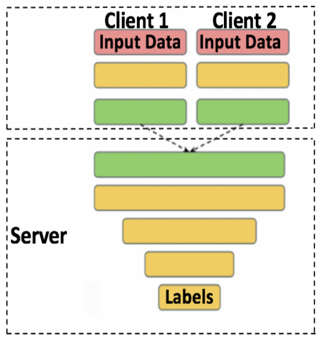
In this method each client trains the network upto a certain layer known as the cut layer and sends the weights to server. The server then trains the network for rest of the layers. This completes the forward propagation. Server then generates the gradients for the final layer and back-propagates the error until the cut layer. The gradient is then passed over to the client. The rest of the back-propagation is completed by client. This is continued till the network is trained. The shape of the cut could be arbitrary and not necessarily, vertical. In this framework as well there is no explicit sharing of raw data.



(a) Simple vanilla split learning



(b) Split learning without label sharing



(c) Split learning for vertically partitioned data

Figure 2: Split learning configurations for health shows raw data is not transferred between the client and server health entities for training and inference of distributed deep learning models with SplitNN.

2.1 Configurations for health

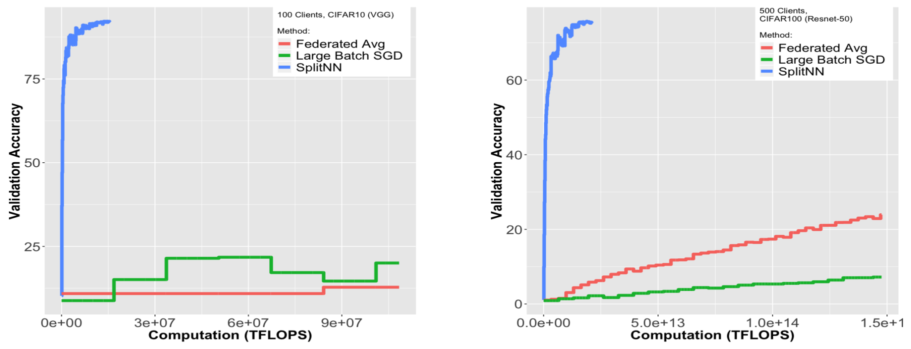
In this section we propose several configurations of splitNN for various practical health settings: **Simple vanilla configuration for split learning:** This is the simplest of splitNN configurations as shown in Fig 2a. In this setting each client, (for example, radiology center) trains a partial deep network up to a specific layer known as the "cut layer." The outputs at the cut layer are sent to a server which completes the rest of the training without looking at raw data (radiology images) from clients. This completes a round of forward propagation without sharing raw data. The gradients are now back propagated at the server from its last layer until the cut layer. The gradients at the cut layer (and only these gradients) are sent back to radiology client centers. The rest of back propagation is now completed at the radiology client centers. This process is continued until the distributed split learning network is trained without looking at each others raw data.

U-shaped configurations for split learning without label sharing: While the method described above requires sharing of labels, we can mitigate this issue by using a U-shaped configuration that does not require any label sharing. In this setup we wrap the network around at end layers of server's network and send the outputs back to client entities (as seen in Fig.2b). While the server still retains a majority of its layers, the clients generate the gradients from the end layers and use them for backpropagation without sharing the corresponding labels.

Vertically partitioned data for split learning: This configuration allows for multiple institutions holding different modalities of patient data to learn distributed models without data sharing. In Fig. 2c, we show example configurations of splitNN suitable for such multi-modal multi-institutional collaboration. As a concrete example we walkthrough the case where radiology centers collaborate with pathology test centers and a server for disease diagnosis. As shown in Fig. 2c radiology centers holding imaging data modalities train a partial model upto the cut layer. In the same way the pathology test center having patient test results trains a partial model upto its own cut layer. The outputs at the cut layer from both these centers are then concatenated and sent to the disease diagnosis server that trains the rest of the model. Iterative forward and backward passes are continued until convergence.

3 Results about resource efficiency

We share a comparison from Gupta & Raskar (2018) of validation accuracy and required client computational resources in Figure 3 for the three techniques of federated learning, large batch synchronous SGD and splitNN as they are tailored for distributed deep learning. As seen in this figure, the comparisons were done on the CIFAR 10 and CIFAR 100 datasets using VGG and Resnet-50 architectures for 100 and 500 client based setups respectively. We empirically demonstrate that SplitNN outperforms the techniques of federated learning and large batch synchronous SGD in terms



(a) Accuracy vs client-side flops on 100 clients with VGG on CIFAR 10

(b) Accuracy vs client-side flops on 500 clients with Resnet-50 on CIFAR 100

Figure 3: We show dramatic reduction in computational burden (in tflops) while maintaining higher accuracies when training over large number of clients with splitNN. Blue line denotes distributed deep learning using splitNN, red line indicate federated averaging and green line indicates large batch SGD.

of higher accuracies with drastically lower computational requirements on the side of clients. In tables 1 and 2 we share more comparisons from Gupta & Raskar (2018) on computing resources in TFlops and communication bandwidth in GB required by these techniques. SplitNN again has a drastic improvement of computational resource efficiency on the client side. In the case with a relatively smaller number of clients the communication bandwidth required by federated learning is less than splitNN. These improvements on the client side resource efficiency are even more dramatic due to the presence of a smaller number of parameters in earlier layers of convolutional neural networks (CNN's) like VGG and Resnet in addition to the fact that computation is split due to the cut layers. This uneven distribution of network parameters holds for the vast majority of modern CNN's, a property that SplitNN can effectively exploit.

Method	**100 Clients**	**500 Clients**
Large Batch SGD	29.4 TFlops	5.89 TFlops
Federated Learning	29.4 TFlops	5.89 TFlops
SplitNN	0.1548 TFlops	0.03 TFlops

Table 1: Computation resources consumed per client when training CIFAR 10 over VGG (in teraflops) are drastically lower for SplitNN than Large Batch SGD and Federated Learning.

Method	**100 Clients**	**500 Clients**
Large Batch SGD	13 GB	14 GB
Federated Learning	3 GB	2.4 GB
SplitNN	6 GB	1.2 GB

Table 2: Computation bandwidth required per client when training CIFAR 100 over ResNet (in gigabytes) is lower for splitNN than large batch SGD and federated learning with a large number of clients. For setups with a smaller number of clients, federated learning requires a lower bandwidth than splitNN. Large batch SGD methods popular in data centers use a heavy bandwidth in both settings.

4 Conclusion and future work

Simple configurations of distributed deep learning do not suffice for various practical setups of collaboration across health entities. We propose novel configurations of a recently proposed distributed deep learning technique called splitNN that is dramatically resource efficient in comparison to currently available distributed deep learning methods of federated learning and large batch synchronous

SGD. SplitNN is versatile in allowing for many plug and play configurations based on the required application. SplitNN is also scalable to large-scale settings and can use any state of the art deep learning architectures. In addition, the boundaries of resource efficiency can be pushed further in distributed deep learning by combining splitNN with neural network compression methods Lin et al. (2017); Louizos et al. (2017); Han et al. (2015) for seamless distributed learning with edge devices. Looking at combinations of split learning and differential privacy, secure multi-party computation is an interesting direction for future work as well given that there has been active research in recent times in all these areas.

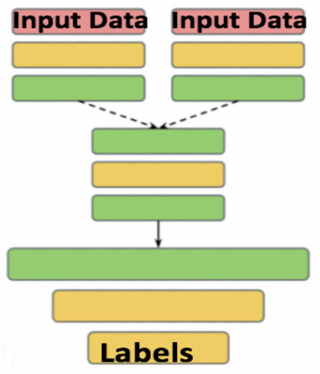
5 Supplementary

5.1 Secondary configurations

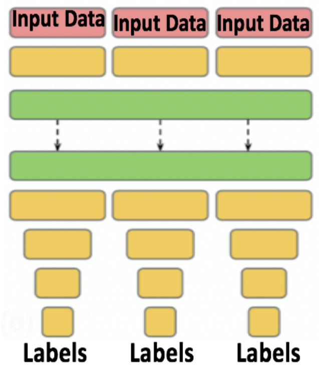
In this subsection we propose some more split learning configurations of splitNN for versatile collaborations in health to train and infer from distributed deep learning models without sharing raw patient data.
Extended vanilla split learning: As shown in Fig. 4a we give another modification of vanilla split learning where the result of concatenated outputs is further processed at another client before passing it to the server.

Configurations for multi-task split learning: As shown in Fig. 4b, in this configuration multi-modal data from different clients is used to train partial networks up to their corresponding cut layers. The outputs from each of these cut layers are concatenated and then sent over to multiple servers. These are used by each server to train multiple models that solve different supervised learning tasks.

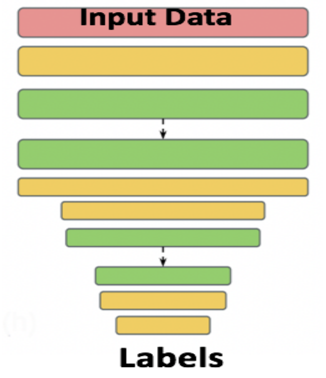
Tor Syverson et al. (2004) **like configuration for multi-hop split learning:** This configuration is an analogous extension of the vanilla configuration. In this setting multiple clients train partial networks in sequence where each client trains up to a cut layer and transfers its outputs to the next client. This process is continued as shown in Fig. 4c as the final client sends its activations from its cut layer to a server to complete the training.



(a) Extended vanilla split learning



(b) Split learning for multi-task output with vertically partitioned input



(c) 'Tor' Syverson et al. (2004) like multi-hop split learning

Figure 4: Split learning configurations for health shows raw data is not transferred between the client and server health entities for training and inference of distributed deep learning models with SplitNN.

References

Daniel J Abadi, Adam Marcus, Samuel R Madden, and Kate Hollenbach. Scalable semantic web data management using vertical partitioning. In *Proceedings of the 33rd international conference* *on Very large data bases*, pp. 411–422. VLDB Endowment, 2007.

Sanjay Agrawal, Vivek Narasayya, and Beverly Yang. Integrating vertical and horizontal partitioning into automated physical database design. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pp. 359–370. ACM, 2004.

Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of DNA and RNA binding proteins by deep learning. *Nature biotechnology*, 33(8): 831, 2015.

George J Annas et al. HIPAA regulations-a new era of medical-record privacy? *New England Journal of Medicine*, 348(15):1486–1490, 2003.

CDC. HIPAA privacy rule and public health. guidance from CDC and the US Department of Health and Human Services. *MMWR: Morbidity and mortality weekly report*, 52(Suppl. 1):1–17, 2003.

Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, 2018.

Lawrence O Gostin, Laura A Levit, Sharyl J Nass, et al. *Beyond the HIPAA privacy rule: Enhancing privacy, improving health through research.* National Academies Press, 2009.

Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

Jakub Konečny, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

Jakub Konečny, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen AWM van der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pp. 3288–3298, 2017.

David D Luxton, Robert A Kayl, and Matthew C Mishkind. mHealth data security: The need for HIPAA-compliant standardization. *Telemedicine and e-Health*, 18(4):284–288, 2012.

H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

Rebecca T Mercuri. The HIPAA-potamus in health care data security. *Communications of the ACM*, 47(7):25–28, 2004.

Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 2017.

Shamkant Navathe, Stefano Ceri, Gio Wiederhold, and Jinglie Dou. Vertical partitioning algorithms for database design. *ACM Transactions on Database Systems (TODS)*, 9(4):680–710, 1984.

Daniele Ravı, Charence Wong, Fani Deligianni, Melissa Berthelot, Javier Andreu-Perez, Benny Lo, and Guang-Zhong Yang. Deep learning for health informatics. *IEEE journal of biomedical and health informatics*, 21(1):4–21, 2017.

Benjamin Shickel, Patrick James Tighe, Azra Bihorac, and Parisa Rashidi. Deep EHR: A survey of recent advances in deep learning techniques for electronic health record (EHR) analysis. *IEEE journal of biomedical and health informatics*, 22(5):1589–1604, 2018.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pp. 4424–4434, 2017.

Paul Syverson, R Dingledine, and N Mathewson. Tor: The second generation onion router. In *Usenix Security*, 2004.