Group Knowledge Transfer: Federated Learning of Large CNNs at the Edge

NIPS'20

Presented by,

Pavana Prakash

Chaoyang He, Murali Annavaram, Salman Avestimehr

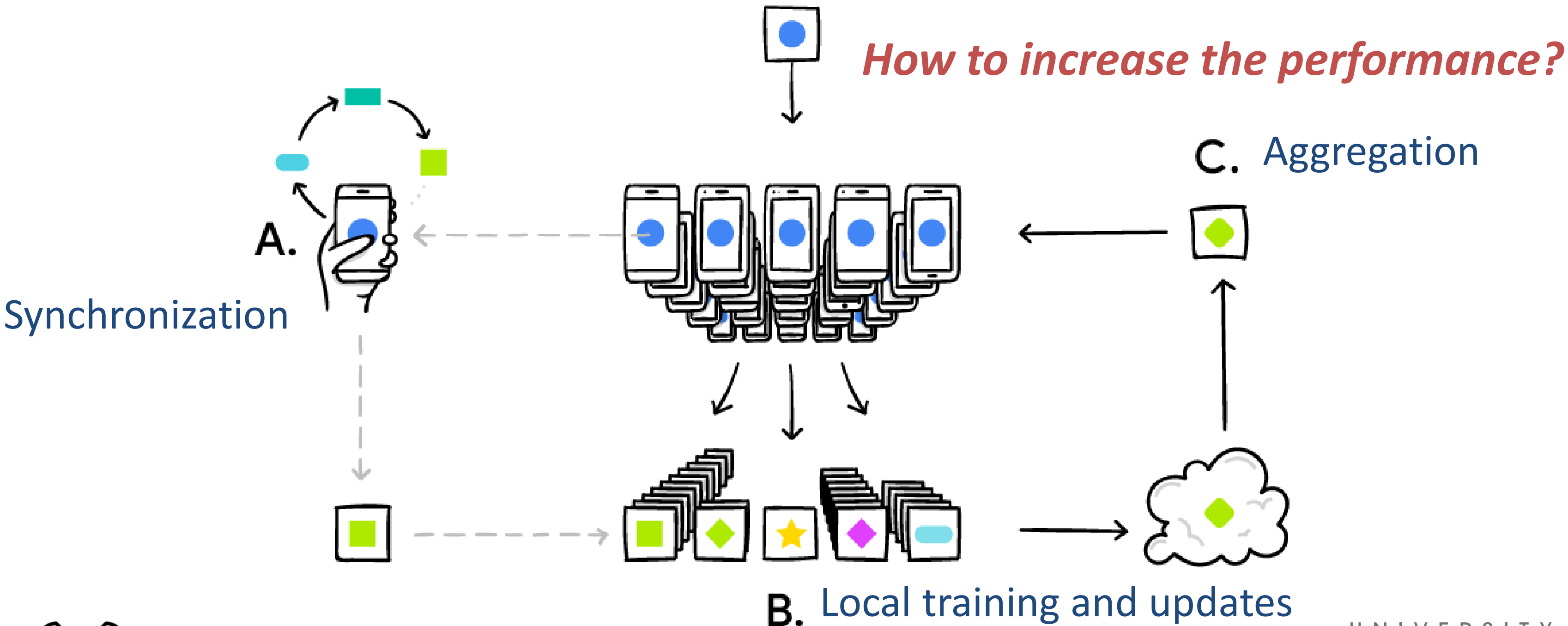
University of Southern California

Agenda

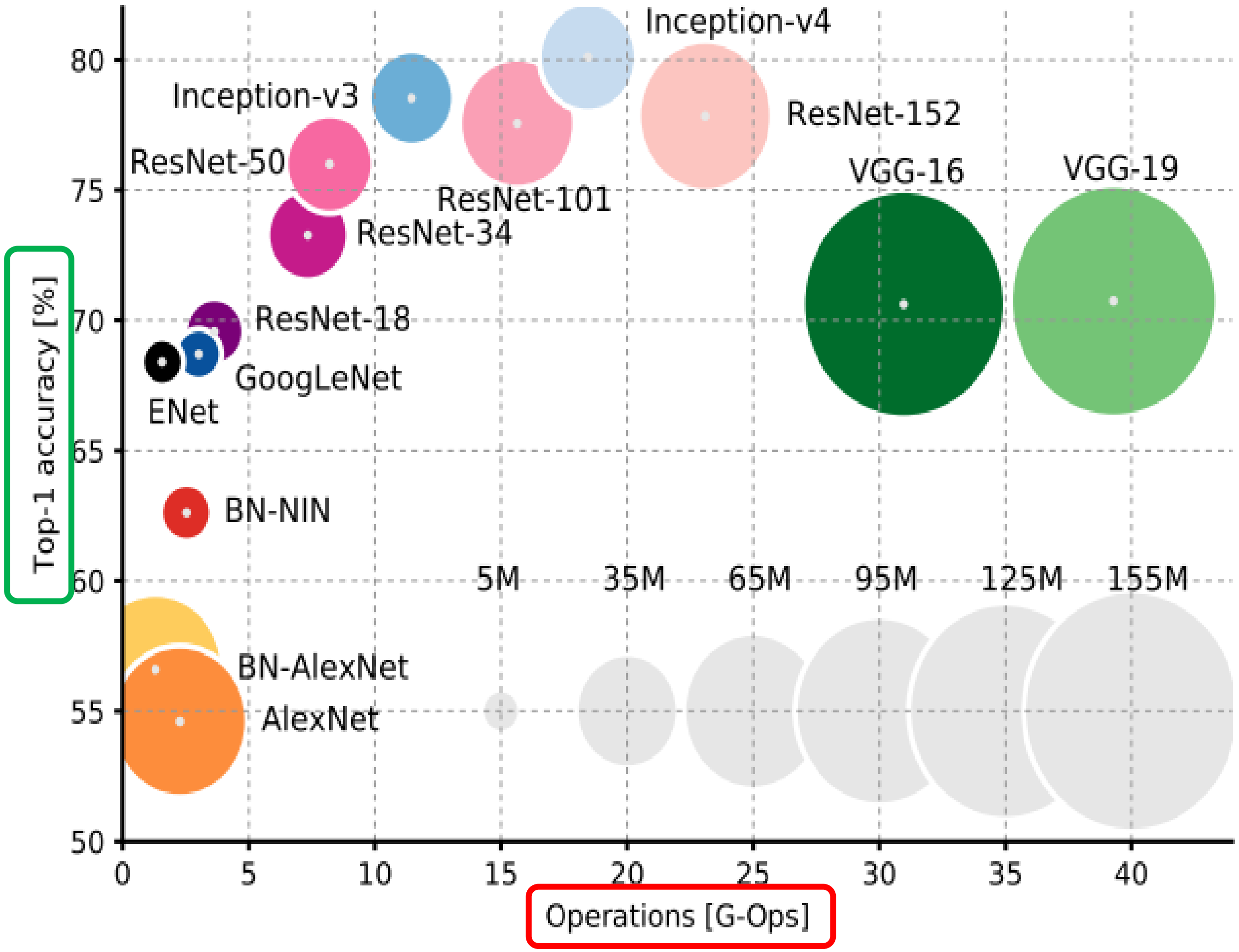
- Motivation
- Preliminaries
- ***FedGKT***: Group Knowledge Transfer
- Experiments
- Conclusion

Introduction

- Federated learning (FL) :
  - No exchange of data
  - Enables devices to learn collaboratively from a shared model
- *Components*: coordinator + collection of devices



Motivation



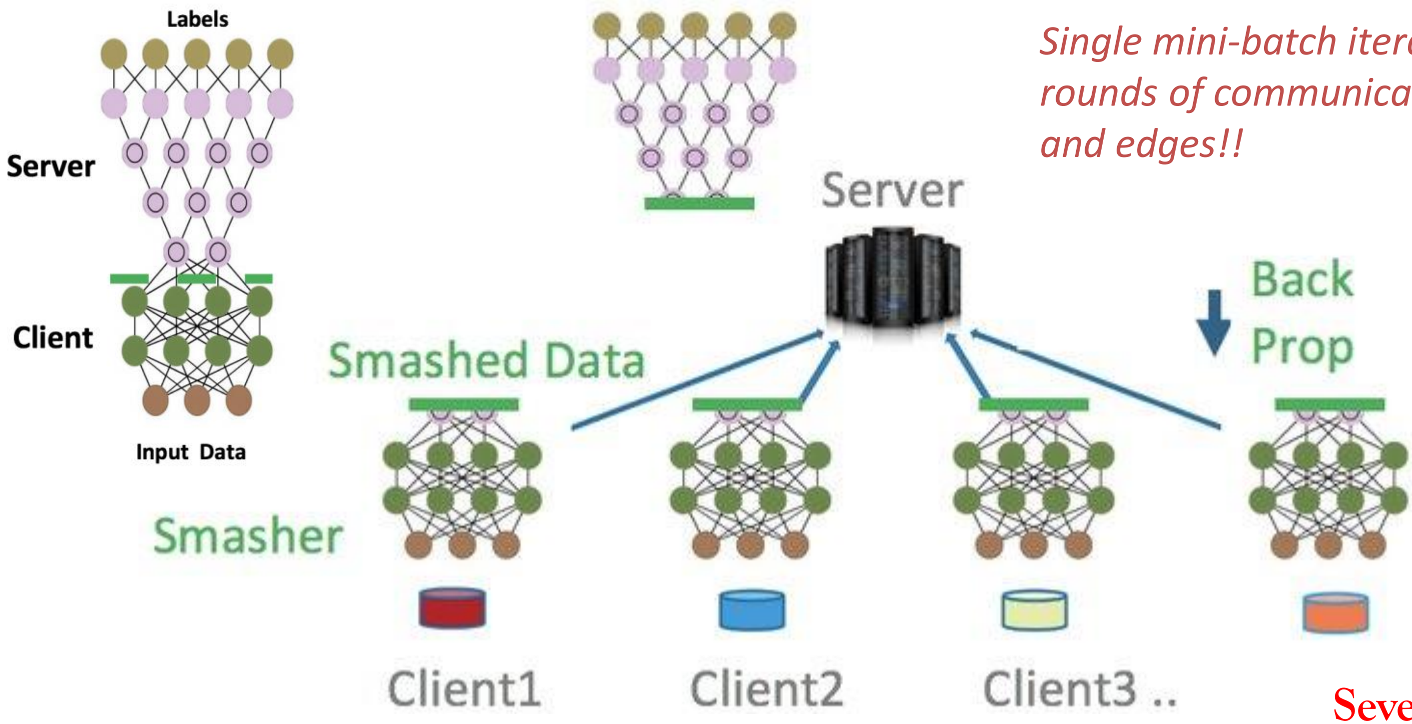
Scaling up CNN size known to effectively improve model accuracy.

Large model impedes training on resource-constrained devices!

FL may place undue burden on the compute capability of edge nodes

Source : A. Canziani, A. Paszke, and E. Culurciello. An analysis of deep neural network models for practical applications. In IEEE International Symposium on Circuits & Systems, 2016.

Preliminary - Split Learning



Single mini-batch iteration requires multiple rounds of communication between server and edges!!

Severe straggler problem!

- Each client trains a partial deep network up to the cut layer
- Activations and gradients at the cut layer are sent to server which completes the rest of the training without looking at raw data.

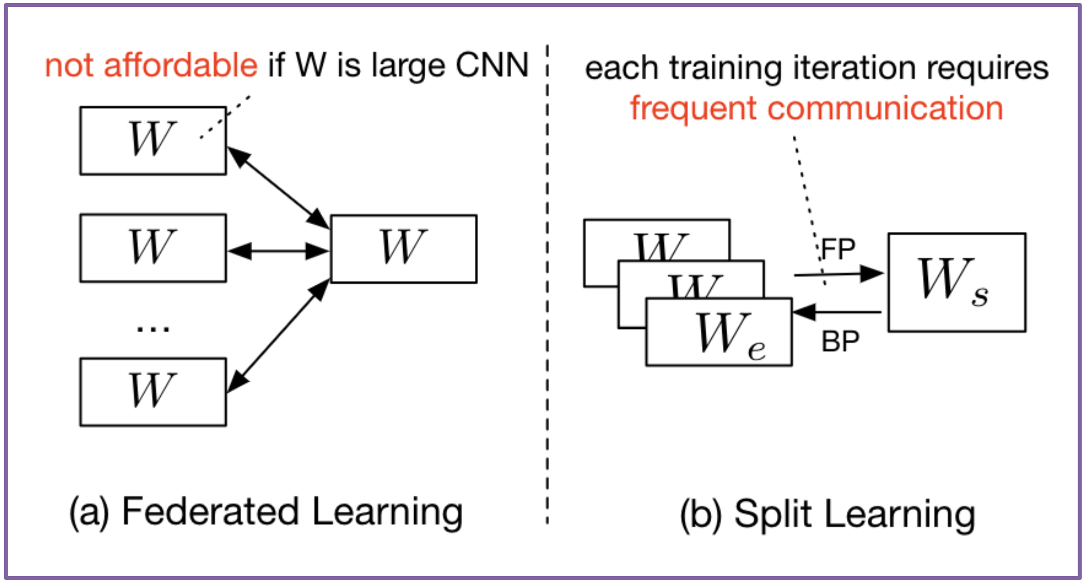
Singh, Abhishek, Praneeth Vepakomma, Otkrist Gupta, and Ramesh Raskar. "Detailed comparison of communication efficiency of split learning and federated learning." arXiv preprint arXiv:1909.09145 (2019).

ANTS LAB

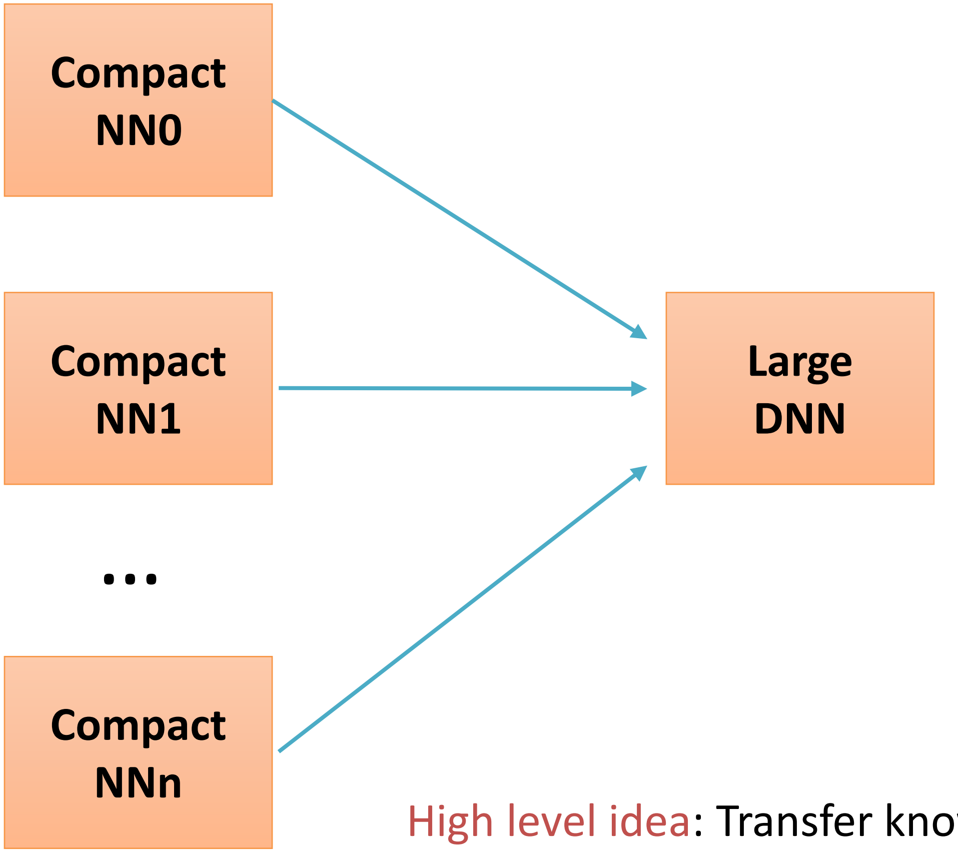
UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

Motivation

- Unrealistic FL assumption - client has enough computational power with GPUs to train large CNNs
  - Expensive computation
- Split learning has a severe straggler problem
  - Expensive communication



Overview - FedGKT



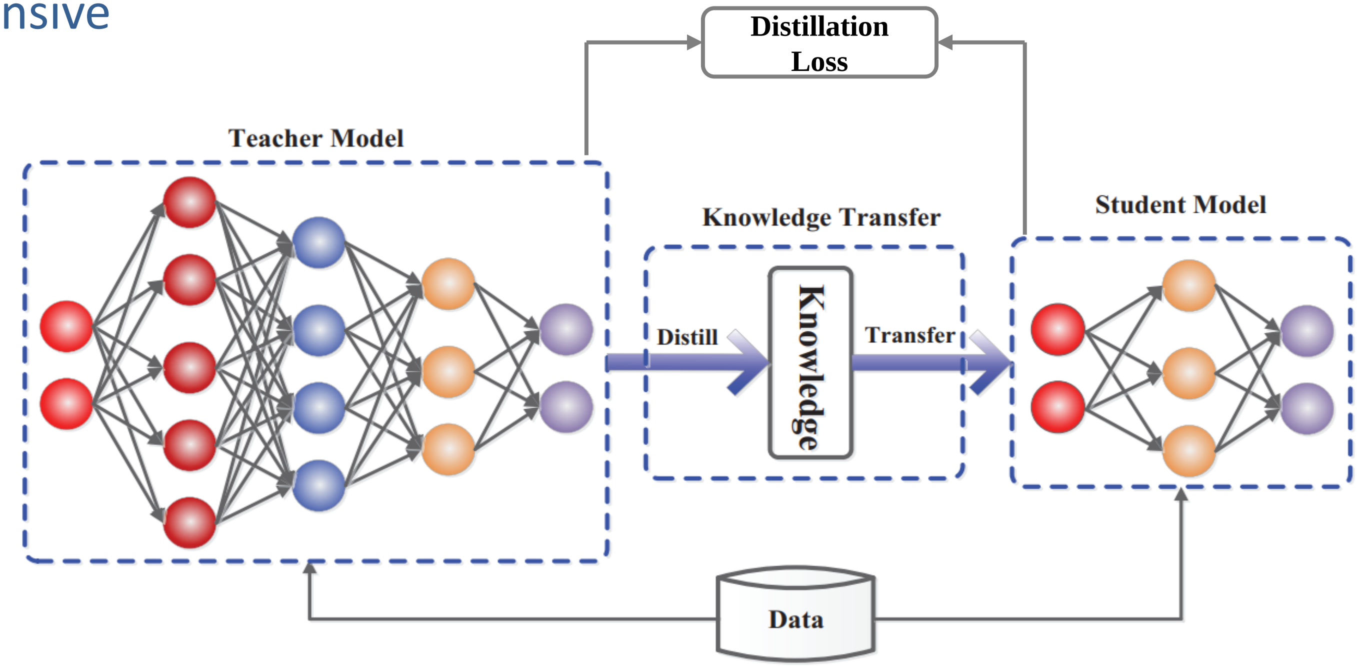
High level idea: Transfer knowledge from many small networks to a larger one which has more capacity to obtain high accuracy.

ANTS LAB

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

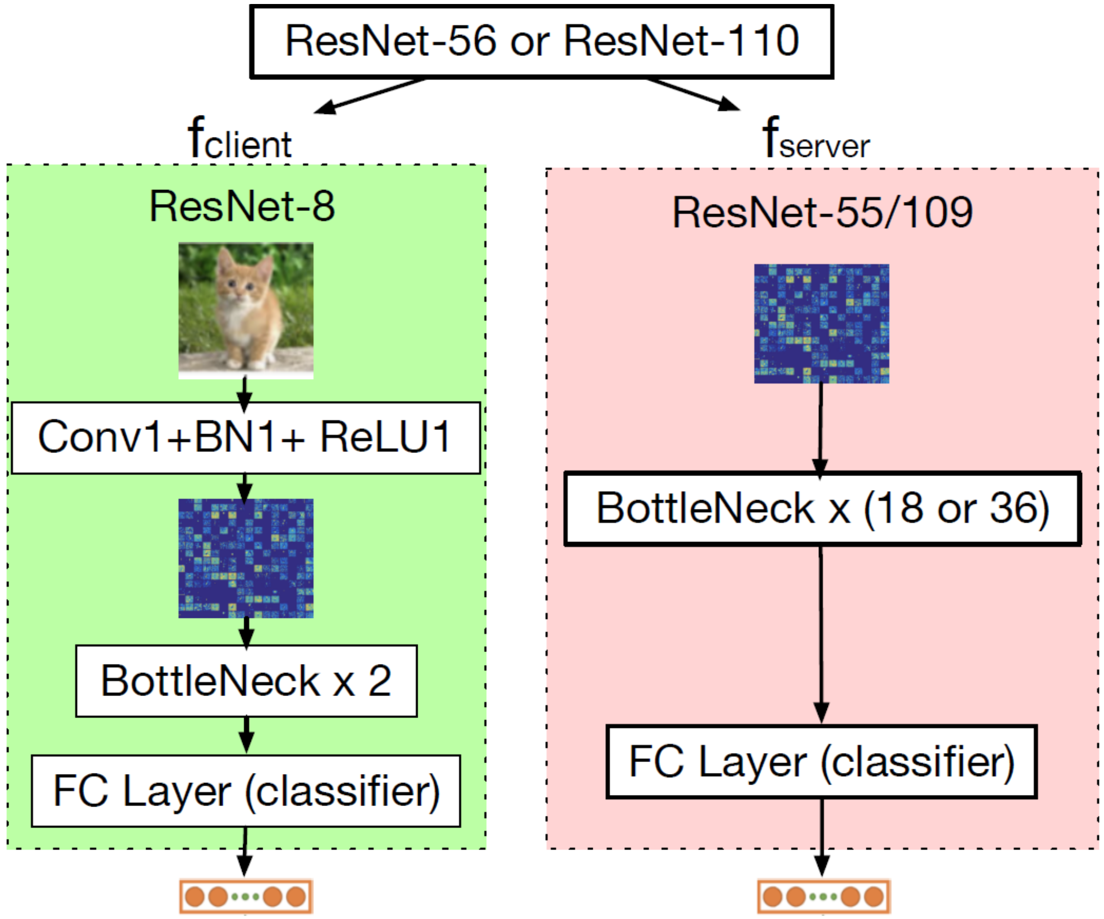
Preliminary – Knowledge Distillation (KD)

- *What is KD?* Process of transferring knowledge from a large model to a smaller one without loss of validity
- *Why KD?* Large models (very DNN) have higher knowledge capacity than small models but computationally expensive; and small models are less expensive



Gou, Jianping, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. "Knowledge distillation: A survey." International Journal of Computer Vision 129, no. 6 (2021): 1789-1819.

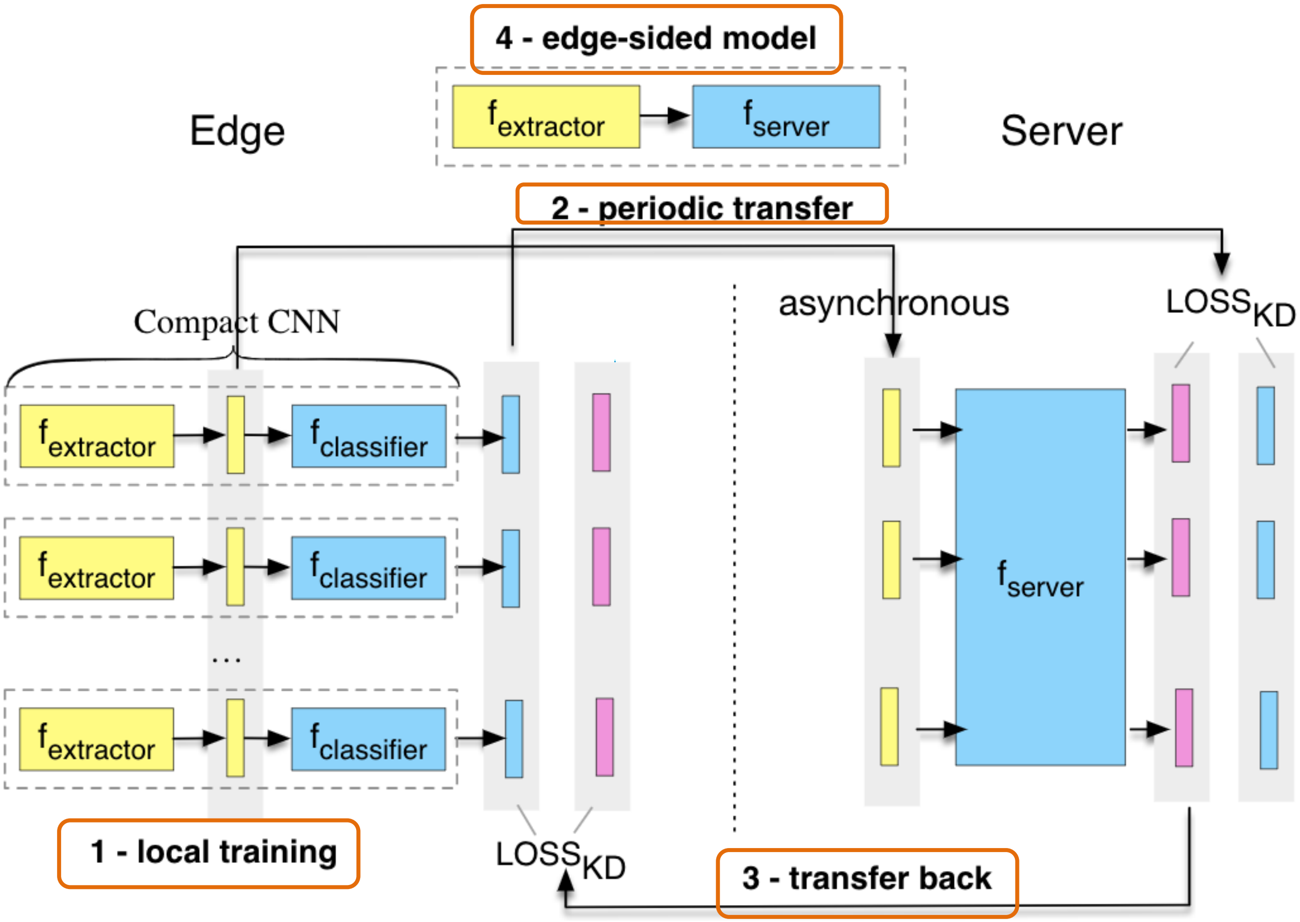
FL Reformulation: Group Knowledge Transfer



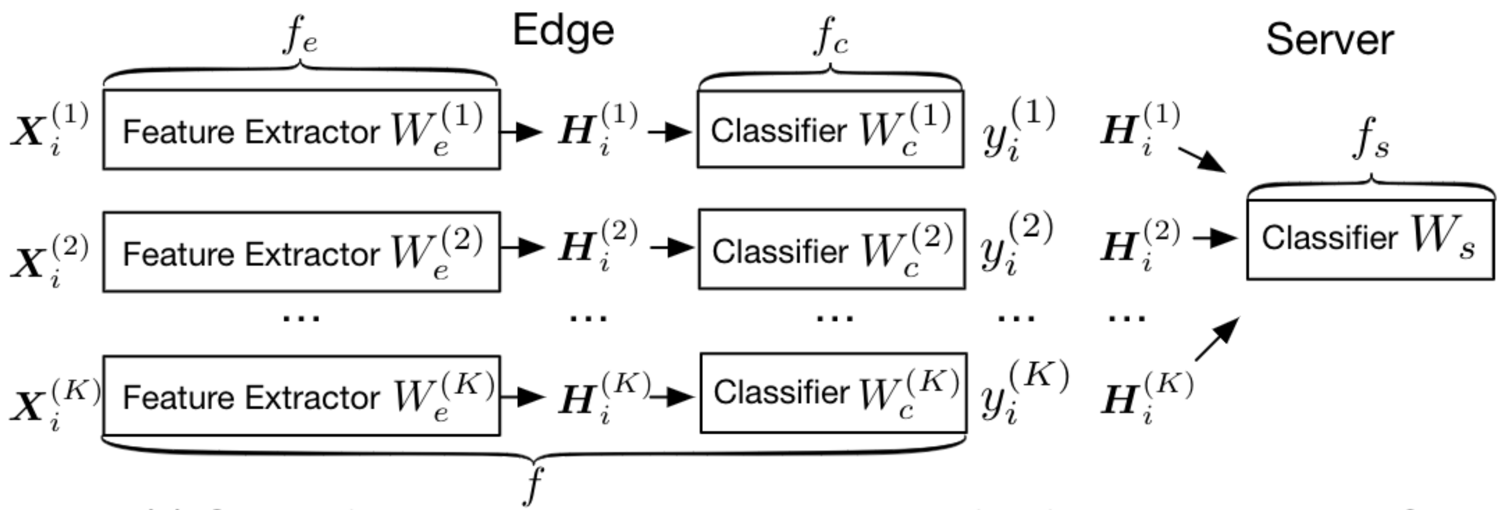
CNN architectures on the edge and server

FedGKT

Alternating and periodical knowledge transfer



Method: FedGKT



(c) Our Reformulation: consolidating benefits from both FL and SL

- ✓ Communication bandwidth for transferring H to the server is substantially less than communicating all model parameters!

- X – training sample
- Y – label
- K clients
- N samples
- W – network weight
- H – feature map (hidden vector)
- f_e, f_c, f_s – architecture of feature extractor, classifier, server model, respectively.

Algorithm – server-side

ServerExecute():
for each round $t = 1, 2, ..., T$ **do**
  for each client k **in parallel do**
    // the server broadcasts $\boldsymbol{Z}_c^{(k)}$ *to the client*
    $\boldsymbol{H}^{(k)}, \boldsymbol{Z}_c^{(k)}, \boldsymbol{Y}^{(k)} \leftarrow$ **ClientTrain**$(k, \boldsymbol{Z}_s^{(k)})$
  $\boldsymbol{Z}_s \leftarrow$ empty dictionary
  for each local epoch i from 1 to E_s **do**
    for each client k **do**
      for *idx*, $\boldsymbol{b} \in \{\boldsymbol{H}^{(k)}, \boldsymbol{Z}_c^{(k)}, \boldsymbol{Y}^{(k)}\}$ **do**
        $\boldsymbol{W}_s \leftarrow \boldsymbol{W}_s - \eta_s \nabla \ell_s(\boldsymbol{W}_s; \boldsymbol{b})$
        if $i == E_s$ **then**
          $\boldsymbol{Z}_s^{(k)}[idx] \leftarrow f_s(\boldsymbol{W}_s; \boldsymbol{h}^{(k)})$
  // illustrated as "transfer back" in Fig. 1(a)
  for each client k **in parallel do**
    send the server logits $\boldsymbol{Z}_s^{(k)}$ to client k

Algorithm – client-side

ClientTrain$(k, \boldsymbol{Z}_s^{(k)})$:
// illustrated as "local training "in Fig. 1(a)
for each local epoch i from 1 to E_c **do**
    for batch $\boldsymbol{b} \in \{\boldsymbol{X}^{(k)}, \boldsymbol{Z}_s^{(k)}, \boldsymbol{Y}^{(k)}\}$ **do**
        // $\ell_c^{(k)}$ is computed using Eq. (7)
        $\boldsymbol{W}^{(k)} \leftarrow \boldsymbol{W}^{(k)} - \eta_k \nabla \ell_c^{(k)}(\boldsymbol{W}^{(k)}; \boldsymbol{b})$
// extract features and logits
$\boldsymbol{H}^{(k)}, \boldsymbol{Z}_c^{(k)} \leftarrow$ empty dictionary
for *idx*, batch $\boldsymbol{x}^{(k)}, \boldsymbol{y}^{(k)} \in \{\boldsymbol{X}^{(k)}, \boldsymbol{Y}^{(k)}\}$ **do**
    $\boldsymbol{h}^{(k)} \leftarrow f_e^{(k)}(\boldsymbol{W}_e^{(k)}; \boldsymbol{x}^{(k)})$
    $\boldsymbol{z}_c^{(k)} \leftarrow f_c(\boldsymbol{W}_c^{(k)}; \boldsymbol{h}^{(k)})$
    $\boldsymbol{H}^{(k)}[idx] \leftarrow \boldsymbol{h}^{(k)}$
    $\boldsymbol{Z}_c^{(k)}[idx] \leftarrow \boldsymbol{z}_c^{(k)}$
return $\boldsymbol{H}^{(k)}, \boldsymbol{Z}_c^{(k)}, \boldsymbol{Y}^{(k)}$ to server

FL - Distributed Optimization

FL global objective:

$$\min_{\boldsymbol{W}} F(\boldsymbol{W}) \overset{\text{def}}{=} \min_{\boldsymbol{W}} \sum_{k=1}^{K} \frac{N^{(k)}}{N} \cdot f^{(k)}(\boldsymbol{W})$$

$$f^{(k)}(\boldsymbol{W}) = \frac{1}{N^{(k)}} \sum_{i=1}^{N^{(k)}} \ell(\boldsymbol{W}; \boldsymbol{X}_i, y_i)$$

Reformulation:

Server-side

$$\underset{\boldsymbol{W}_s}{\operatorname{argmin}} F_s(\boldsymbol{W}_s, \boldsymbol{W}_e^*) = \underset{\boldsymbol{W}_s}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{i=1}^{N^{(k)}} \ell_s \left(f_s(\boldsymbol{W}_s; \boldsymbol{H}_i^{(k)}), y_i^{(k)} \right)$$

$$\textit{subject to: } \boldsymbol{H}_i^{(k)} = f_e^{(k)}(\boldsymbol{W}_e^{(k)}; \boldsymbol{X}_i^{(k)})$$

Client-side

$$\underset{(\boldsymbol{W}_e^{(k)}, \boldsymbol{W}_c^{(k)})}{\operatorname{argmin}} F_c(\boldsymbol{W}_e^{(k)}, \boldsymbol{W}_c^{(k)}) = \underset{(\boldsymbol{W}_e^{(k)}, \boldsymbol{W}_c^{(k)})}{\operatorname{argmin}} \sum_{i=1}^{N^{(k)}} \ell_c \left(f^{(k)}((\boldsymbol{W}_e^{(k)}, \boldsymbol{W}_c^{(k)}); \boldsymbol{X}_i^{(k)}), y_i^{(k)} \right)$$

$$= \underset{(\boldsymbol{W}_e^{(k)}, \boldsymbol{W}_c^{(k)})}{\operatorname{argmin}} \sum_{i=1}^{N^{(k)}} \ell_c \big(f_c^{(k)}(\boldsymbol{W}_c^{(k)}; \underbrace{f_e^{(k)}(\boldsymbol{W}_e^{(k)}; \boldsymbol{X}_i^{(k)})}_{\boldsymbol{H}_i^{(k)}}), y_i^{(k)} \big)$$

s - server, e – feature extractor of the client, c – classifier of the client

Challenges

- Each client is expected to adequately solve the inner optimization to accurately generate ***H*** but in FL, each edge device's dataset is small and may be inadequate to train an extractor solely based on the local dataset
- Outer and inner optimizations are correlated making the outer optimization difficult to converge if the client-side feature extractors are not trained adequately.
- *Hence use KD!* KL divergence loss attempts to bring the soft label and the ground truth close to each other by absorbing the knowledge gained

$$\ell_s = \ell_{CE} + \sum_{k=1}^{K} \ell_{KD}\left(\boldsymbol{z}_s, \boldsymbol{z}_c^{(k)}\right) = \ell_{CE} + \sum_{k=1}^{K} D_{KL}\left(\boldsymbol{p}_k \| \boldsymbol{p}_s\right)$$

$$\ell_c^{(k)} = \ell_{CE} + \ell_{KD}\left(\boldsymbol{z}_s, \boldsymbol{z}_c^{(k)}\right) = \ell_{CE} + D_{KL}\left(\boldsymbol{p}_s \| \boldsymbol{p}_k\right)$$

$$\boxed{\boldsymbol{p}_k^i = \frac{\exp\left(z_c^{(k,i)}/T\right)}{\sum_{i=1}^{C} \exp\left(z_c^{(k,i)}/T\right)}}$$

$$\boxed{\boldsymbol{p}_s^i = \frac{\exp\left(z_s^i/T\right)}{\sum_{i=1}^{C} \exp(z_s^i/T)}}$$

Improved Alternating Minimization

Server-side

$$\underset{\boldsymbol{W}_s}{\operatorname{argmin}} F_s(\boldsymbol{W}_s, \boldsymbol{W}_e^{(k)*}) = \underset{\boldsymbol{W}_s}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{i=1}^{N^{(k)}} \ell_{CE}\big(f_s(\boldsymbol{W}_s; \underbrace{f_e^{(k)}(\boldsymbol{W}_e^{(k)*}; \boldsymbol{X}_i^{(k)})}_{\boldsymbol{H}_i^{(k)}}), y_i^{(k)}\big) + \sum_{k=1}^{K} \ell_{KD}\big(\boldsymbol{z}_c^{(k)*}, \boldsymbol{z}_s\big)$$

$$\textit{where}\;\; \boldsymbol{z}_c^{(k)*} = f_c^{(k)}(\boldsymbol{W}_c^{(k)}; \underbrace{f_e^{(k)}(\boldsymbol{W}_e^{(k)*}; \boldsymbol{X}_i^{(k)})}_{\boldsymbol{H}_i^{(k)}}), \textit{and}\; \boldsymbol{z}_s = f_s(\boldsymbol{W}_s; \boldsymbol{H}_i^{(k)})$$

Client-side

$$\underset{\boldsymbol{W}^{(k)}}{\operatorname{argmin}} F_c(\boldsymbol{W}_s^*, \boldsymbol{W}^{(k)}) = \underset{\boldsymbol{W}^{(k)}}{\operatorname{argmin}} \sum_{i=1}^{N^{(k)}} \ell_{CE}\big(f_c^{(k)}(\boldsymbol{W}_c^{(k)}; \underbrace{f_e^{(k)}(\boldsymbol{W}_e^{(k)}; \boldsymbol{X}_i^{(k)})}_{\boldsymbol{H}_i^{(k)}}), y_i^{(k)}\big) + \ell_{KD}\big(\boldsymbol{z}_s^*, \boldsymbol{z}_c^{(k)}\big)$$

$$\textit{where}\;\; \boldsymbol{z}_c^{(k)} = f_c^{(k)}(\boldsymbol{W}_c^{(k)}; \underbrace{f_e^{(k)}(\boldsymbol{W}_e^{(k)}; \boldsymbol{X}_i^{(k)})}_{\boldsymbol{H}_i^{(k)}}), \textit{and}\; \boldsymbol{z}_s^* = f_s(\boldsymbol{W}_s^*; \boldsymbol{H}_i^{(k)})$$

Optimize two random variables (weights) alternatively by fixing one and optimizing (training) the other for several epochs until convergence.

Experiments

- Implementation
  - FedML, an open-source federated learning research library
  - Server node has 4 NVIDIA RTX 2080Ti GPUs
  - CPU-based nodes as clients training small CNNs.
- Task and Dataset
  - CIFAR-10, CIFAR-100, CINIC-10 (I.I.D. and non-I.I.D.)
  - 16 edge clients
- Baselines
  - FedAvg
  - Split Learning-based method
- Model Architectures
  - ResNet-56
  - ResNet-110 on the server
  - ResNet-8 on the edge-sided partition on the clients

Test Accuracy

Test Accuracy of ResNet-56 and ResNet-110 on Three Datasets.

Model	Methods	CIFAR-10		CIFAR-100		CINIC-10	
		I.I.D.	non-I.I.D.	I.I.D.	non-I.I.D.	I.I.D.	non-I.I.D.
ResNet-56	**FedGKT (ResNet-8, ours)**	**92.97**	**86.59**	**69.57**	**63.76**	**81.51**	**77.80**
	FedAvg (ResNet-56)	92.88	86.60	68.09	63.78	81.62	77.85
	Centralized (ResNet-56)	93.05		69.73		81.66	
	Centralized (ResNet-8)	78.94		37.67		67.72	
ResNet-110	**FedGKT (ResNet-8, ours)**	**93.47**	**87.18**	**69.87**	**64.31**	**81.98**	**78.39**
	FedAvg (ResNet-56)	93.49	87.20	68.58	64.35	82.10	78.43
	Centralized (ResNet-56)	93.58		70.18		82.16	
	Centralized (ResNet-8)	78.94		37.67		67.72	

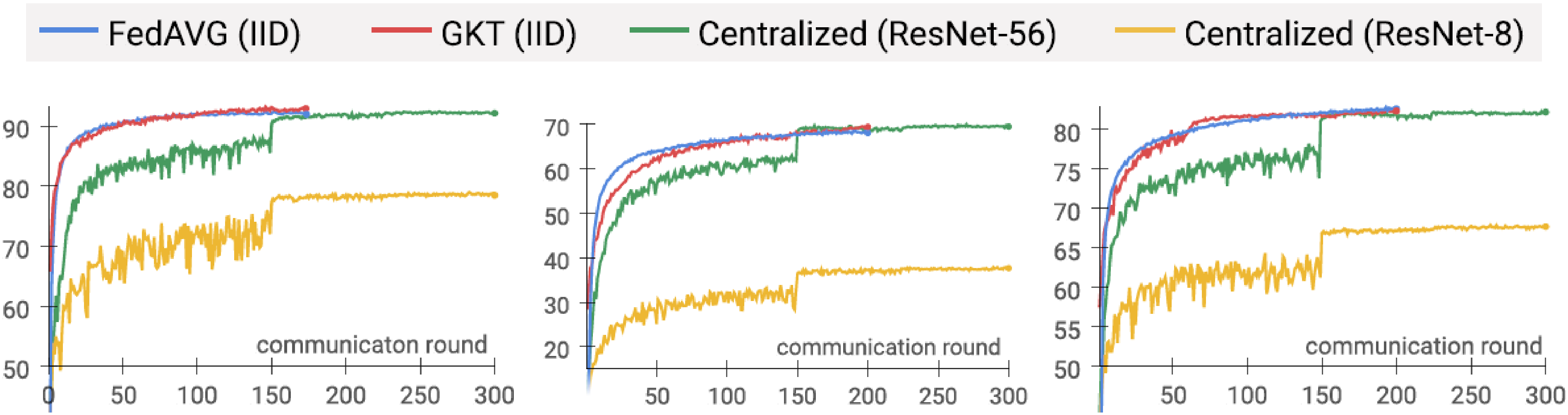
FedGKT has superior accuracy in case of Non-IID and nearly the same in case of IID

ANTS LAB

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

Results

Test Accuracy of ResNet-56 (Edge Number = 16)

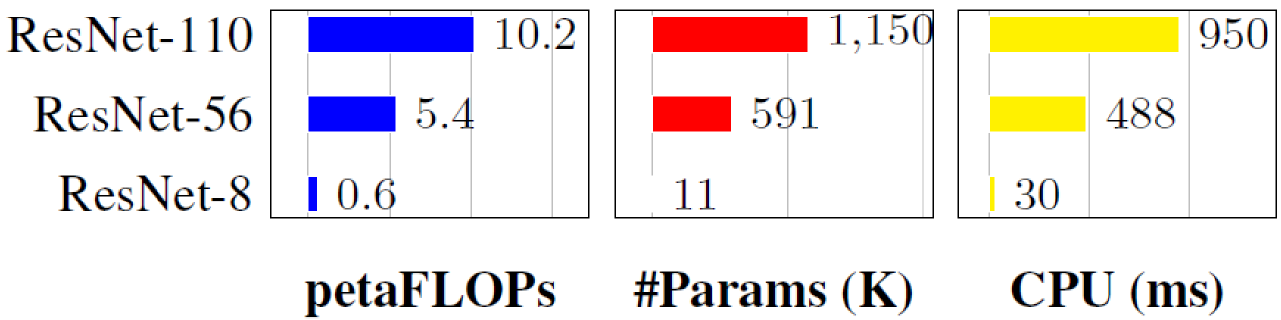


(a) ResNet-56 on CIFAR-10 (b) ResNet-56 on CIFAR-100 (c) ResNet-56 on CINIC-10

FedGKT obtains comparable or even better accuracy than FedAvg in both I.I.D. and non-I.I.D. settings

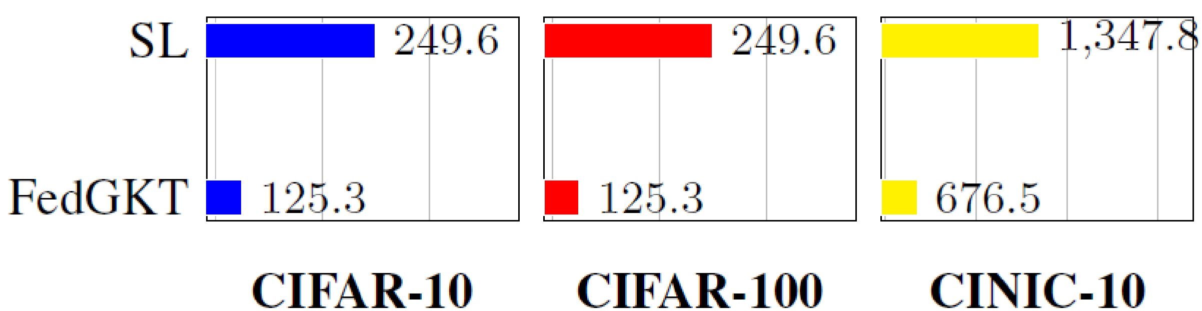
Efficiency Evaluation

Edge Computational Efficiency (CIFAR-100)



Compared to the FedAvg baseline, the computational cost on the edge of FedGKT (ResNet-8) is 9 times less than that of ResNet-56 and 17 times less than that of ResNet-110

Communication Efficiency (ResNet-56)



FedGKT uses fewer feature map exchanges with the server than SL

ANTS LAB

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

Evaluations

FedGKT with Different # of Edge

	8	16	64	128
FedGKT	69.51	69.57	69.65	69.59

Adding more edge nodes does not negatively affect accuracy

Small CNNs on CIFAR-10

	ResNet-4	ResNet-6	ResNet-8
Test Accuracy	88.86	90.32	92.97

When the model size is reduced even more substantially, it does reduce the overall accuracy

Conclusion

- FedGKT is memory and computation efficient, similar to SL.
- Reduces communication frequency by training in a local SGD manner like FedAvg
- Reduces communication bandwidth requirement by exchange of hidden features (as in SL), as opposed to exchanging the entire model (as in FedAvg)
- Naturally supports asynchronous training, which circumvents the severe synchronization issue in SL.

Discussion

- **Privacy and robustness**
  - Since hidden map exchange happens at the training phase, attacker's access is limited to the evolving and untrained feature map.
  - But, given that the model and gradient exchange may also leak privacy, lack of analysis between gradient, model, and hidden map is a limitation.
- **Communication cost**
  - Compared to the entire model weight or gradient, hidden vector is definitely much smaller.
- **Label deficiency**
  - FedGKT can only work on supervised learning.
- **Model personalization**
  - The final trained model is a combination of the global server model and the client model, to help clients learn personalized models. For example, we can fine-tune the client model for several epochs.

THANK YOU

ANTS LAB

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

Pavana Prakash

Department of Electrical and Computer Engineering
University of Houston
Houston, TX

UNIVERSITY of HOUSTON | ENGINEERING