Hwamei: A Learning-based Synchronization Scheme for Hierarchical Federated Learning

Tianyu Qi, Yufeng Zhan, Peng Li, Jingcai Guo, and Yuanqing Xia

Tianyu Qi
qitianyu@bit.edu.cn

School of Automation
Beijing Institute of Technology

July 21, 2023

Outline

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

Introduction

Development of Edge Computing:

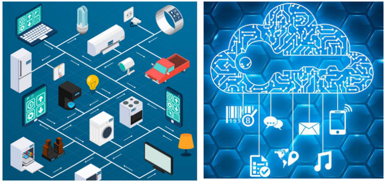


Figure: Edge & Cloud Computing

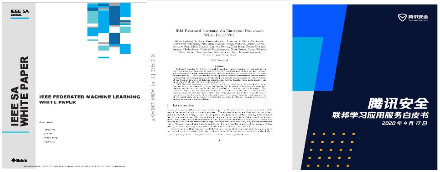


Figure: Privacy security policy

- Mobile devices continue to generate vast amounts of data.
- The independent storage of data on devices presents challenges for centralized learning.
- There is a growing global emphasis on privacy and security concerns.

BEIJING INSTITUTE OF TECHNOLOGY

FedAvg:

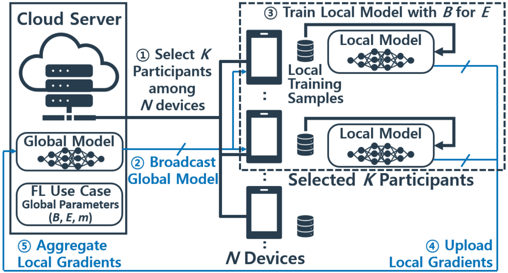


Figure: The design of FedAvg

Training of FedAvg:

- Local training: The model is deployed on devices, utilizing locally available data.
- Cloud aggregation: The model is aggregated on the cloud server.
- Limitation: Frequent model transmissions result in high communication overhead.

Hierarchical Federated Learning:

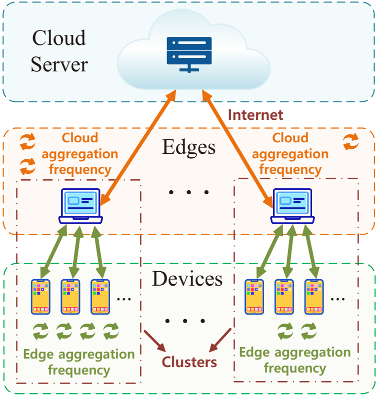


Figure: A synchronization scheme based on HFL

Advantage:

1. Large scale: Reduce the communication overhead.
2. Low convergence bound.

Challenge:

1. Heterogeneity.(System, Data, Communication)
2. High energy consumption.
3. The aggregation frequency is difficult to determine under 2-layers framework.

Outline

1. Introduction
2. **Background and Motivation**
3. The Design of Hwamei
4. Evaluation
5. Conclusion

Background and Motivation

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Hierarchical Federated Learning:

- Local SGD (Device i):

$$f(w_i) = \frac{1}{|\mathcal{D}_i|} \sum_{(x,y)\in\mathcal{D}_i} f(w_i, x, y)$$

- Edge aggregation (Edge j):

$$w_j^e = \sum_{i=1}^{N_j} \frac{|\mathcal{D}_i| w_i}{\sum_{i=1}^{N_j} |\mathcal{D}_i|}$$

- Cloud aggregation:

$$w = \sum_{j=1}^{M} \frac{|\mathcal{D}_j| w_j^e}{\sum_{j=1}^{M} |\mathcal{D}_j|}$$

Proximal Policy Optimization:

- PPO algorithm's objective function:

$$J^{\theta'}(\theta) \approx \sum_{(\boldsymbol{s}_t,\boldsymbol{a}_t)} \min\left(\frac{p_\theta(\boldsymbol{a}_t \mid \boldsymbol{s}_t)}{p_{\theta'}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)} A^{\theta'}(\boldsymbol{s}_t, \boldsymbol{a}_t), \right.$$
$$\left. \operatorname{clip}\left(\frac{p_\theta(\boldsymbol{a}_t \mid \boldsymbol{s}_t)}{p_{\theta'}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)}, 1-\varepsilon, 1+\varepsilon\right) A^{\theta'}(\boldsymbol{s}_t, \boldsymbol{a}_t)\right)$$

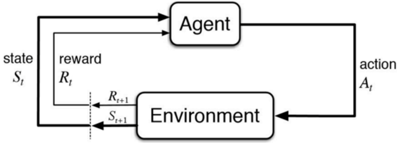


Figure: Framework of RL

Background and Motivation

System dynamics:

1. Training performance from different devices with different co-running tasks. (Raspberry PI & stress-ng)
2. The communication with local (China) and overseas (USA) edges to the same cloud. (Data from Alibaba Cloud)

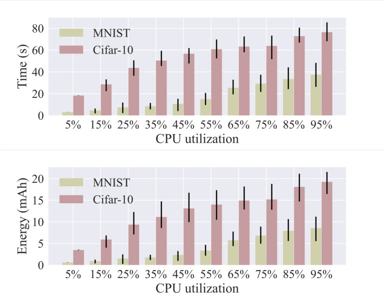


Figure: Training time and energy of Raspberry Pi

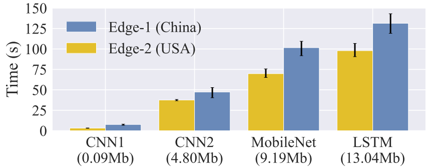


Figure: Edge-to-cloud communication time in different regions

How synchronization scheme affects HFL:

- System setting: an Alibaba cloud server, 5 laptops as edges, and 50 Raspberry Pi as devices.
- *Var-Freq A*: Cluster devices under the edge by training speed. Increase the edge and cloud aggregation frequency of slower clusters.
- *Var-Freq B*: Based on *Var-Freq A*, reduce the aggregation frequency of fast devices with high energy consumption.

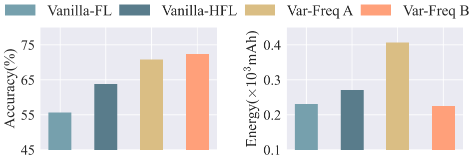


Figure: Accuracy and energy within different frameworks of MNIST

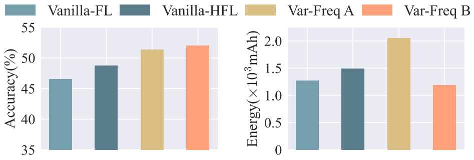


Figure: Accuracy and energy within different frameworks of Cifar-10

Background and Motivation

Observation1

- The time and energy consumption during FL training is dynamic.
- The edge-to-cloud communication time varies from one region to another.

Observation2

- Changing the aggregation frequency of each edge and device after clustering can improve the training performance.
- A reasonable aggregation frequency can maximize model accuracy and energy efficiency.

How to find the right frequency in dynamic and heterogeneous systems?

Outline

The Desgin of Hwamei

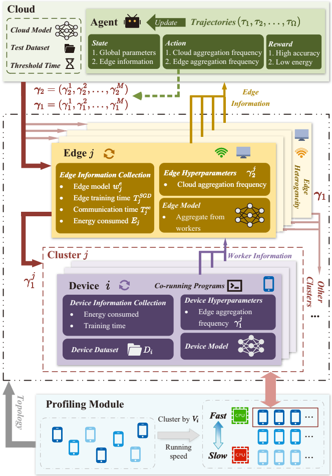


Figure: Overview of Hwamei

Overview:

- Cluster the devices by profiling module.
- Agent deployed on the cloud collects information from edges.
- Agent assigns aggregation frequency to edges and devices.

Profiling module:

- All devices run the profiling task.
- Devices get $V_i = [T_i^{pro}\ E_i^{pro}],\ i \in \{1, 2, \ldots, N\}$.
- Cluster the devices by V_i using *k-Means* algorithm.

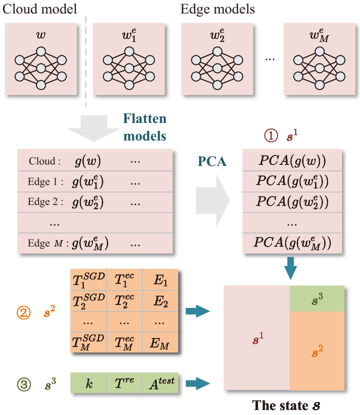


Figure: Composition of the state

State:

- Model parameters:

$$\boldsymbol{s}^1 = PCA\{[g(w)^T \; g(w_1^e)^T \; g(w_2^e)^T \; \ldots \; g(w_M^e)^T]^T\}$$

- Time and energy consumption:

$$\boldsymbol{h}_j = [T_j^{SGD} \; T_j^{ec} \; E_j]^T$$

$$\boldsymbol{s}^2 = [\boldsymbol{h}_1 \; \boldsymbol{h}_2 \; \ldots \; \boldsymbol{h}_M]^T$$

- Global information:

$$\boldsymbol{s}^3 = [k \; T^{re} \; A^{test}]$$

- Splice:

$$\boldsymbol{s} = cat\{(\boldsymbol{s}_1, cat\{(\boldsymbol{s}_3, \boldsymbol{s}_2), dim = 0\}), dim = 1\}$$

The Desgin of Hwamei

北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

Action:

- The aggregation frequency $\boldsymbol{\gamma}_1 = \{\gamma_1^1, \gamma_1^2, \ldots, \gamma_1^M\}$ and $\boldsymbol{\gamma}_2 = \{\gamma_2^1, \gamma_2^2, \ldots, \gamma_2^M\}$.

Reward:

- The reward after the k-th cloud communication:

$$r_k = A^{test}(k) - A^{test}(k-1) - \epsilon E(k)$$

Workflows:

1. Initialize the parameters.
2. Train HFL for several rounds and train PCA modules.
3. The agent makes decision and push $(\boldsymbol{s}_k, \boldsymbol{a}_k, r_k, \boldsymbol{s}_{k+1})$ to memory.
4. Repeat step 3 until $T^{re} < 0$.
5. Update the agent and clean the memory.

Algorithm 1 *Arena*'s Training Process

Initialize the topology by profiling module;
Initialize threshold time T, remaining time $T^{re} = T$, global model $w(0)$, round of cloud aggregations $k = 0$;
Train once cloud aggregation by given aggregation frequencies, get $w(1)$, $w_j^e(1)$, and record T^{use};
Train PCA module by $w(1)$ and $w_j^e(1)$;
Update $T_{init}^{re} = T^{re} - T^{use}$; k++;
for 1 **to** Ω **do**
  while true do
    Observe state $\boldsymbol{s}_k$;
    Choose actions $\boldsymbol{a}_k$, that is $\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_2$;
    Train HFL by $\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_2$, record T^{use};
    Get reward r_k and $\boldsymbol{s}_{k+1}$, update $T^{re} = T^{re} - T^{use}$;
    Push $(\boldsymbol{s}_k, \boldsymbol{a}_k, r_k, \boldsymbol{s}_{k+1})$ to agent memory; k++;
    if $T^{re} < 0$ **then**
      Set $k = 1$, $T^{re} = T_{init}^{re}$;
      break
    end if
  end while
  Update the agent and clear agent memory;
end for

Figure: Training process of Hwamei

Outline

Evaluation

Settings:

- Testbed: Raspberry Pi, Laptop, Alibaba Cloud.
- Dataset: MNIST, Cifar-10 with CNN.
- Benchmarks: *Vanilla-FL*, *Vanilla-HFL*, *Favor*, *Share*, *Hwamei*.
- Heterogeneity:
  1. System: 5 categories CPU utilization from 10% to 80%.
  2. Edge communication: Sampling from Real edge communication time.
  3. Data distribution: Label non-IID.

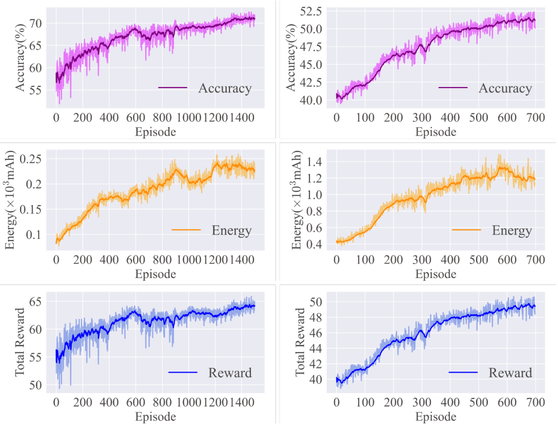
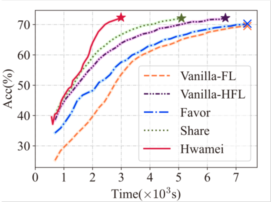


Figure: Training the DRL agent of Hwamei.

Evaluation

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

Training performance:



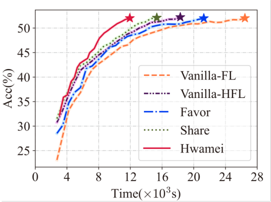


Figure: Accuracy of testing MNIST & Cifar-10

Impact of profiling module:

TABLE I
PERFORMANCE OF CLUSTER VS NON-CLUSTER ON HWAMEI

	Time	Cluster		non-Cluster	
		Accuracy	**Energy**	**Accuracy**	**Energy**
MNIST	2100s	63.0%	114mAh	61.7%	126mAh
	2400s	67.6%	154mAh	65.2%	172mAh
	2700s	69.4%	180mAh	67.9%	212mAh
	3000s	72.3%	226mAh	70.8%	253mAh
Cifar-10	7500s	45.2%	548mAh	44.1%	619mAh
	9000s	49.0%	704mAh	47.8%	843mAh
	10500s	50.3%	957mAh	49.1%	1124mAh
	12000s	52.1%	1190mAh	50.7%	1358mAh

Result

- Hwamei saves 51.1% and 34.7% time in average.
- The profiling module enables the system to fully use device resources.

Evaluation

Training with different threshold time:

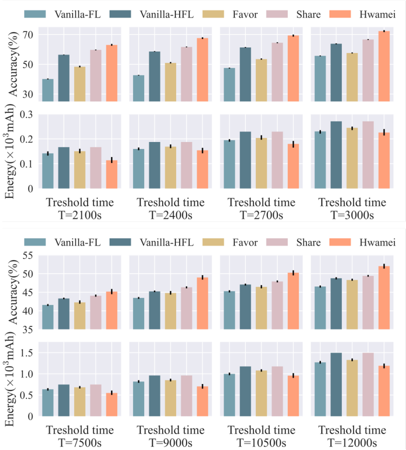


Figure: Accuracy and energy within different times

Training with different non-IID levels:

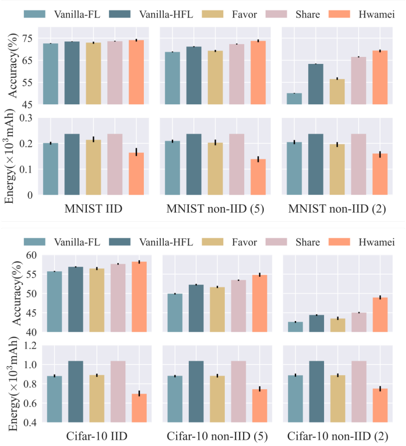


Figure: Accuracy and energy within different non-IID

Outline

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

Conclusion

Conclusion

1. We propose an intelligent HFL synchronization scheme based on DRL, which can co-optimize the model performance and training efficiency.
2. We develop an HFL testbed with Raspberry Pi and Alibaba Cloud and collect the real-world data.
3. We conduct extensive experiments comparing with the state-of-the-arts.

Thank you for your attention!

Questions?