SplitFed

Federated Learning meets Split Learning

0424052076 - Abdus Samee
0424052053 - Mobaswirul Islam

Research Paper

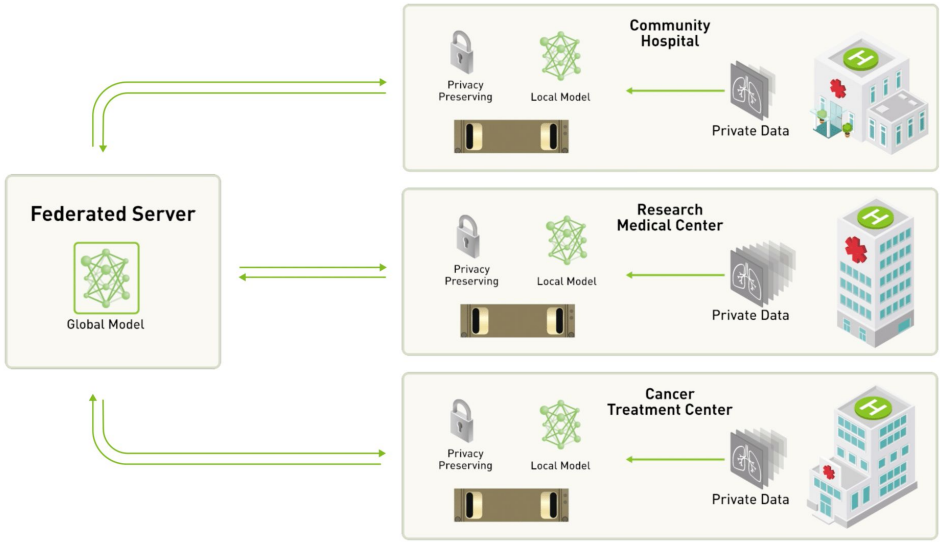
SplitFed: When Federated Learning Meets Split Learning

Chandra Thapa, Pathum Chamikara, Mahawaga Arachchige, Seyit Camtepe, Lichao Sun

Conference: AAAI - `22

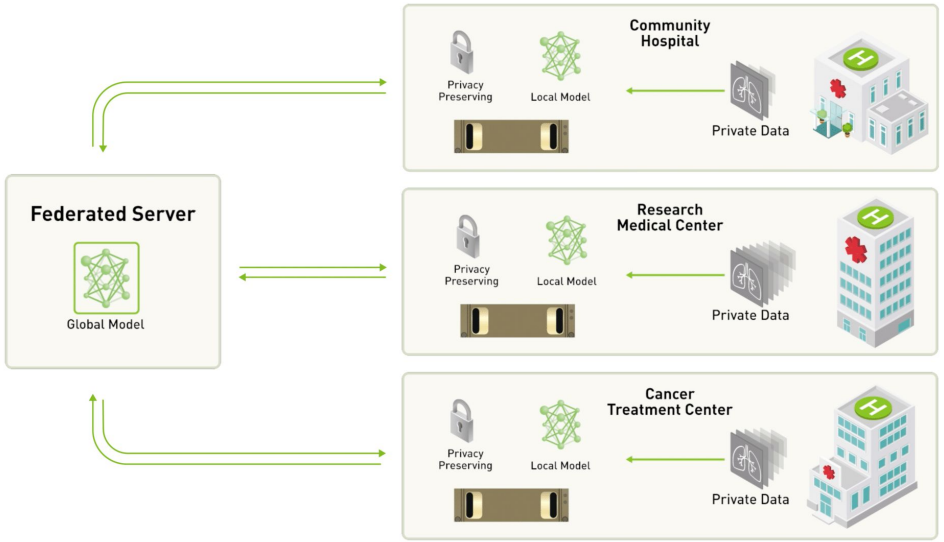
Federated Learning

- Diverse Data Sources
- Data Security or Privacy
- Data Transfer costs



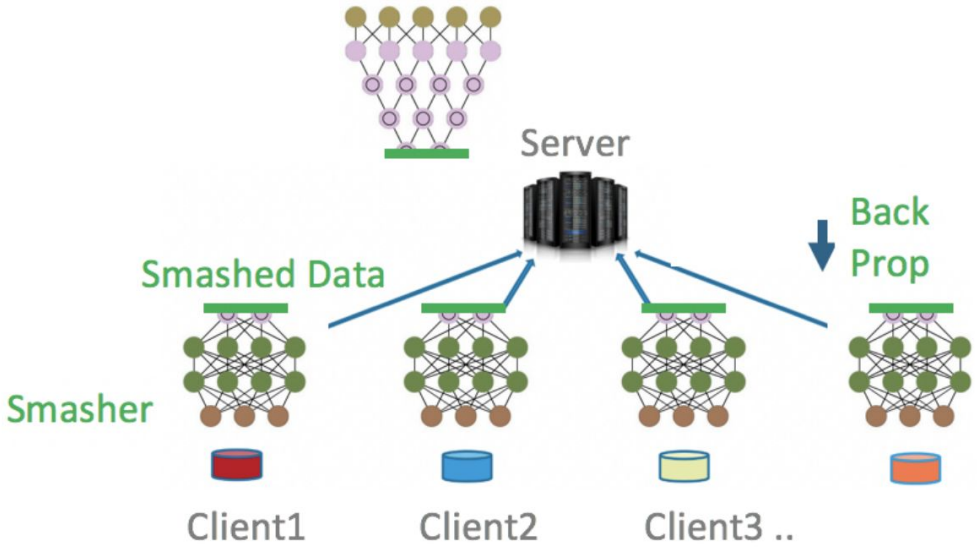
Federated Learning

- Less Robust
- Global & Local Model



Split Learning

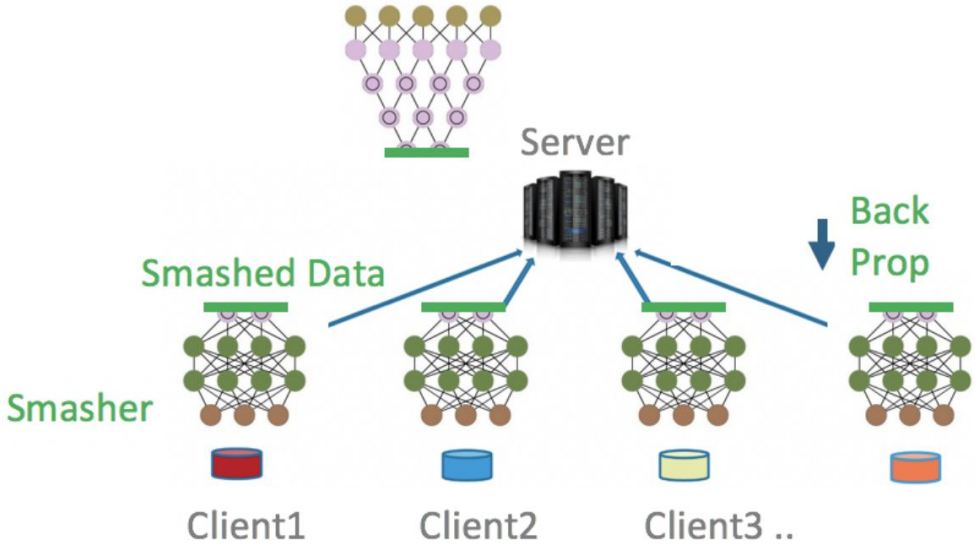
- More Robust Model
- Enhanced Privacy
- Reduced Communication Overhead



Split Learning

- Slower

  (Relay Based Training)



Federated Learning vs Split Learning

- SL provides better model privacy than FL
  - In FL, not all devices are capable of training full models
  - Also, server and clients have full access to the global and local models
  - In SL, the full ML model is splitted into multiple smaller network portions
  - Clients hold the initial layers of the model and the server holds the final layers
  - This reduces processing load too and works better in resource-constrained environments

- SL performs slower than FL due to relay-based training across multiple clients
  - In SL, relay-based training makes client resources idle
  - Only one client engages with the server at an instance
  - No training time overhead in FL

SplitFed Learning

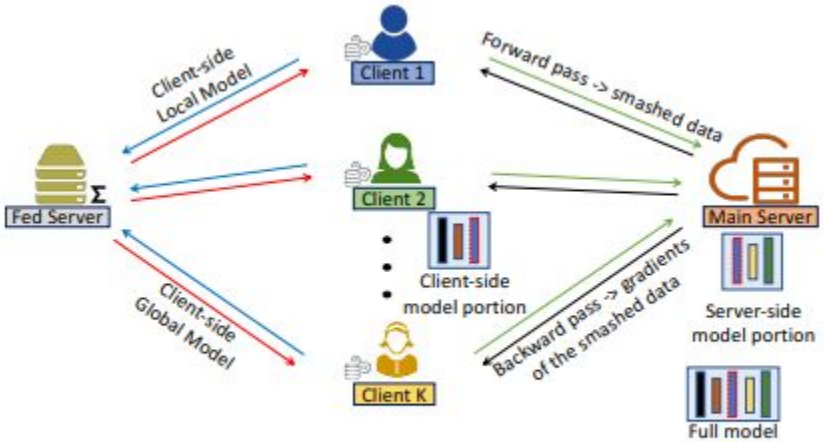


Figure 1: Overview of splitfed learning (SFL) system model.

SplitFed Learning

- Better Privacy (Incorporating differential privacy, PixelDP)
- Efficient
- Resource-Constrained devices can be used
  - Monitoring medical IoT devices
  - Financial fraud detection

Comparison

	SL	FL	SFL
Model aggregation	No	Yes	Yes
Model privacy advantage by splitted model	Yes	No	Yes
Client-side training	Sequential	Parallel	Parallel
Distributed computing	Yes	Yes	Yes
Access to raw data	No	No	No

Table 1: An abstract comparison of split learning (SL), federated learning (FL), and splitfed learning (SFL).

SFL - Overall Structure

- FL strength: parallel processing among distributed clients
- SL strength: splitted network
- Client: low-resource machine
- Main server: high-resource cloud or machine
- Fed server: conduct FedAvg on client-side updates & synchronize client-side global model; not costly

Algorithm

Notations: (1) S_t is a set of K clients at t time instance, (2) $\mathbf{A}_{k,t}$ is the smashed data of client k at t, (3) $\mathbf{Y}_k$ and $\hat{\mathbf{Y}}_k$ are the true and predicted labels, respectively, of the client k, (4) $\nabla \ell_k$ is the gradient of the loss for the client k, (5) n and n_k are the total sample size and sample size at a client k, respectively.

/* Runs on Fed Server */
EnsureFedServer executes:
- **if** *t=0* **then**
  - Initialize $\mathbf{W}_t^{\mathrm{C}}$ (global client-side model)
  - Send $\mathbf{W}_t^{\mathrm{C}}$ to all K clients for ClientUpdate($\mathbf{W}_{k,t}^{\mathrm{C}}$)
- **else**
  - **for** each client $k \in S_t$ in parallel **do**
    - $\mathbf{W}_{k,t}^{\mathrm{C}} \leftarrow$ ClientBackprop($d\mathbf{A}_{k,t}$)
  - **end**
  - Client-side global model updates:
    $\mathbf{W}_{t+1}^{\mathrm{C}} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{W}_{k,t}^{\mathrm{C}}$
  - Send $\mathbf{W}_{t+1}^{\mathrm{C}}$ to all K clients for ClientUpdate($\mathbf{W}_{k,t}^{\mathrm{C}}$)
- **end**

/* Runs on Main Server */
EnsureMainServer executes:
- **if** time instance *t=0* **then**
  - Initialize $\mathbf{W}_t^{\mathrm{S}}$ (global server-side model)
- **else**
  - **for** each client $k \in S_t$ in parallel **do**
    - **while** *local epoch* $e \neq E$ **do**
      - $(\mathbf{A}_{k,t}, \mathbf{Y}_k) \leftarrow$ ClientUpdate($\mathbf{W}_{k,t}^{\mathrm{C}}$)
      - Forward propagation with $\mathbf{A}_{k,t}$ on $\mathbf{W}_t^{\mathrm{S}}$, compute $\hat{\mathbf{Y}}_k$
      - Loss calculation with $\mathbf{Y}_k$ and $\hat{\mathbf{Y}}_k$
      - Back-propagation calculate $\nabla \ell_k(\mathbf{W}_t^{\mathrm{S}}; \mathbf{A}_t^{\mathrm{S}})$
      - Send $d\mathbf{A}_{k,t} := \nabla \ell_k(\mathbf{A}_t^{\mathrm{S}}; \mathbf{W}_t^{\mathrm{S}})$ (i.e., gradient of the $\mathbf{A}_{k,t}$) to client k for ClientBackprop($d\mathbf{A}_{k,t}$)
    - **end**
  - **end**
  - Server-side model update:
    $\mathbf{W}_{t+1}^{\mathrm{S}} \leftarrow \mathbf{W}_t^{\mathrm{S}} - \eta \frac{n_k}{n} \sum_{i=1}^{K} \nabla \ell_i(\mathbf{W}_t^{\mathrm{S}}; \mathbf{A}_t^{\mathrm{S}})$
- **end**

Privacy Protection

- Inherent: Model-to-Data approach & Split network
- Client-side: Differential privacy & PixelDP noise layer
- Fed server:
  - Apply calibrated noise iteratively until model converges or reaches specific iteration count
  - Final convergence will achieve (ε, □) DP
  - Values of (ε, □) should be as small as possible
- Main server:
  - Potential leakage from smashed data
  - Integrate noise layer in client-side model based on PixelDP

DP and PixelDP

- Each client computes gradients by back propagation
- L_2 norm of each gradient is clipped
- Calibrated noise is added to the average gradient

- Calculate sensitivity of a process
- Laplacian noise is applied to randomize any data
- This private version of smashed data is then forwarded to the main server

Total Cost Analysis

Method	Comms. per client	Total comms.	Total model training time
FL	$2\|\mathbf{W}\|$	$2K\|\mathbf{W}\|$	$\{T + 2\frac{\|\mathbf{W}\|}{R} + T_{\text{fedavg}}\}$
SL	$(\frac{2p}{K})q + 2\beta\|\mathbf{W}\|$	$2pq + 2\beta K\|\mathbf{W}\|$	$T + 2\frac{pq}{R} + 2\beta\frac{\|\mathbf{W}\|}{R}K$
SFLV1	$(\frac{2p}{K})q + 2\beta\|\mathbf{W}\|$	$2pq + 2\beta K\|\mathbf{W}\|$	$T + 2\frac{pq}{KR} + 2\frac{\beta\|\mathbf{W}\|}{R} + T_{\text{fedavg}}$
SFLV2	$(\frac{2p}{K})q + 2\beta\|\mathbf{W}\|$	$2pq + 2\beta K\|\mathbf{W}\|$	$T + 2\frac{pq}{KR} + 2\frac{\beta\|\mathbf{W}\|}{R} + \frac{T_{\text{fedavg}}}{2}$

Table 2: Total cost analysis of the four DCML approaches for one global epoch.

- |W|: Model size
- K: No. of clients
- R: Communication rate
- p: Dataset size
- q: Smashed layer size
- T: Global epoch
- T_{fedavg}: Full model agg. time
- β: Ratio of client-side model

Training time: SFLV2 < SFLV1 < SL

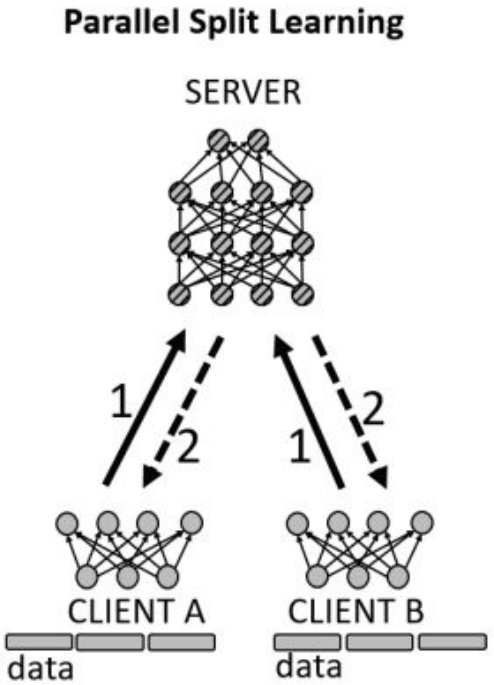
Experimental Results

Dataset	Architecture	Normal	FL	SL	SFLV1	SFLV2
HAM10000	ResNet18	79.3%	77.5%	79.1%	79%	79.2%
HAM10000	AlexNet	80.1%	75 %	73.8%	70.5%	74.9%
FMNIST	LeNet	92.7%	91.9 %	90.4%	89.6%	90.4%
FMNIST	AlexNet	90.5%	89.7%	84.7%	86%	81%
CIFAR10	LeNet	72.1%	69.4 %	62.7%	62.6%	63.8%
MNIST	AlexNet	98.8%	98.7 %	95.1%	96.9%	92%
MNIST	ResNet18	99.3%	99.2 %	99.2%	99%	99.2%

Table 5: Test Results (five clients for DCML)

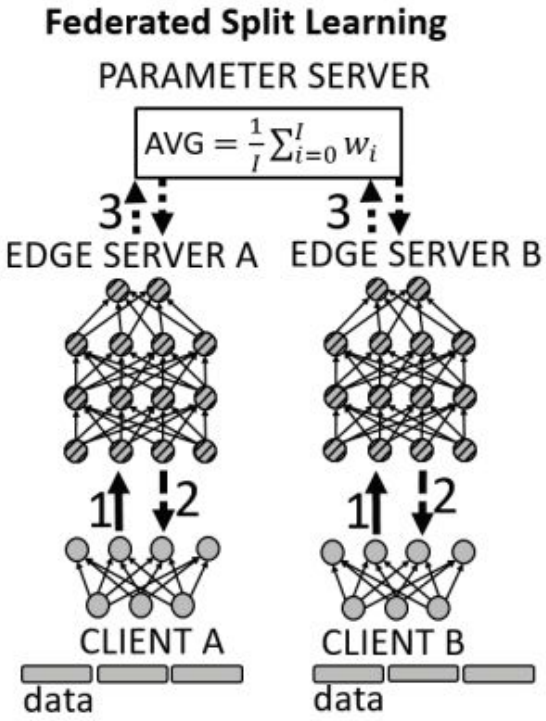
Literature Review

[Privacy-Sensitive Parallel Split Learning: (PSL)]



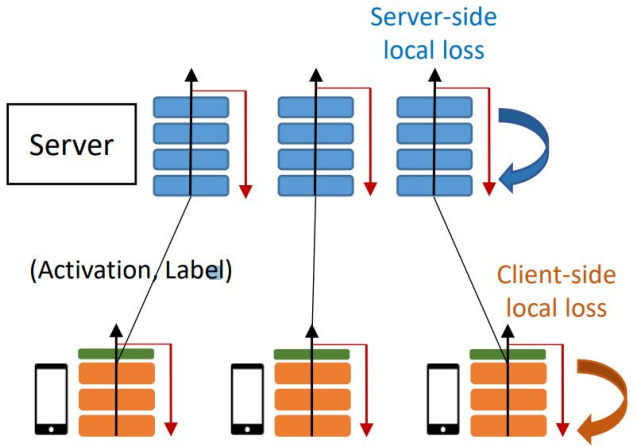
Literature Review

[Federated or Split? A Performance and Privacy Analysis of Hybrid Split and Federated Learning Architecture(FSL)]



Literature Review

[Accelerating Federated Learning with Split Learning on Locally Generated Losses]

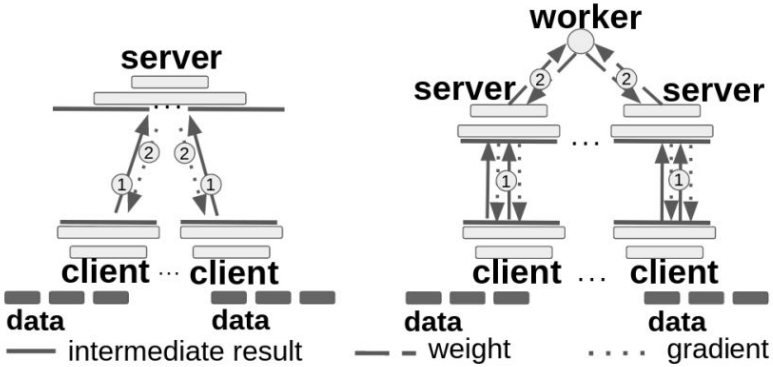


Literature Review

Federated Learning: Opportunities and Challenges

- One Shot FL
- Incentive Mechanism
- FL as a service
- Asynchronous FL
- Blockchain FL

Literature Review



[Combining Split and Federated Architectures for Efficiency and Privacy in Deep Learning]

Further Research Scope / Our Proposed Works

- Heterogeneous Scenario
- Sync-Switch
- Model Compression
- Quantization

Thank you