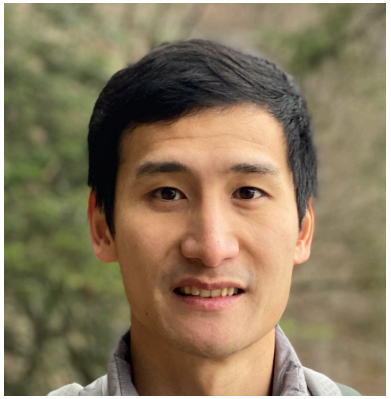
Federated Learning with Matched Averaging

Hongyi Wang
(University of Wisconsin-Madison)

Joint work with:

Mikhail Yurochkin
(IBM Research)



Yuekai Sun
(U Michigan)

Dimitris Papailiopouslos
(U Wisconsin-Madison)

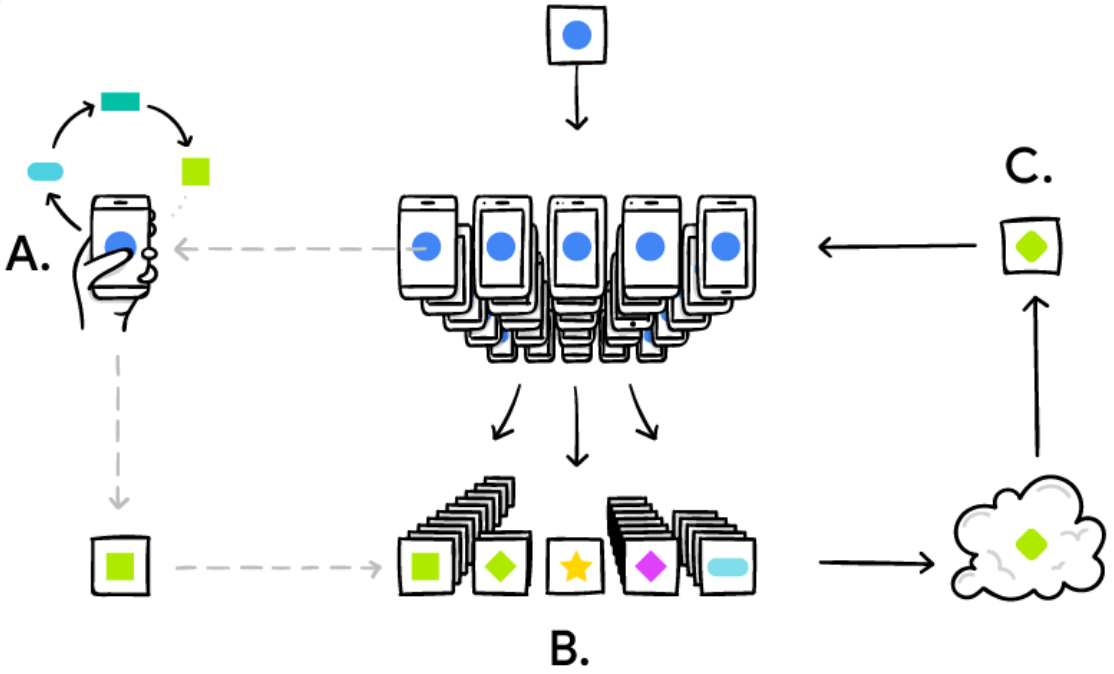
Yasaman Khazaeni
(IBM Research)

ICLR

IBM Research

MIT-IBM Watson AI Lab

Brief Introduction to Federated Learning



A. Users' local usage (i.e. data collection + local training)
B. Data Center Aggregation
C. Model Sharing

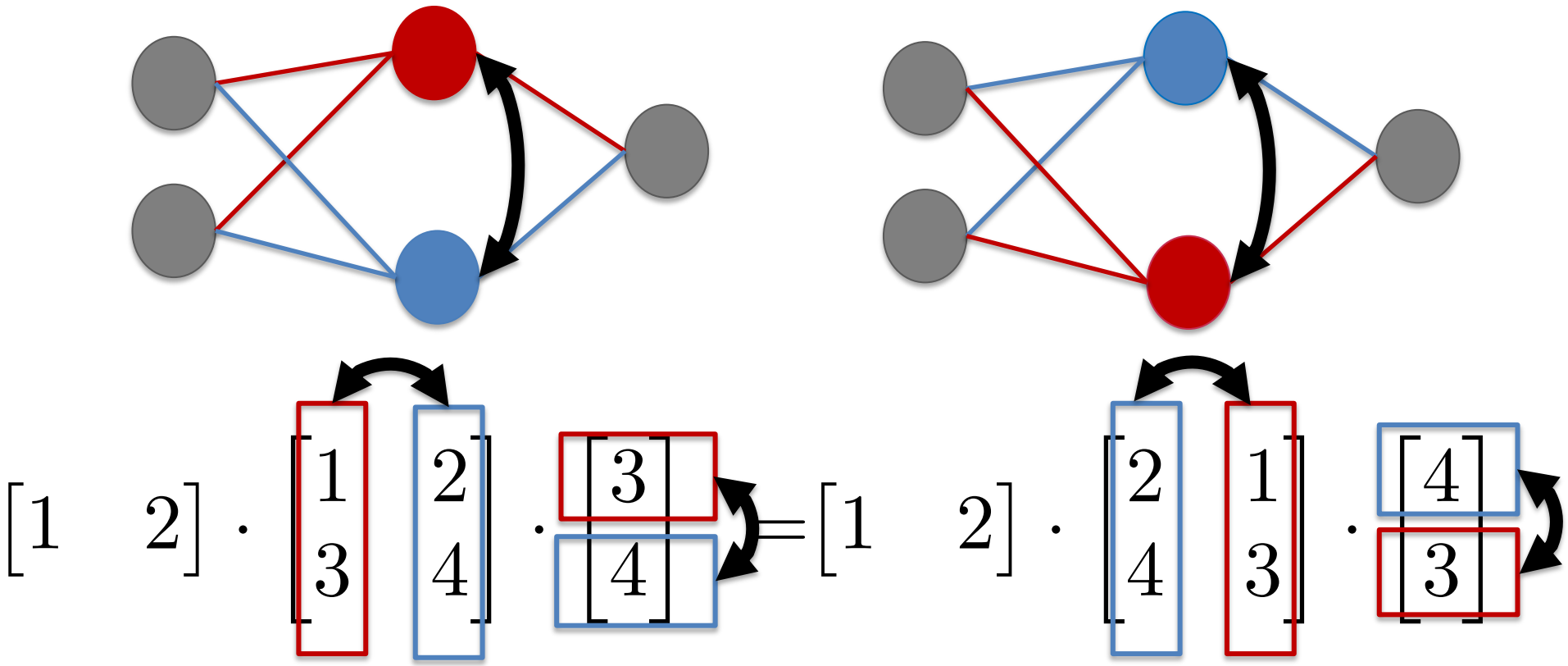
TL;DR: "*Bring machine learning algorithms to the data instead of bringing data to the algorithms*".

Challenges:

- Local data generated by users **heterogeneously**
- State-of-the-art federated learning approach (e.g. FedAvg) can lead to **poor convergence** performance under some cases
- Federated learning hasn't been well-explored on **large-scale applications**

ICLR

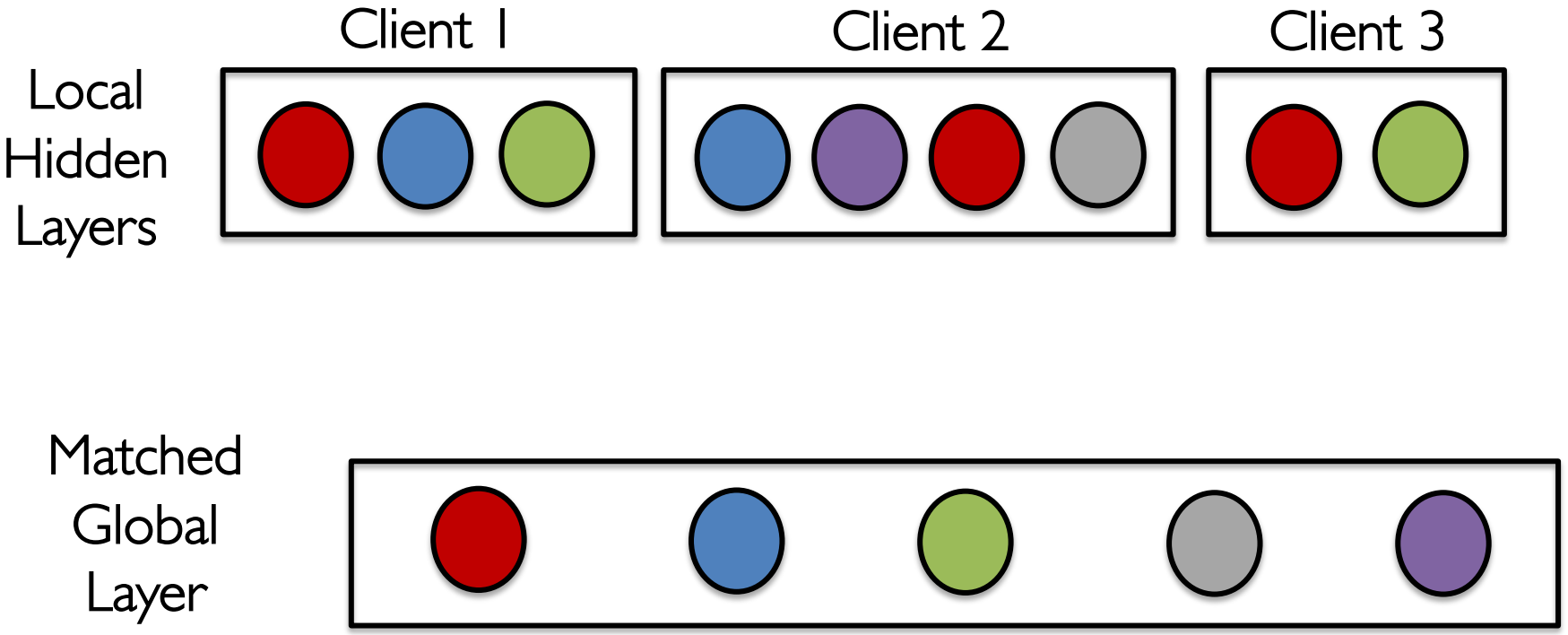
Permutation Invariance of Neural Network



TL;DR: "*Due to* ***permutation invariance*** *of hidden neurons, direct element-wise averaging may not be a good idea. We should instead find a way to* ***align*** *hidden neurons before averaging them.*"

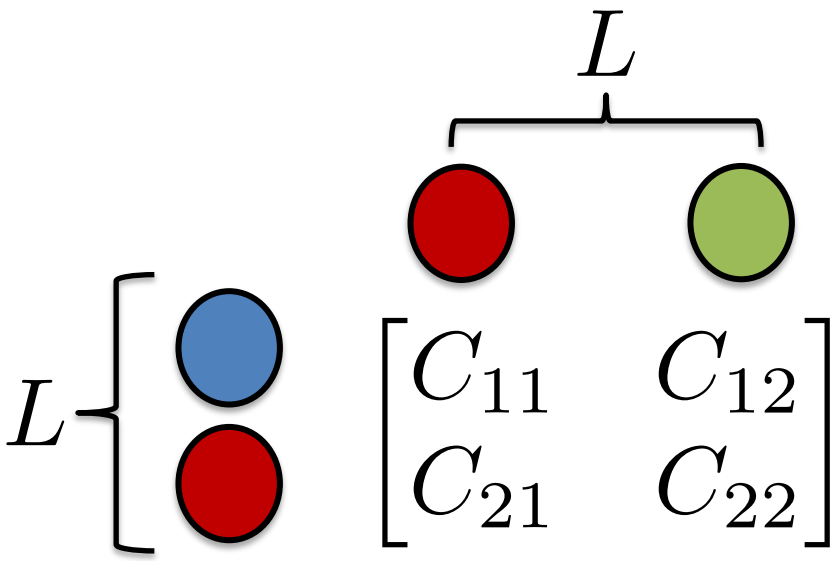
Probabilistic federated neural matching (PFNM) (*Yurochkin et.al @ ICML19'*)

Neural Matching Formulation



Probabilistic federated neural matching (PFNM) (*Yurochkin et.al @ ICML19'*)

Neural Matching Formulation

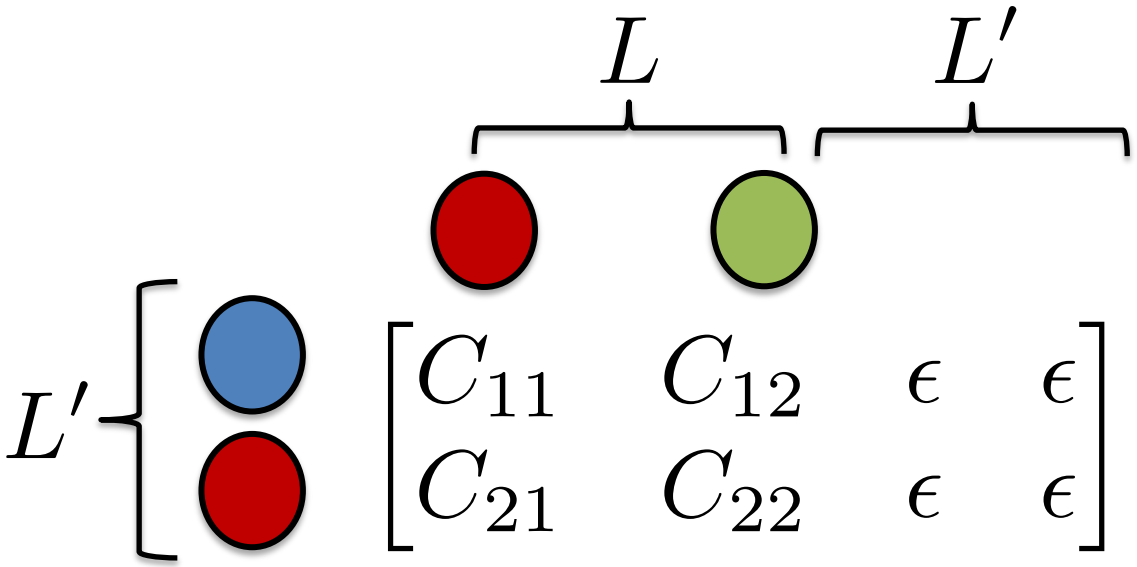


$$\min_{B} \sum_{l=1}^{L} \sum_{i=1}^{L} B_{li} C_{li}$$

$$\text{s.t.} \sum_{i} B_{li} = 1, \forall l,$$

$$\sum_{l} B_{li} = 1, \forall i.$$

Neural Matching Formulation

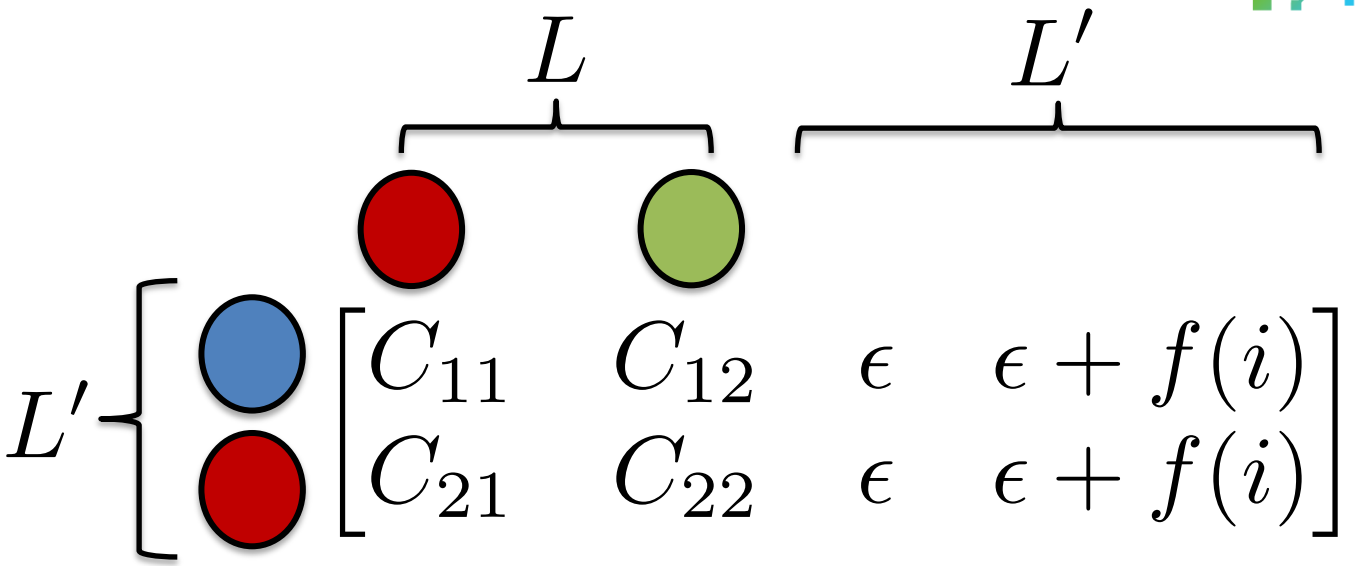


$$\min_{B} \sum_{l=1}^{L'} \sum_{i=1}^{L+L'} B_{li} C_{li}$$

$$\text{s.t.} \sum_{i} B_{li} = 1, \forall\ l,$$

$$\sum_{l} B_{li} = 1, \forall\ i; B_{li} \in \{0, 1\}, \forall\ i, l.$$

Neural Matching Formulation

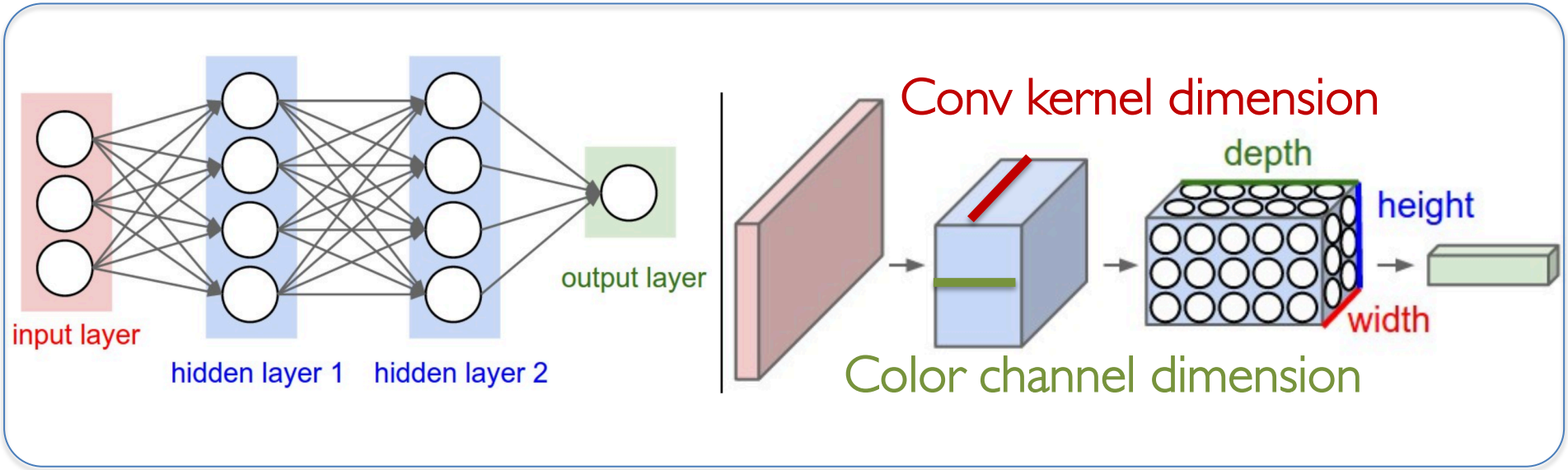


$$\min_B \sum_{l=1}^{L'} \sum_{i=1}^{L+L'} B_{li} C_{li} \qquad C_{li} = \begin{cases} c(\theta_l, \theta_i), & i \leq L \\ \epsilon, & L < i \leq L + L' \end{cases}$$

$$\text{s.t.} \sum_i B_{li} = 1, \forall\ l, \qquad f(i) \text{ increasing in } i$$

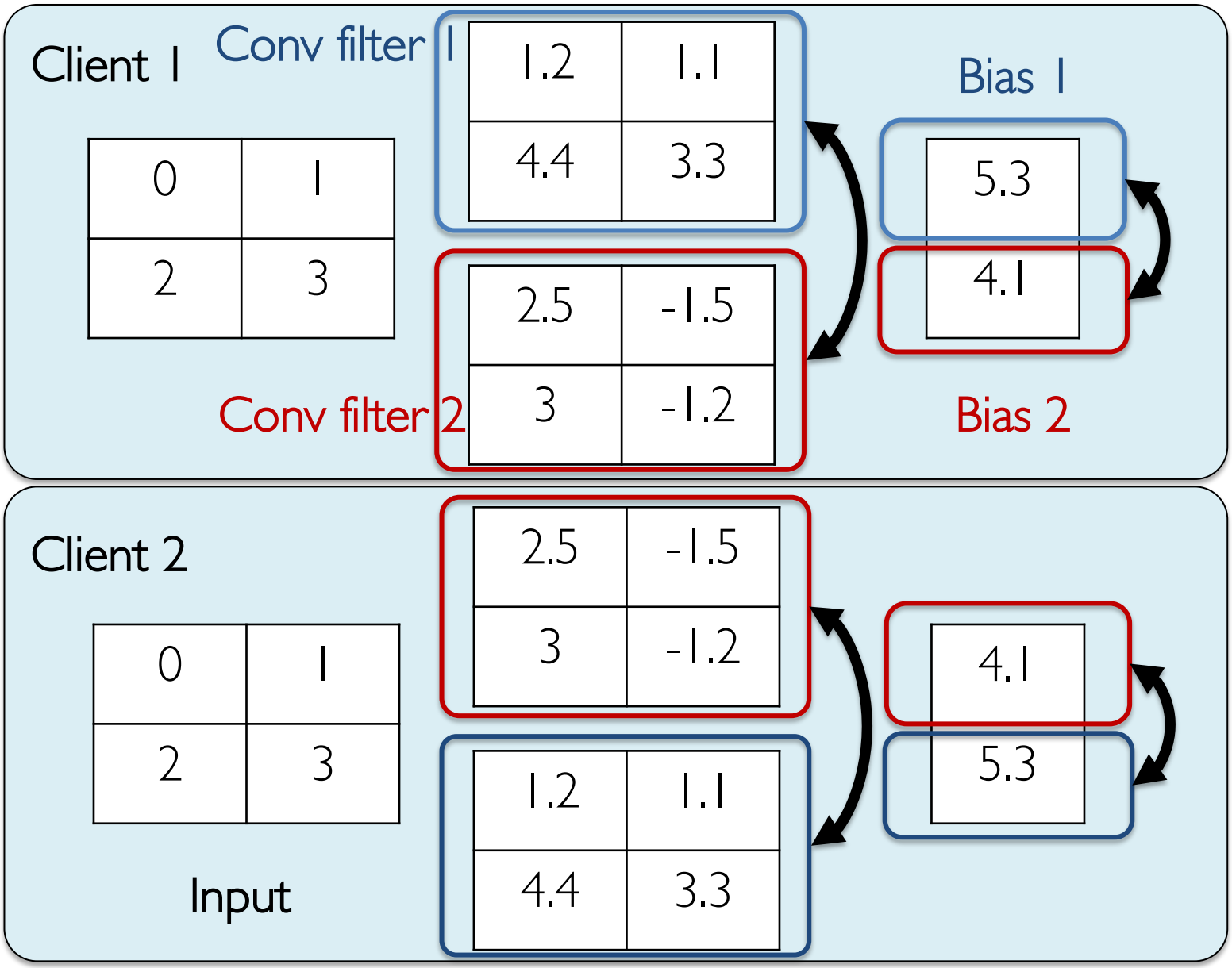
$$\sum_l B_{li} = 1, \forall\ i; B_{li} \in \{0, 1\}, \forall\ i, l.$$

Permutation Invariance Class in CNN



- The **convolutional kernels** in a convolution layer are permutation invariant.
- To match convolutional layers, we conduct matching over the **kernel dimension**.

Permutation Invariance Class in CNN



Permutation Invariance Class in LSTM

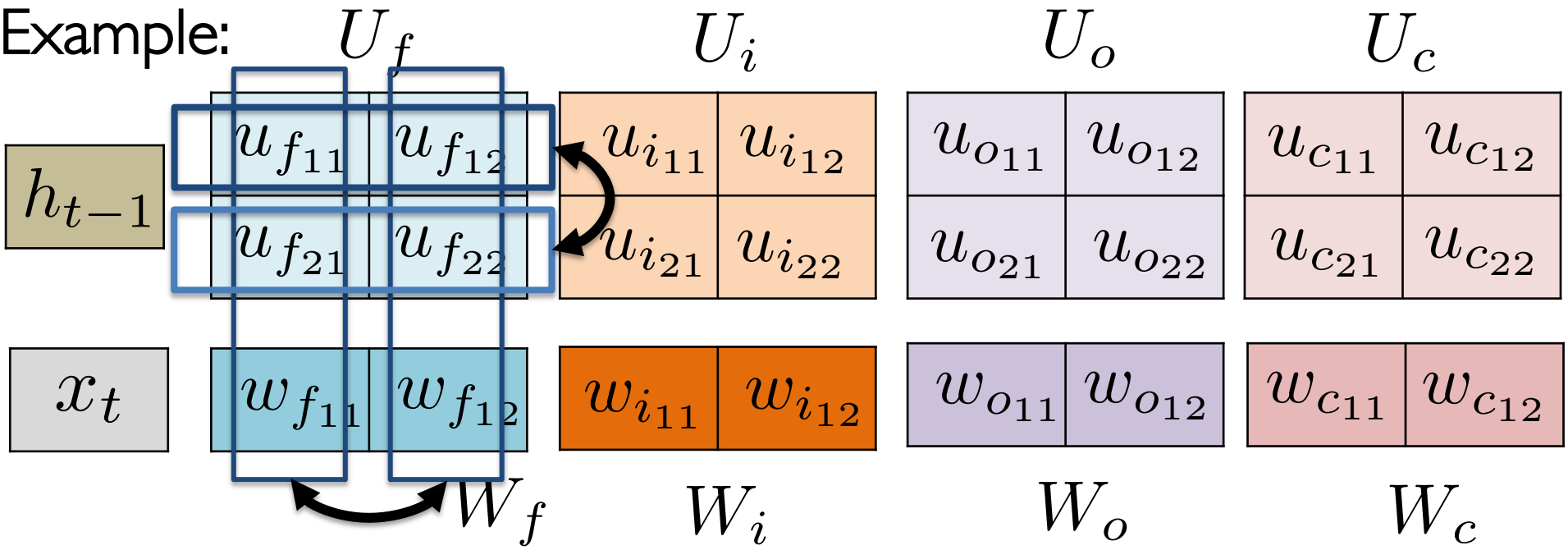
ICLR

$$h_t = \sigma(h_{t-1}\Pi^\top U\Pi + x_t W\Pi)$$

Matching U requires $\|\Pi^\top U_j \Pi - U_{j'}\|_2^2$ which is NP-hard!

Fortunately, we can directly use the same permutation appear in $W\Pi$ to permutate U.

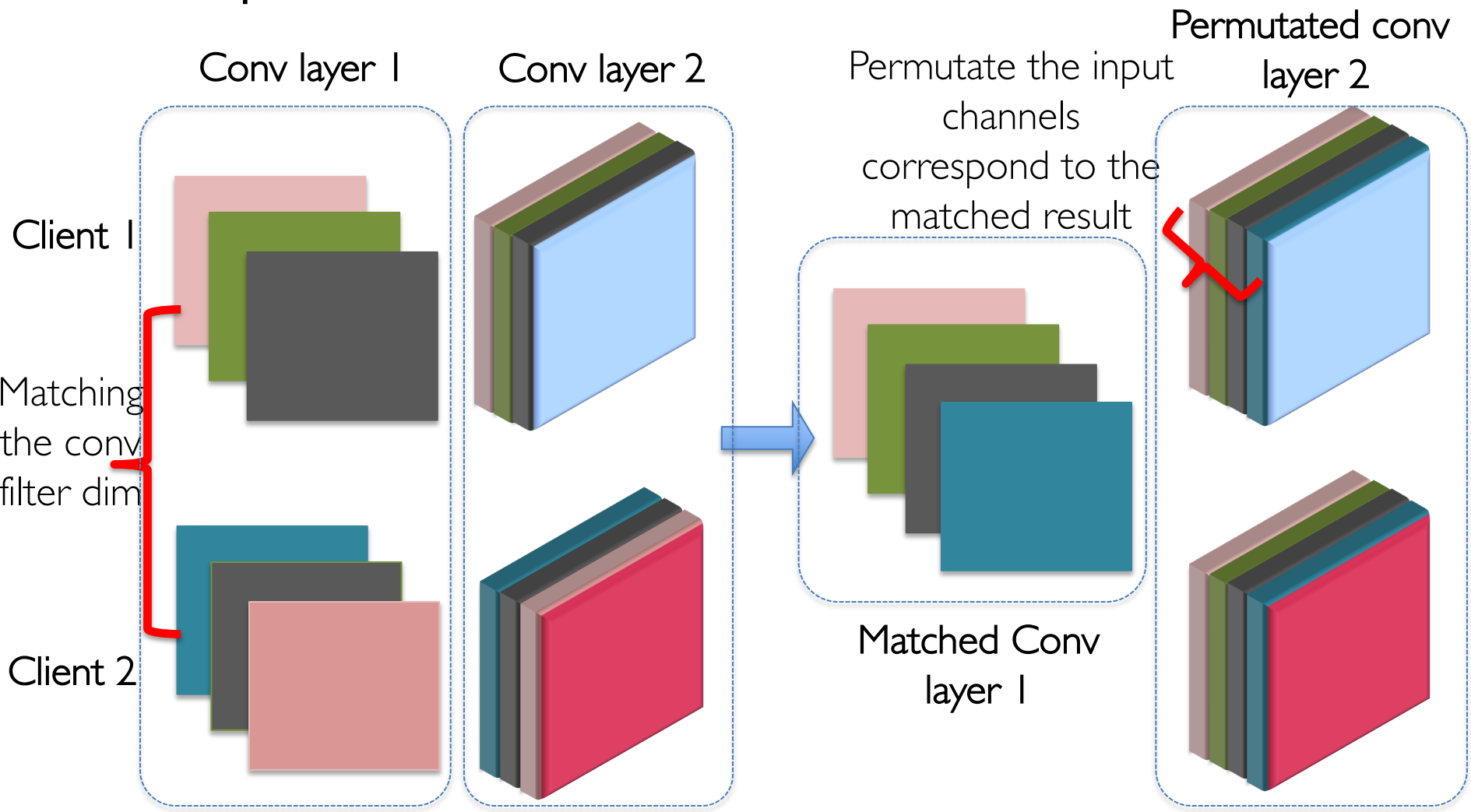
Example:



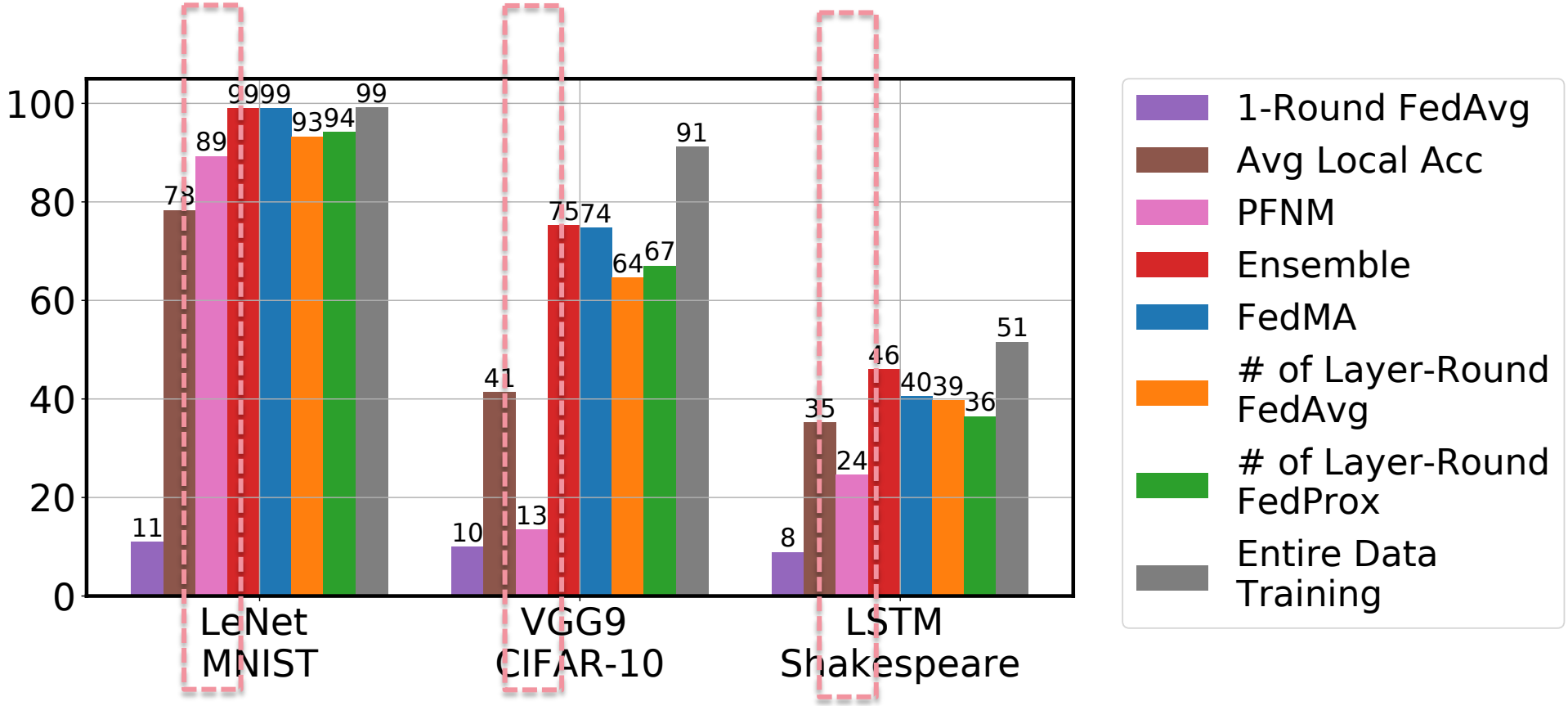
One-shot Matching

ICLR

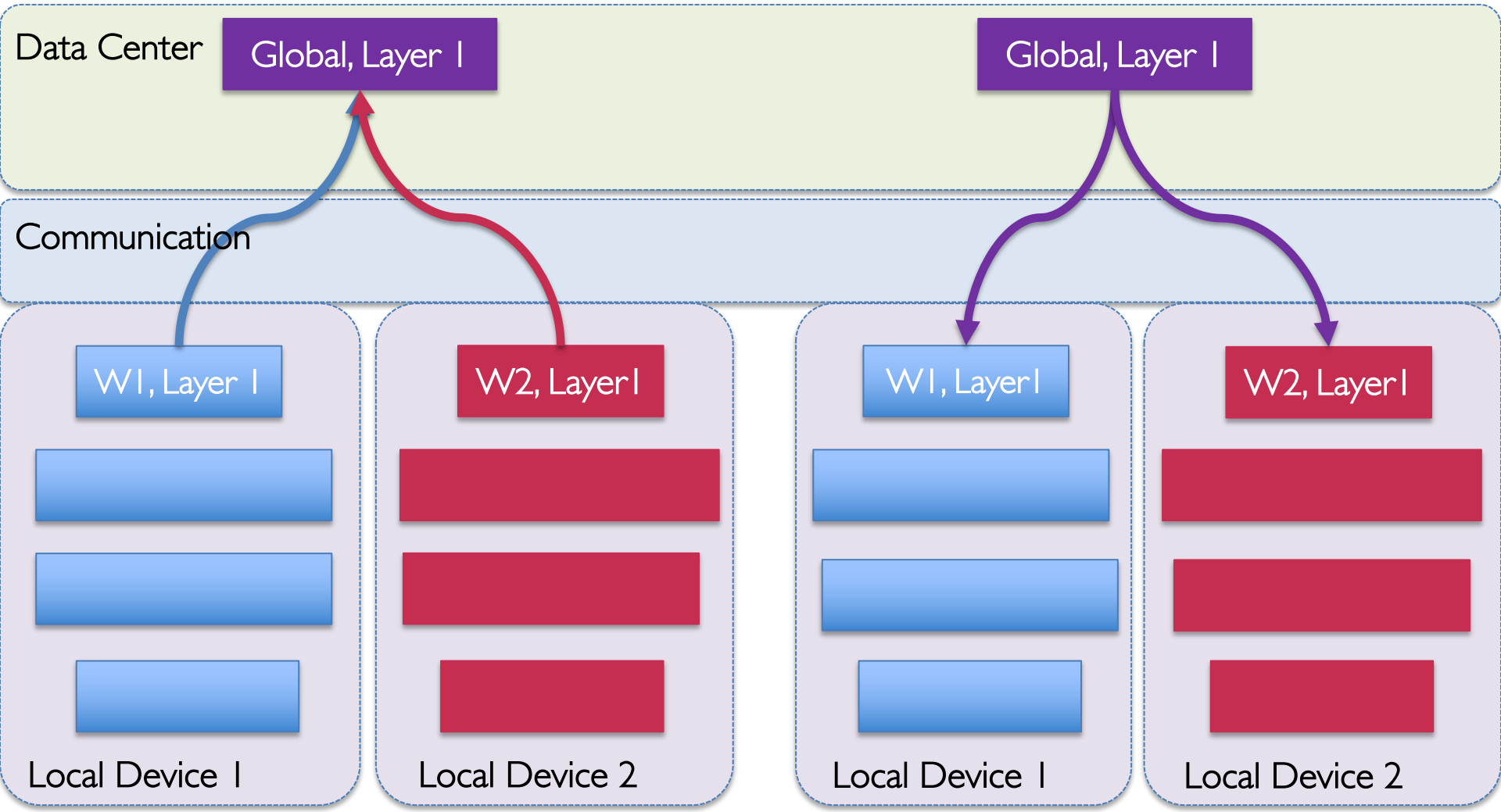
An Example on CNN:

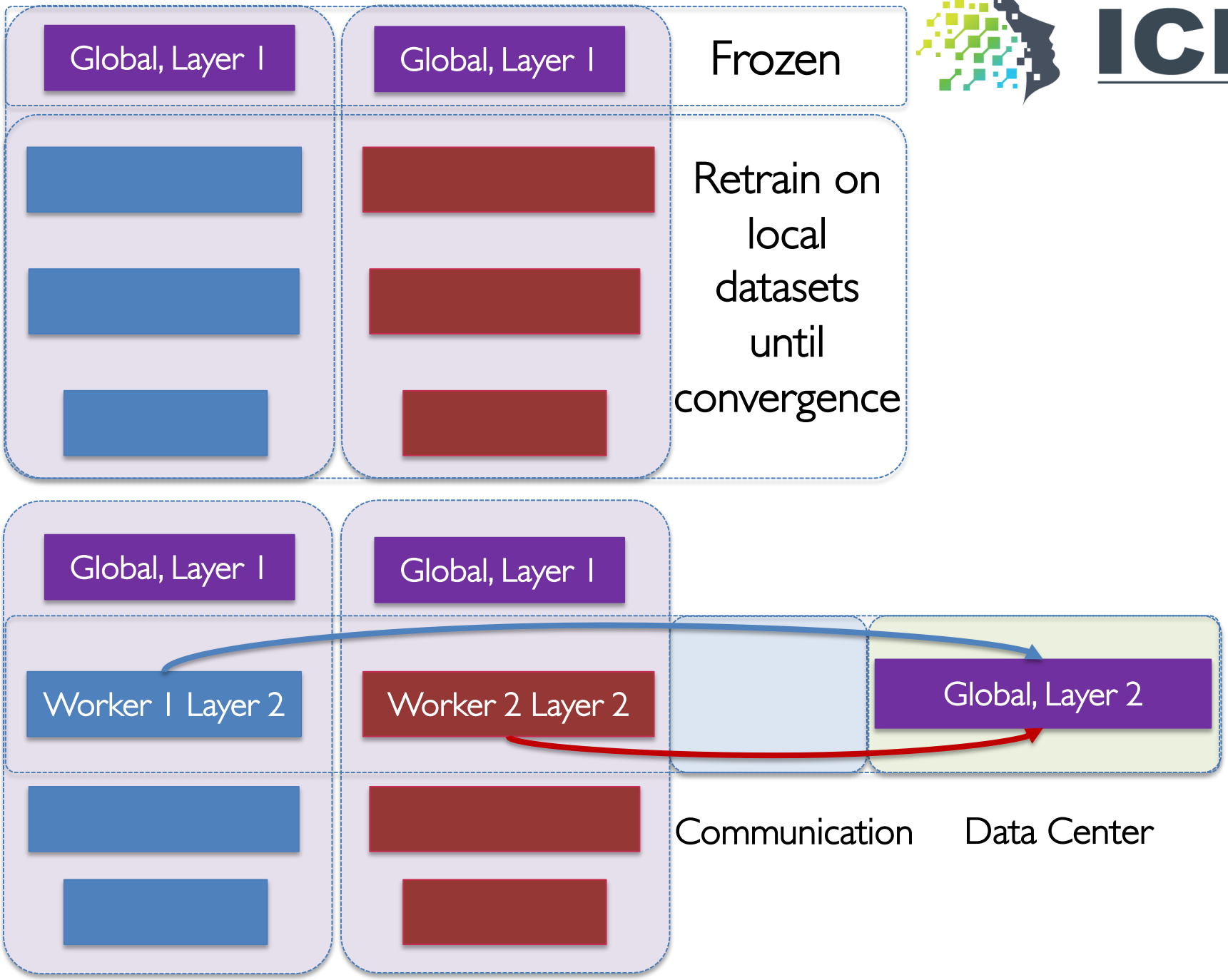


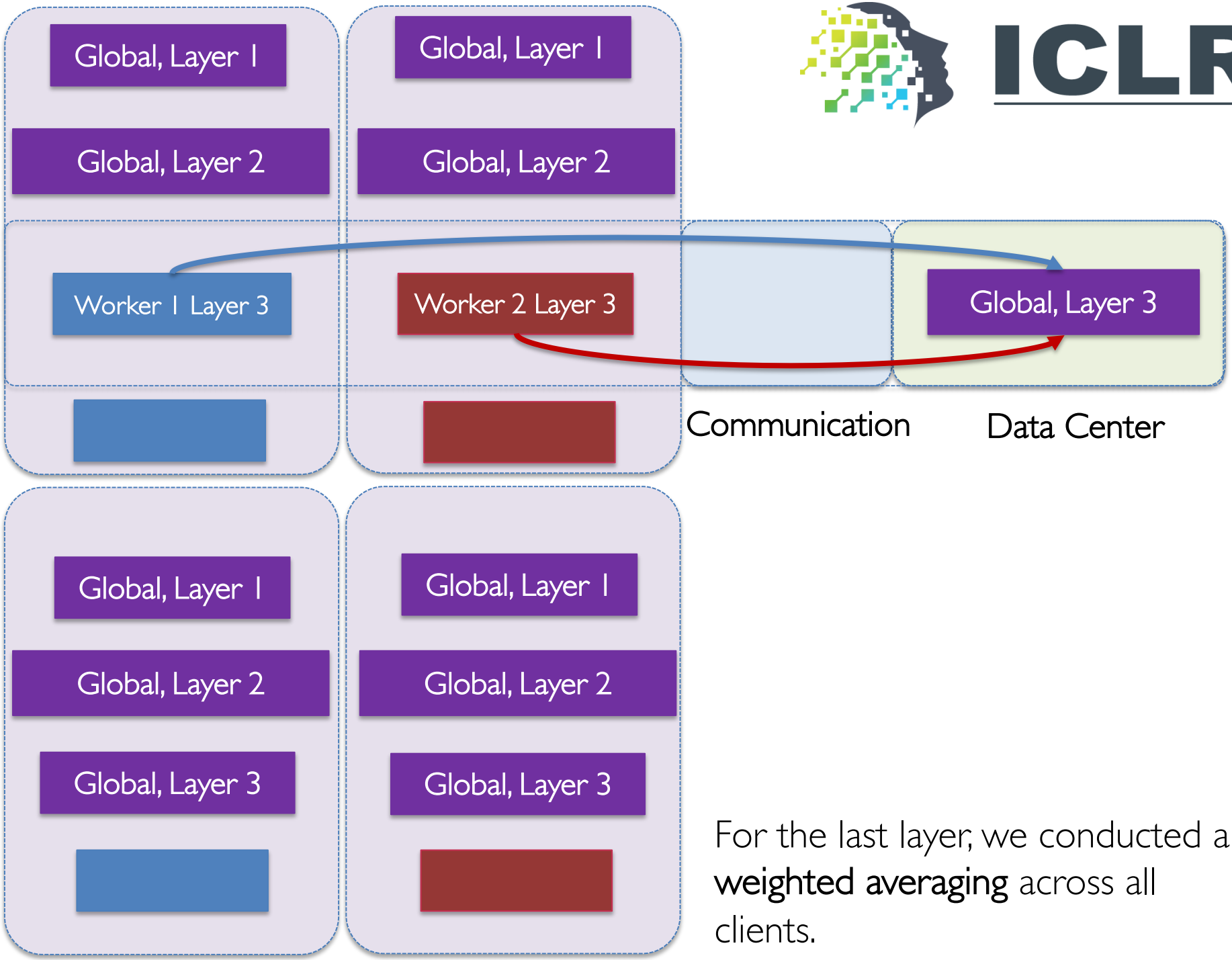
First Result on One-shot Matching



FedMA: Layer-wise matched averaging

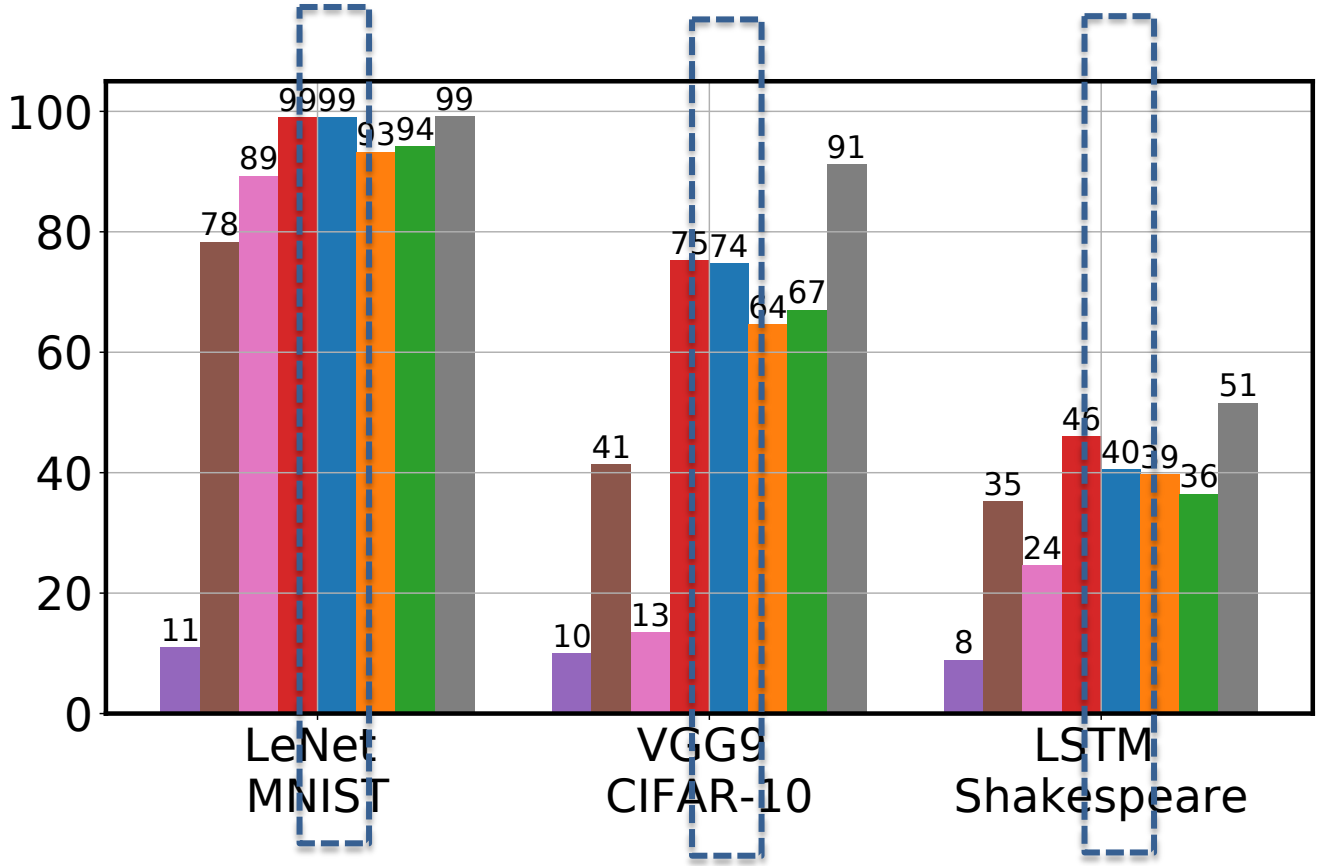


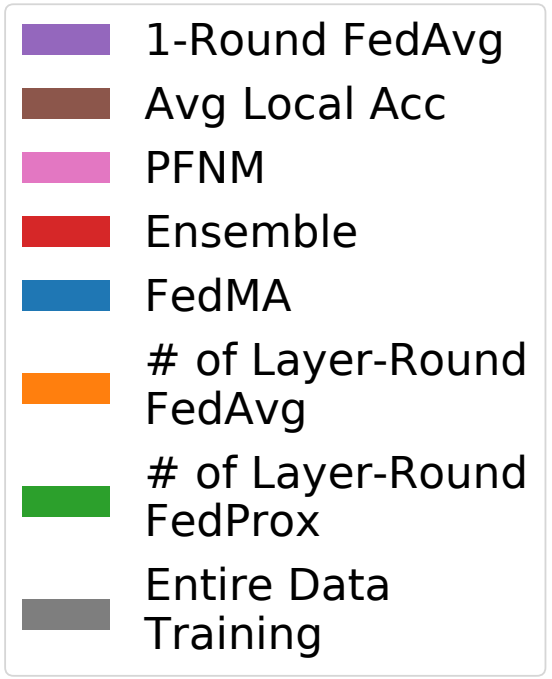

Global, Layer 1
Global, Layer 1
Frozen
Retrain on local datasets until convergence
Global, Layer 1
Global, Layer 1
Worker 1 Layer 2
Worker 2 Layer 2
Global, Layer 2
Communication
Data Center



For the last layer, we conducted a **weighted averaging** across all clients.

First Result on One-shot Matching

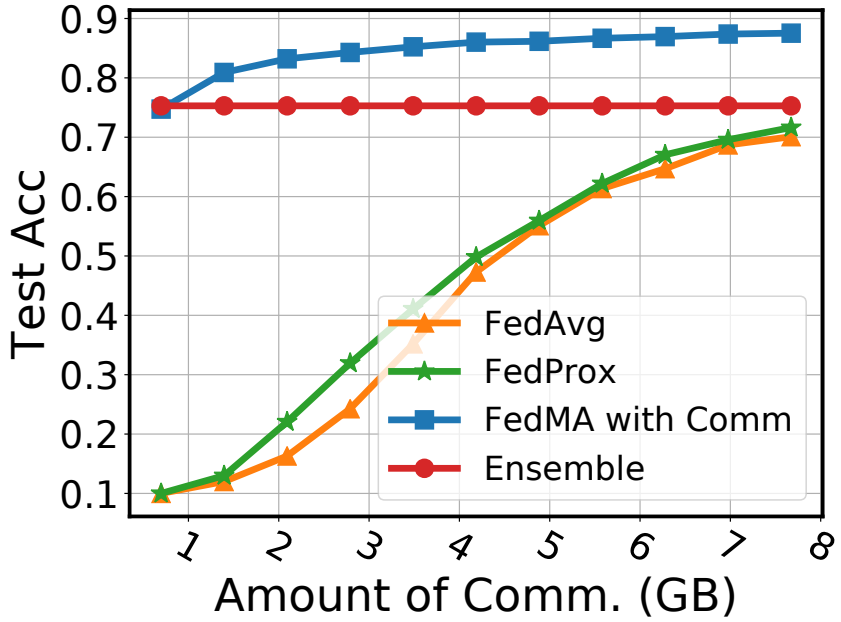


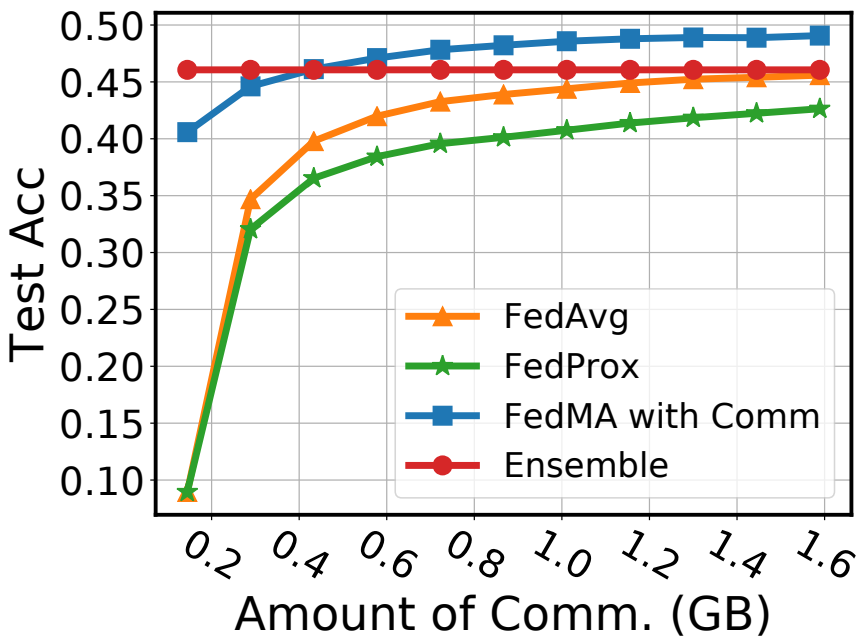


FedMA: matching with comm.

ICLR

- After the first round of layer-wise matching, we broadcast the matched model back to local workers.
- We slice the matched global model to recover the original local model size.
- We then repeat the layer-wise matching process.





VGG-9 trained on CIFAR-10

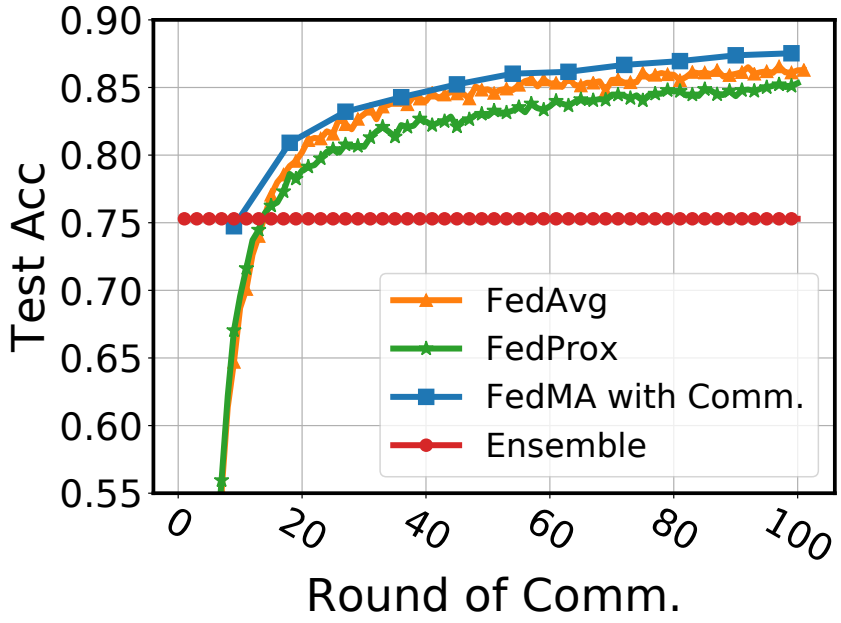
LSTM trained on Shakespeare

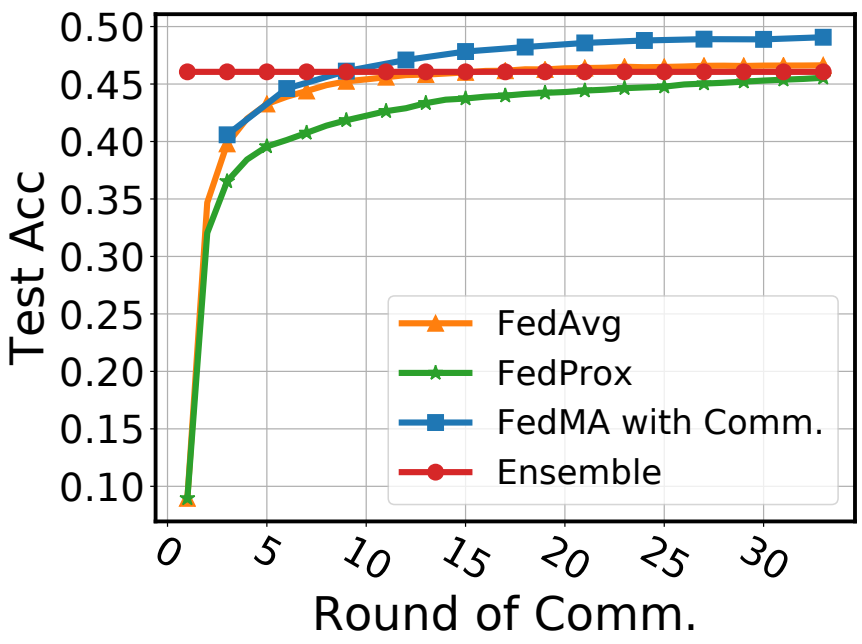
16 local workers; heterogeneous setup; convergence w.r.t. message size

FedMA matching with comm.

ICLR

- After the first round of layer-wise matching, we broadcast the matched model back to local workers.
- We slice the matched global model to recover the original local model size.
- We then repeat the layer-wise matching process.





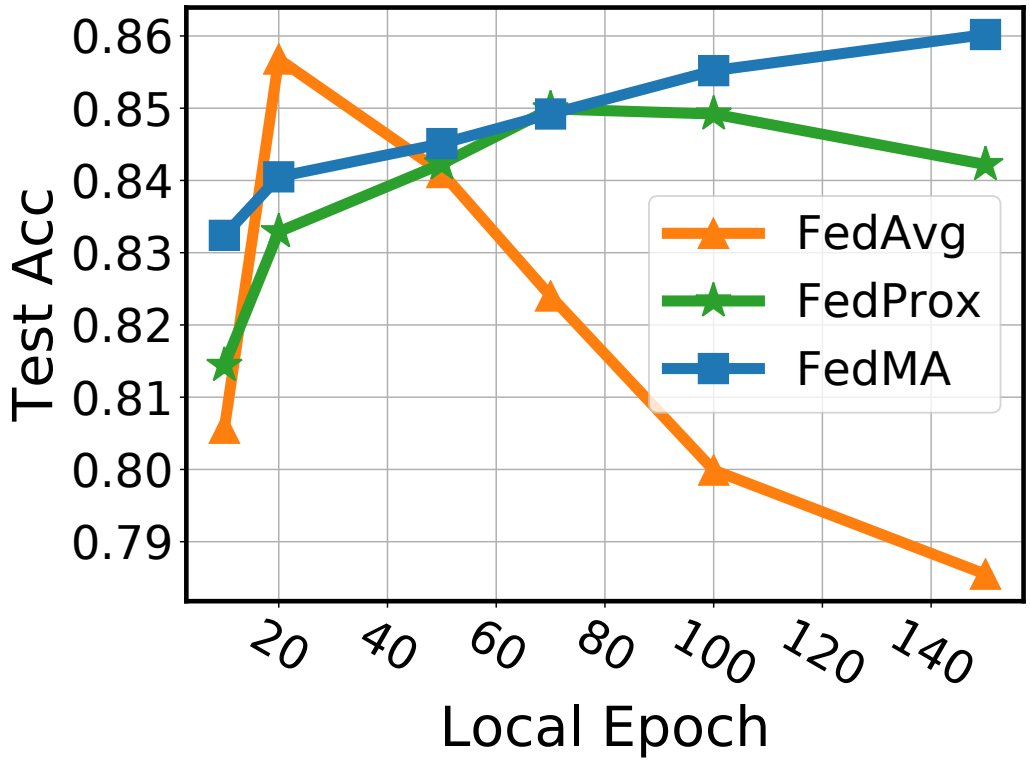
VGG-9 trained on CIFAR-10

LSTM trained on Shakespeare

16 local workers; heterogeneous setup; convergence w.r.t. communication round

Effect of local training period

- Local training epochs (denoted by *E*) is a hyper-parameter shared among FedMA, FedAvg, and FedProx.
- It has been studied that a "too-long" local training period leads to bad global model in FedAvg.
- We study the effect of *E* on FedMA.

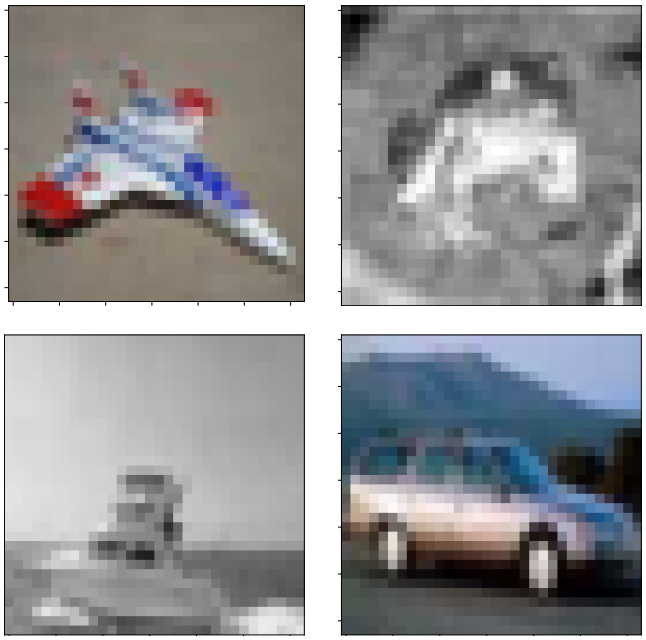


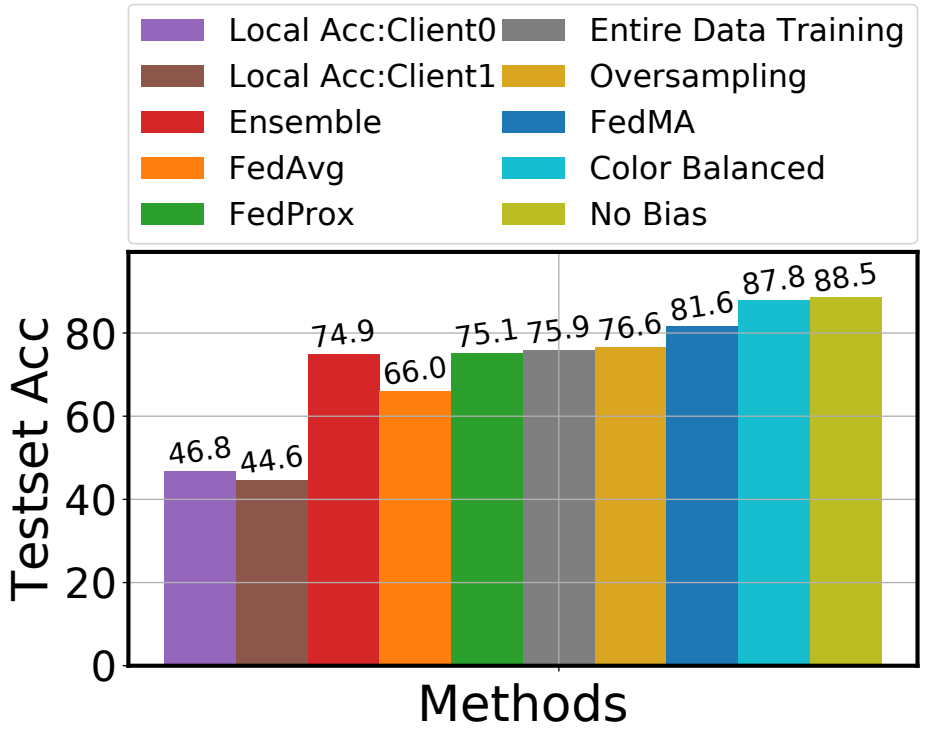
16 local clients;
heterogeneous data partition

Handling Data Bias

ICLR

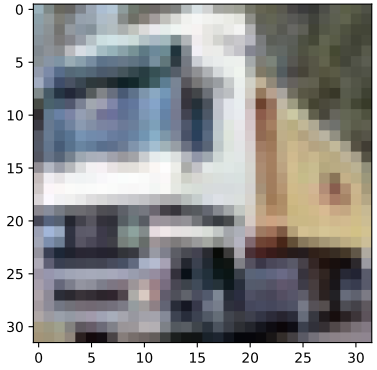
- Randomly sample **5 classes** where **95%** images are **grayscale** and the rest **5%** are **color** in the training set.
- In the test set, the fraction of grayscale and color images are **balanced**.
- We split all grayscale images to one client and all color image to the other client.



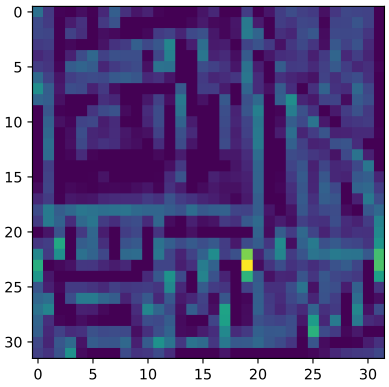


Inspired by "identifying and understanding deep learning phenomena" workshop on ICML19.

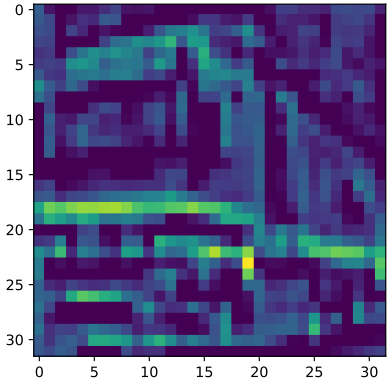
Explainability:



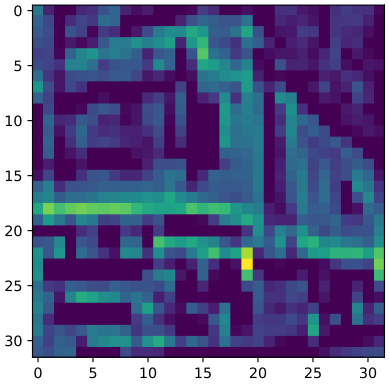
Example: Datapoint #2 of CIFAR-10



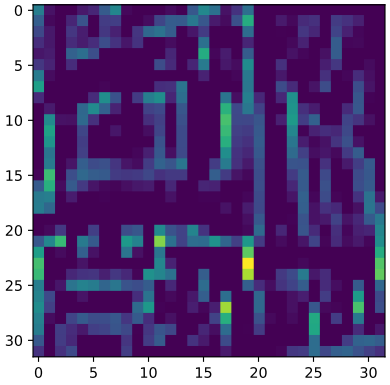
Filter 31 of Client 1



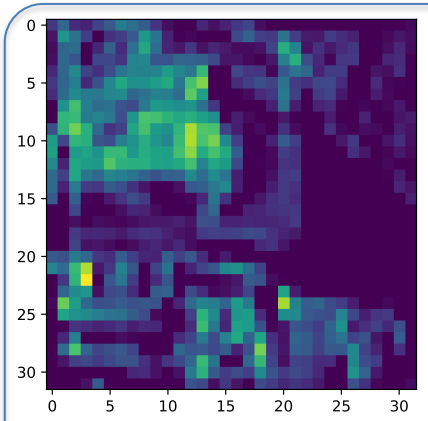
Filter 17 of Client 2



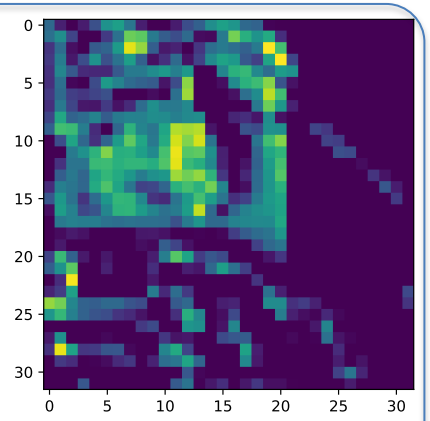
Filter 29 of Global Model



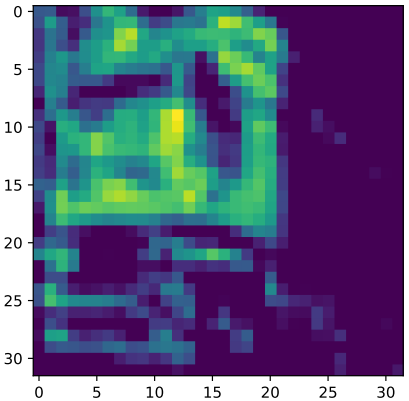
Averaged filter 29



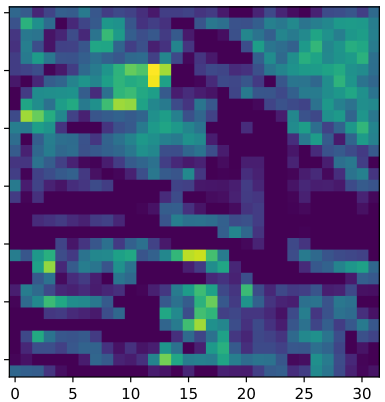
Filter 0 of Client1



Filter 23 of Client2



Filter 0 of Global Model

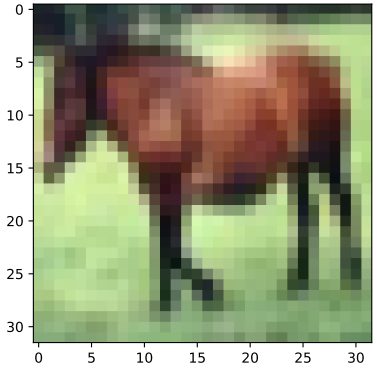


Averaged filter 0

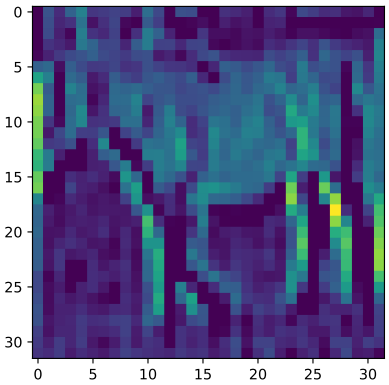
Since plotting the **conv filters** is not easy, we visualize the **representations** coming out of the matched filters

ICLR

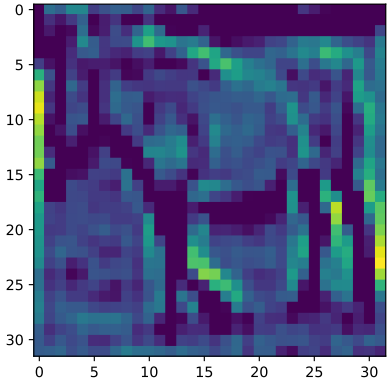
Matching outcome:



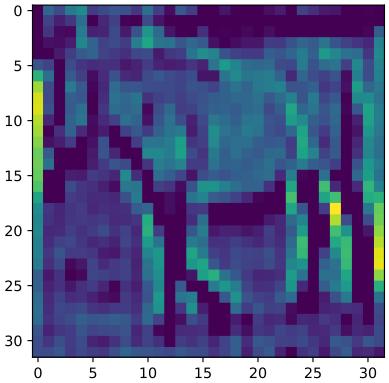
Example: Datapoint #8 of CIFAR-10



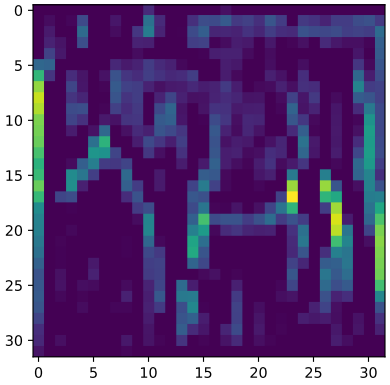
Filter 31 of Client 1



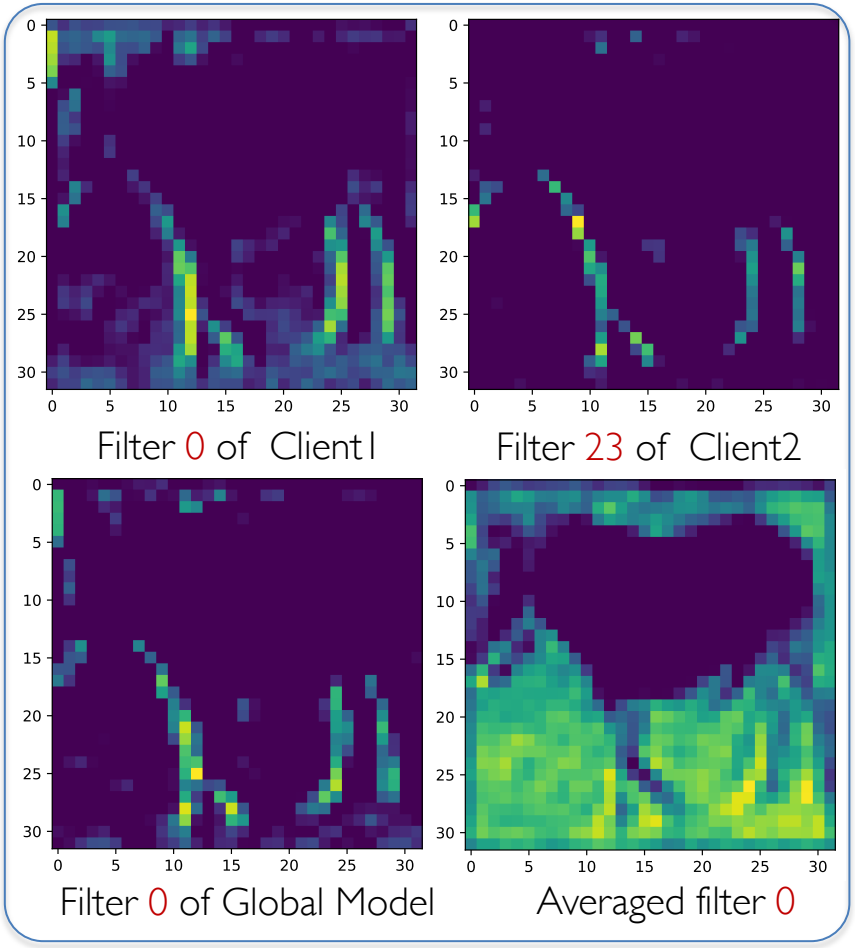
Filter 17 of Client 2



Filter 29 of Global Model



Averaged filter 29



Filter 0 of Client1

Filter 23 of Client2

Filter 0 of Global Model

Averaged filter 0

Averaging the conv filters can break the "feature maps".

ICLR

ICLR

Code available at:

https://github.com/IBM/FedMA

Thank you!