Hermes: An Efficient Federated Learning Framework for Heterogeneous Mobile Clients

Ang Li[1], Jingwei Sun[1], Pengcheng Li[2*], Yu Pu[2], Hai Li[1], Yiran Chen[1]
[1]Department of Electrical and Computer Engineering, Duke University
[2]Alibaba DAMO Academy
[1]{ang.li630, jingwei.sun, hai.li, yiran.chen}@duke.edu, [2]{landy0220@gmail.com, y.pu@@alibaba-inc.com}

Presented by Tian Liu
Oct 25, 2021

Outline

Federated Learning

State-of-the-Art Solutions

Hermes: less communication cost, less memory footprint, better accuracy

Evaluations

Conclusions

Federated Learning

- Distributed ML where multiple devices collaboratively learn a shared global model
- Addressing limited data in local mobile devices
- Privacy preservation
- **Large communication cost**
- **Data heterogeneity**
  - **Global model bad generalization**

Figure 1: An example of federated learning for the task of next-word prediction on mobile phones. Devices communicate with a central server periodically to learn a global model. Federated learning helps preserve user privacy and reduce strain on the network by keeping data localized.

State-of-the-Art Solutions

- Communication cost
  - Compress the communication
    - Quantization: reduce bit number of data
    - Sparsification: only transmit subset of data
    - Hybrid
- Data heterogeneity
  - Fine-tuning, Multi-task learning, Contextualization
  - 2-steps: global collaborative learning + local fine-tuning
  - Incurs extra computational costs
- Very few address both simultaneously
  - LG-FedAvg
    - 2 parts + 2 steps: global and local parameters, normal FedAvg + global params only
    - Inadequate reduction, suboptimal splitting, evaluated in unrealistic way
  - HeteroFL
    - Adaptively allocates submodels to each devices based on computation capability
    - Not based on local data
  - SFSL
    - Identify submodel based on local history data, transmit only submodel
    - Designed for recommender system
- None of existing approaches consider inference accuracy

Hermes: Efficient FL

- Reduce communication cost
- Improve computation efficiency for inference
- Learn a personalized model for each device

Solutions:

- Each device learn a pruned subnetwork
- Only transmit subnetwork parameters

Challenges:

- How to learn the subnetwork?
- How to aggregate local updates?

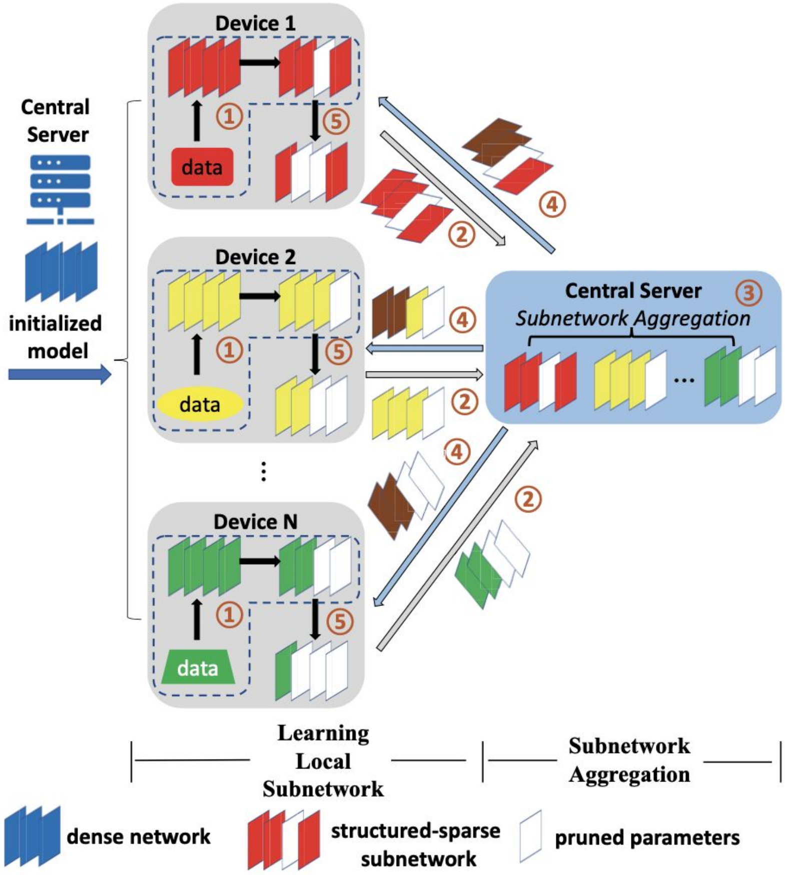

Figure 3: The overview of the Hermes framework.

How to learn the subnetwork?

- Unstructured pruning
  - Parameter wise, high flexibility but requires hardware support for computation efficiency
- **Structured pruning**
  - Channel-wise or filter-wise, less flexible but hardware-friendly
- Adds sparsity regulation in objective function

$$F(\mathbf{W}) = F_D(\mathbf{W}) + \lambda R(\mathbf{W})$$

$$R(\mathbf{W}) = R_{conv}(\mathbf{W}) + R_{fc}(\mathbf{W}),$$

$$R_{conv}(\mathbf{W}) = \sum_{l=1}^{W_{conv}} \left(\sum_{f_l=1}^{F_l} \|\mathbf{W}^{(l)}_{f_l,:,:,:}\|_g + \sum_{ch_l=1}^{Ch_l} \|\mathbf{W}^{(l)}_{:,ch_l,:,:}\|_g \right),$$

$$R_{fc}(\mathbf{W}) = \sum_{l=1}^{W_{fc}} \left(\sum_{row_l=1}^{Row_l} \|\mathbf{W}^{(l)}_{row_l,:}\|_g + \sum_{col_l=1}^{Col_l} \|\mathbf{W}^{(l)}_{:,col_l}\|_g \right),$$

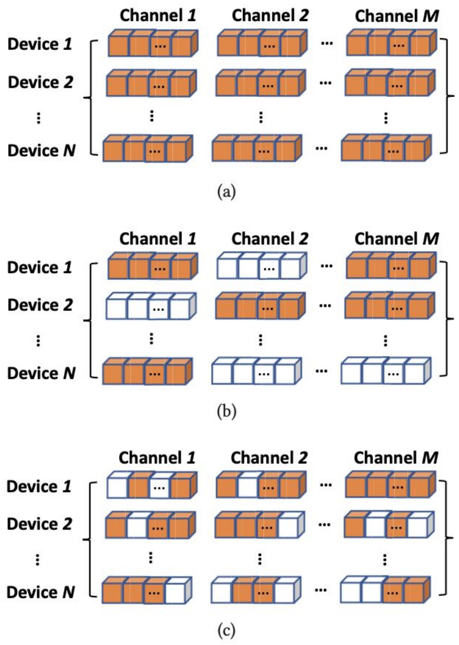

Figure 4: Illustration of the difference between the structured pruning and unstructured pruning. (a) The parameter matrix representations of channels in a layer. (b) The effect of the channel-wise structured pruning, where the white and orange channels are the pruned ones and retained ones, respectively. (c) The unstructured pruning scheme.

How to aggregate local updates?

Algorithm 1: Training Algorithm of Hermes.

Data: $(D_1, \ldots, D_N)$ where D_k is the local data on kth device

Server Executes:

initialize the global model $\boldsymbol{W}$

for *each round* $T = 1, 2, \ldots$ **do**

  $k \leftarrow \max(N \times K, 1)$ `/* N available devices, random sampling rate K */`

  $S_c \leftarrow \{C_1, \ldots, C_k\}$ `/* the selected k participating devices indexed by k */`

  for $C_k \in S_t$ ***in parallel*** **do**

    $\boldsymbol{W}_k^T = \boldsymbol{W}^T \odot \boldsymbol{M}_k^T$ `/* the subnetwork of ` C_k ` */`

    $\boldsymbol{W}_k^{T+1} \leftarrow \text{ClientUpdate}(C_k, \boldsymbol{W}_k^T)$

  $\boldsymbol{W}^{T+1} \leftarrow$ (aggregate $\{\boldsymbol{W}_k^{T+1}\}$) `/* using the proposed personalization-preserving aggregation */`

ClientUpdate $(C_k, \boldsymbol{W}_k^T)$**:**

acc $\leftarrow$ (evaluate $\boldsymbol{W}_k^T$ on the local validation data D_k^{val})

if *acc* $>$ *acc*$_{threshold}$ *and* $r_k^T < r_{target}$ **then** `/* ` r_k^T ` is the current pruning rate of kth client's model, ` r_{target} ` is the target pruning rate */`

  $\boldsymbol{M}_k^{T+1} \leftarrow$ (prune $\boldsymbol{W}_k^T$ with the fixed pruning rate r_p)

$\mathcal{B} \leftarrow$ (split local data D_k^{train} into batches);

for *each local epoch i from* 1 *to E* **do**

  for *batch* $b \in \mathcal{B}$ **do**

    $\boldsymbol{W}_k^{T+1} \leftarrow \boldsymbol{W}_k^T \odot \boldsymbol{M}_k^{T+1} - \eta \nabla_{F_k}(\boldsymbol{W}_k^T \odot \boldsymbol{M}_k^{T+1}; b)$ `/* ` η ` is the learning rate, ` $F_k(\cdot)$ ` is the loss function */`

return $\boldsymbol{W}_k^{T+1}$, $\boldsymbol{M}_k^{T+1}$ to server

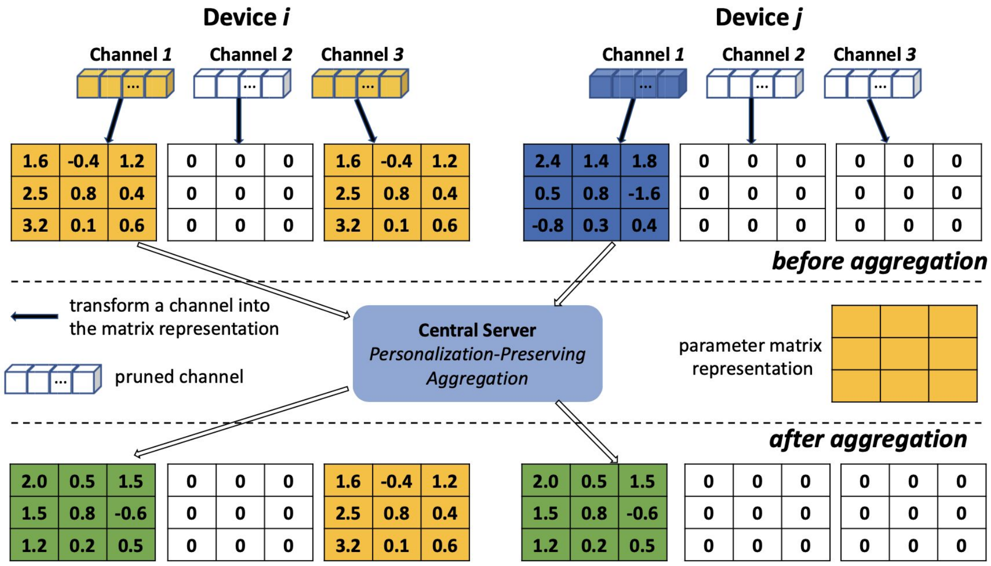

Figure 5: Illustration of the personalization-preserving aggregation on the central server.

Evaluation Scheme

- 2 mobile AI applications
  - Image Classification: VGG16 model, EMNIST for training, CIFAR10 for validation
    - EMNIST: sample 2414 writer's data
  - Human Activity Recognition: 3-layer fully connected neural network
    - HAR: accelerometers and gyroscope data from 30-individuals
- Implemented on Google Pixel 3, Android 9.0, PyTorch 1.5
- Central server Intel Xeon E5-2630@2.6GHz and 128G RAM
- Baselines: Standalone, FedAvg, Top-K, Per-FedAvg, LG-FedAvg

Table 1: Statistical information of datasets.

Dataset	Number of devices	Classes	Non-IID
EMNIST [8]	2414	64	✓
CIFAR10 [26]	400	10	✓
HAR [3]	30	6	✓

Evaluation Metrics

- Training performance
  - Convergence Speed
  - Inference accuracy
  - Communication cost: RTT
- Robustness
  - # of devices
  - Data unbalanceness
  - Target prune rate
- Runtime performance
  - Memory footprint
  - Inference latency
  - Energy consumption

Faster Convergence

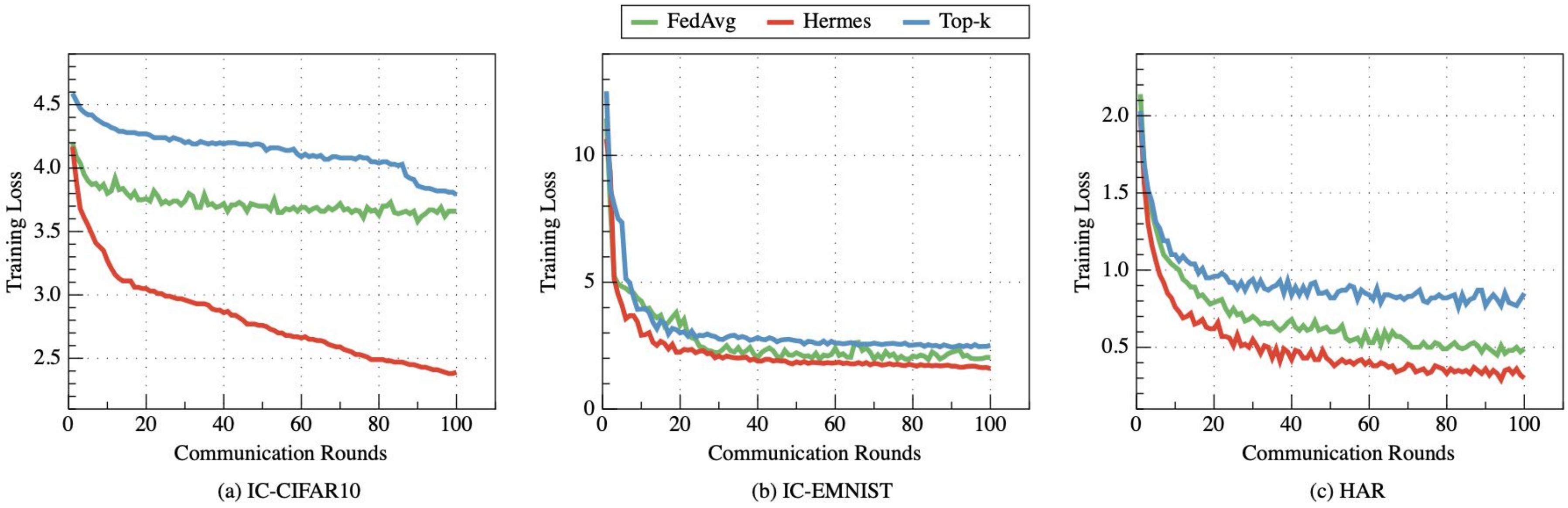

Figure 6: Comparison of convergence speed between FedAvg and Hermes.

Higher Inference Accuracy, Less Communication Cost

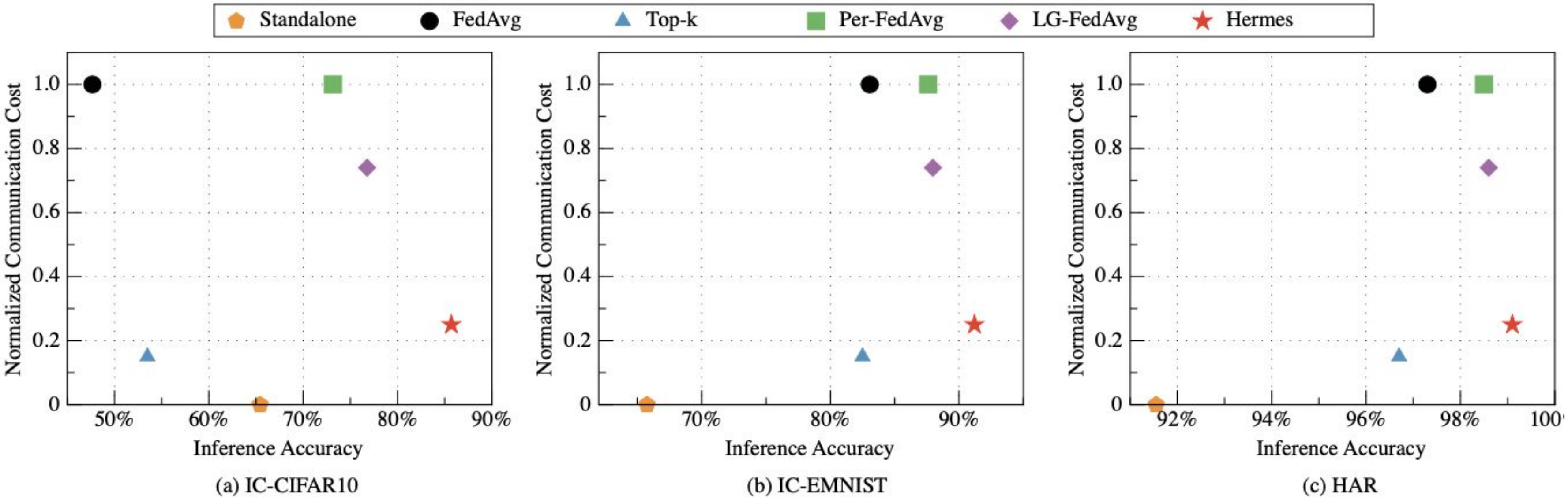

Figure 7: Comparison between Hermes and baselines in inference accuracy-communication cost space.

Robust to # of Devices and Unbalanced Data

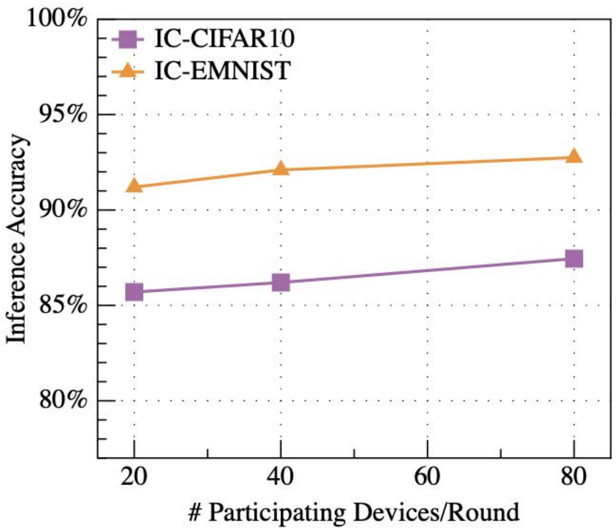

Figure 8: The impact of the number of participating devices on Hermes performance.

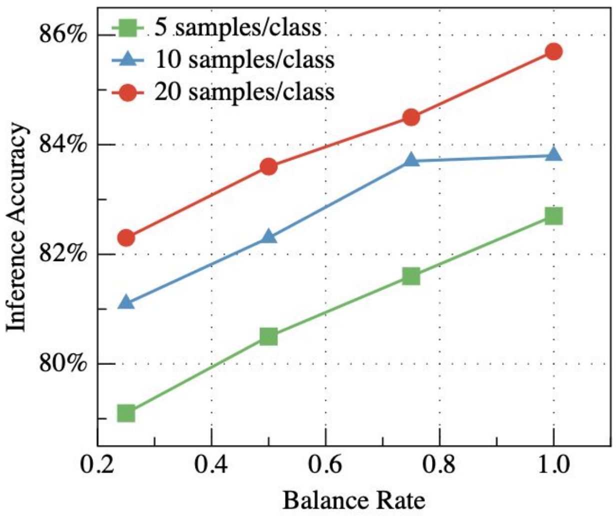

Figure 9: The impact of data volume and balance rate on Hermes performance (IC-CIFAR10).

Robust to Hash Pruning

Table 2: The impact of target pruning rate on Hermes performance.

Application	r_{target}	Accuracy	Communication Cost
IC-CIFAR10	0.3	85.72%	0.54
	0.5	85.91%	0.79
	0.8	86.35%	1
IC-EMNIST	0.3	91.24%	0.53
	0.5	91.35%	0.75
	0.8	91.66%	1

Less Memory Footprint, Inferency Latency, Energy Consumption

Table 3: Memory footprint reduction of Hermes. ($r_{target} = 0.3$)

Application	Hermes Model Size (MB)	Baseline Model Size (MB)
IC-CIFAR10	161.16	537.21
IC-EMNIST	161.43	538.09
HAR	1.32	4.41
All Included	323.91	1081.24

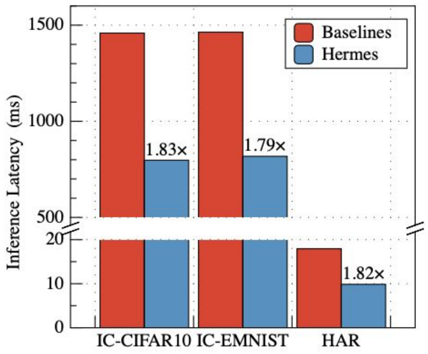

Figure 10: Comparison between Hermes and the baselines on inference time. ($r_{target} = 0.3$)

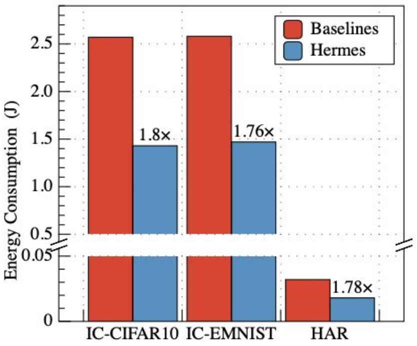

Figure 11: Comparison between Hermes and the baselines on energy consumption. ($r_{target} = 0.3$)

Conclusions

- Problems with Federated Learning
  - Communication cost
  - Data heterogeneity
- Hermes: efficient FL framework
  - Pruned local subnetwork
  - Transmit only subnetwork params
- Faster Convergence, Higher Inference Accuracy, More Robust
- Less Communication Cost, Less Inference Latency, Less Energy Consumption
- Believed to be a significant step towards the realization of efficient FL in heterogeneous mobile devices