Tomtit: Hierarchical Federated Fine-Tuning of Giant Models based on Autonomous Synchronization

Tianyu Qi, Yufeng Zhan, Peng Li and Yuanqing Xia

Tianyu Qi
qitianyuqty@163.com

School of Automation
Beijing Institute of Technology

23rd May, 2024

Outline

Introduction

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

Development of large models:

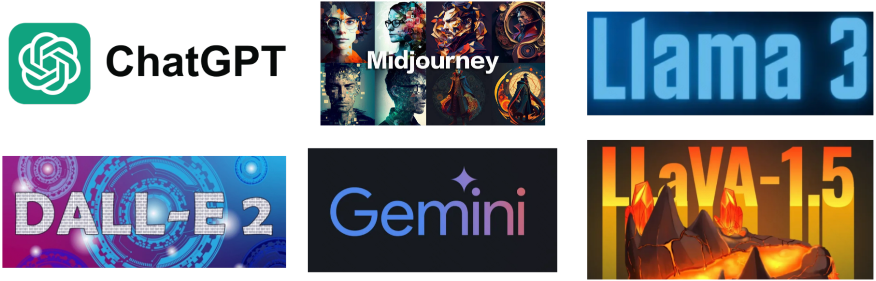


Figure: Large models are popular now

- Large models in various fields start to to evolve into giant ones to pursuit emergent abilities.
- Fine-tuning the pre-trained models using data of specific tasks is a typical usage pattern now.
- Due to the privacy security, fine-tuning for specific tasks is usually short of data.

BEIJING INSTITUTE OF TECHNOLOGY

FL disadvantages:

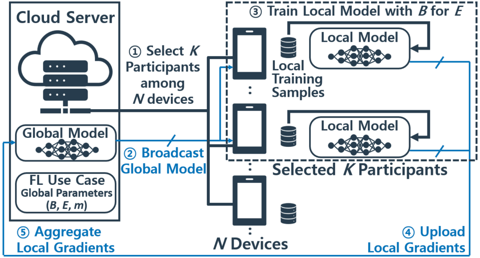


Figure: Overview of FL

- Communication difficulty
- Not conducive to large-scale deployment

HFL advantages:

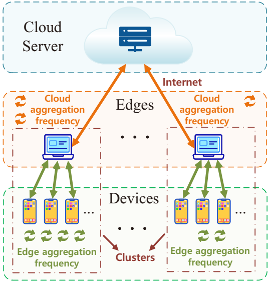


Figure: Overview of HFL

- Optimized communication
- Improve scalability

Hierarchical federated learning (HFL):

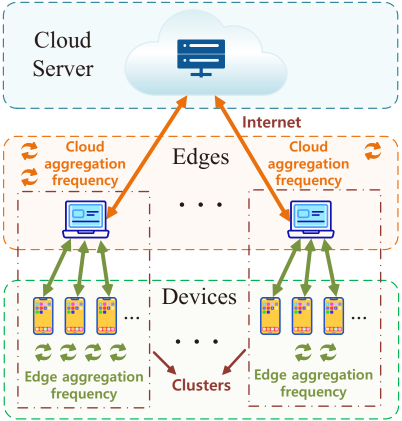


Figure: Overview of HFL

Training of HFL:

- Local training: Models deployed on devices
- Edge aggregation: Devices upload models to edges
- Cloud aggregation: Edges upload models to cloud

Challenge:

- Device heterogeneity
- Edge environment is dynamic
- Distribution of data changes

Our Work

How to design a fine-tuning system that can accelerate federated fine-tuning and improve energy efficiency?

Outline

Hierarchical federated learning:

- Local SGD(Device j):

$$f_j(w_j) = \frac{1}{|\mathcal{D}_j|} \sum_{(x,y)\in\mathcal{D}_j} f_j(w_j, x, y)$$

- Edge aggregation(Edge i):

$$w_i^e = \sum_{j=1}^{N_i} \frac{|\mathcal{D}_j|\, w_j}{\sum_{j=1}^{N_i} |\mathcal{D}_j|}$$

- Cloud aggregation:

$$w = \sum_{i=1}^{M} \frac{|\mathcal{D}_i|\, w_i^e}{\sum_{i=1}^{M} |\mathcal{D}_i|}$$

Hierarchical federated fine-tuning:

- Freeze backbone network parameters, fine-tune the adapter locally.
- During aggregation, only adapter parameters are uploaded for aggregation.

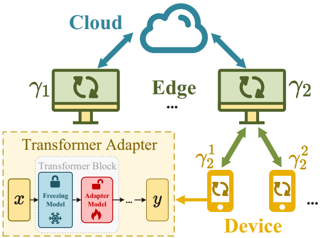
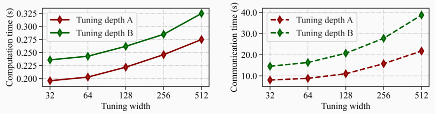


Figure: Framework of hierarchical federated fine-tuning

Background and Motivation

Influence of adapters:

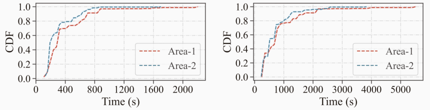


(a) Computation time (b) Communication time

Figure: Overhead of adapter-based fine-tuning

- Adapters can be customized by adjusting depth and width.
- Different adapter configurations, leading to varying local training time, communication time.
- Test on a transformer-based Bert model trained with the T-Rex dataset.

Influence of device heterogeneity:



(a) MobilBert (b) ALBert

Figure: Computing capability statistics under different edges

- Measure the computing capability by Measuring Broadband America (MBA) and AIBenchmark.
- The distribution indicate heterogeneity among devices at network and computation capability.
- The distribution between areas suggests heterogeneity between edges.

Observation 1

Hierarchical federated fine-tuning systems have strong heterogeneity.

Influence of model synchronization frequency:

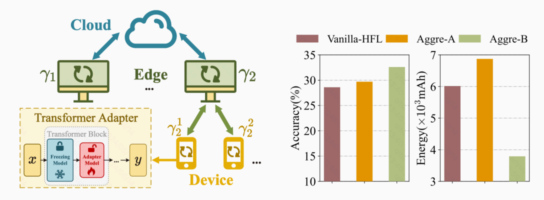


Figure: Accuracy and efficiency of different schemes

- System setting: A testbed consisting of 50 Raspberry Pis as devices and 5 laptops as edges.
- Aggre-A: Synchronization frequencies of devices whose CPU utilization is less than 30% are set to 5, and others are 1.
- Aggre-B: Conduct a fine-grained control by setting synchronization frequencies proportional to device CPU utilization.

Observation 2

Applying HFL federated fine-tuning by a sophisticated synchronization scheme exists a large optimization space.

Background and Motivation

Centralized control challenges:

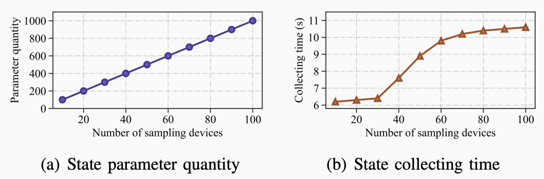


Figure: Overhead of centralized control in large-scale systems

- Almost all works are based on centralized control, which needs to collect global system states and run complex algorithms.
- The number of system parameters collected for centralized control linearly increases as the growth of devices.
- The time needed for decision making also increases to an unaffordable level.

Observation 3

We need to design a distributed solution to address the bottleneck incurred by centralized control.

Outline

Design of Tomtit

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Overview of Tomtit:

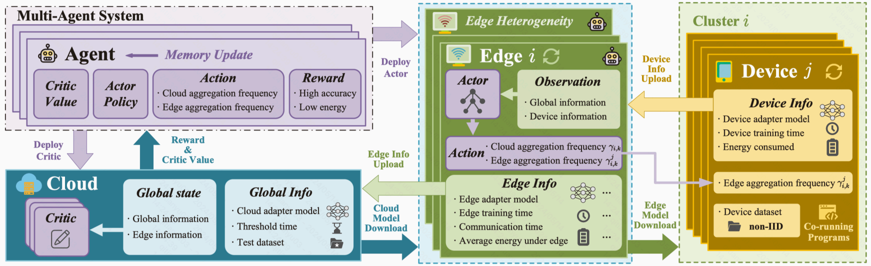


Figure: Tomtit design

Framework:

- Tomtit use a distributed synchronization control based on Multi-Agent Reinforcement Learning.
- Deploy a control module called an agent in each edge server, collecting information from server and devices.
- Each agent determines a synchronization frequency as the action for each associated device.

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Overview of Tomtit:

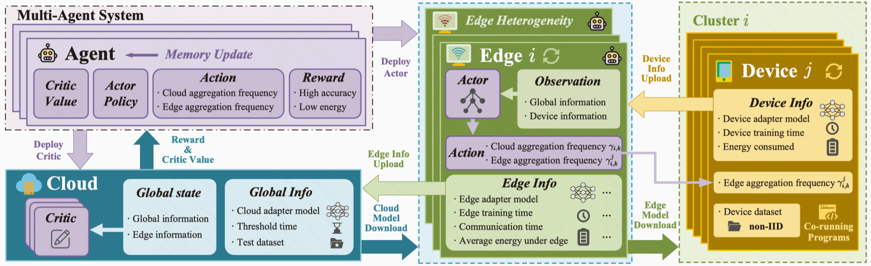


Figure: Tomtit design

Advantages:

- High accuracy: Aim to train adapters within a limited time to enhance model accuracy.
- Low energy consumption: Minimize the average energy consumption of devices as much as possible.
- Autonomy: Enable each agent to make independent and efficient decisions.

Design of Tomtit

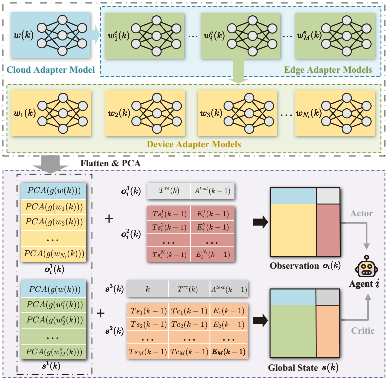


Figure: Composition of the observation and global state

The observation:

- Collect the observation for actor networks in the edges.
- Model parameters:

$$\boldsymbol{o}_i^1(k) = \mathrm{PCA}\{[g(w(k))^T \; g(w_1(k))^T \ldots g(w_{N_i}(k))^T]^T\}$$

- Time and energy consumption:

$$\boldsymbol{\varrho}_i^j(k) = [Ts_i^j(k-1) \; E_i^j(k-1)]$$

$$\boldsymbol{o}_i^2(k) = [\boldsymbol{\varrho}_i^1(k) \; \boldsymbol{\varrho}_i^2(k) \ldots \boldsymbol{\varrho}_i^{N_i}(k)]^T$$

- Global information:

$$\boldsymbol{o}_i^3(k) = [T^{re}(k) \; A^{test}(k-1)]$$

- Splice:

$$\boldsymbol{o}_i(k) = \mathrm{cat}\{(\boldsymbol{o}_i^1(k), \mathrm{cat}\{(\boldsymbol{o}_i^3(k), \boldsymbol{o}_i^2(k)), \mathrm{dim}=0\}), \mathrm{dim}=1\}$$

Design of Tomtit

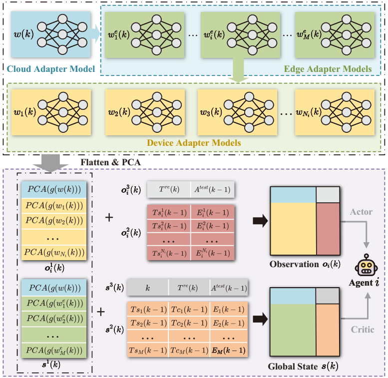


Figure: Composition of the observation and global state

The global state:

Enhancement

- Separate the actor and cirtic network of the agent
- Collect the global state for critic networks in the cloud.

- Model parameters:

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

- Time and energy consumption:

$$\boldsymbol{\varrho}_i(k) = [Ts_i(k-1) \; Tc_i(k-1) \; E_i(k-1)]$$

$$\boldsymbol{s}^2(k) = [\boldsymbol{\varrho}_1(k) \; \boldsymbol{\varrho}_2(k) \dots \boldsymbol{\varrho}_M(k)]^T$$

- Global information:

$$\boldsymbol{s}^3(k) = [k \; T^{re}(k) \; A^{test}(k-1)]$$

- Splice:

$$\boldsymbol{s}(k) = \mathrm{cat}\{(\boldsymbol{s}^1(k), \mathrm{cat}\{(\boldsymbol{s}^3(k), \boldsymbol{s}^2(k)), \mathrm{dim}=0\}), \mathrm{dim}=1\}$$

Design of Tomtit

北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

Action:

- In each round of cloud communication, agent i determines its action $\boldsymbol{a}_i(k) = \{\gamma_{i,k}, \gamma_{i,k}^1, \gamma_{i,k}^2, \dots, \gamma_{i,k}^{N_i}\}$.

Reward:

- Denote the reward given to agent i, where $Q(u) = \Upsilon^u$:

$$r_i(k) = Q(A^{test}(k)) - Q(A^{test}(k-1)) - \epsilon E_i(k)$$

Workflows:

1. Initialize the parameters;
2. Train HFL for several rounds and train PCA modules;
3. Agents make decision and choose action, push $(\boldsymbol{s}(k), \boldsymbol{o}_i(k), \boldsymbol{a}_i(k), r_i(k), \boldsymbol{s}(k+1), \boldsymbol{o}_i(k+1))$ to the memory;
4. Repeat step 3 until $T^{re}(k) < 0$;
5. Update each $Agent_i$ by state-action-reward in the agent's memory pool.

Algorithm 1 *Tomtit*'s Training Process

```
Initialize round of cloud aggregations k = 0, threshold time T, remaining time T^re(0) = T, global model w(0), Agent_i for edge i, i ∈ 𝕄;
Train several cloud aggregation by given aggregation frequencies, get w(1), w_i^e(1), w_j(1), and record T^use(1);
Train PCA module by w(1), w_i^e(1), and w_j(1);
Update T^re_init = T^re(1) − T^use(1); k++;
for 1 to Ω do
  Get s(k) and o_i(k) for Agent_i, i ∈ 𝕄;
  while true do
    a_i(k) = Agent_i.choose_action(o_i(k)), i ∈ 𝕄;
    Train HFL by {a_i(k)}_{i∈𝕄}, record T^use(k) and update T^re(k) = T^re(k − 1) − T^use(k);
    s(k + 1), o_i(k + 1), r_i(k) = Agent_i.step(), i ∈ 𝕄;
    τ_i^k = (s(k), o_i(k), a_i(k), r_i(k), s(k + 1), o_i(k + 1));
    Agent_i.push(τ_i^k), i ∈ 𝕄; k++;
    if T^re(k) < 0 then
      Set k = 1, T^re(k) = T^re_init;
      break
    end if
  end while
  Agent_i.PPO_update_&_clear(), i ∈ 𝕄;
end for
```

Convergence:

Assumption

- The loss function is L-smooth and the Lipschitz constant $L > 0$, i.e., $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$.
- The estimated stochastic gradient is unbiased for devices, i.e., $\mathbb{E}\left[\left\|\tilde{\nabla} f_j(w_j) - \nabla f(w)\right\|^2 \mid w\right] \leq \sigma^2$.

Convergence

- We can get the model updates formula as

$$w(k+1) = w(k) - \eta \sum_{i \in \mathbb{M}} \frac{N_i}{N} \frac{1}{N_i} \sum_{\alpha=0}^{\gamma_{i,k}-1} \sum_{j \in \mathbb{N}_i} \sum_{\beta=0}^{\gamma_{i,k}^{j}-1} \tilde{\nabla} f_j\left(w_j\left(k, \alpha, \beta\right)\right)$$

- The relationship between the models $w(k)$ and $w(k+1)$ is

$$\mathbb{E}[f\left(w(k+1)\right)] - \mathbb{E}[f\left(w(k)\right)] \leq \frac{L}{2} \mathbb{E}\left\|w(k+1) - w(k)\right\|^2 + \mathbb{E}\left\langle \nabla f\left(w(k)\right), w(k+1) - w(k)\right\rangle$$

Design of Tomtit

Theorem

- After subjecting our refined frequency variation method to a round of cloud communication, the convergence bound is

$$\mathbb{E}[f\left(w(k+1)\right)] - \mathbb{E}[f\left(w(k)\right)] \leq \frac{L^2\eta^3}{4}\widetilde{\gamma_1}\widetilde{\gamma_2}\left(\left(\widetilde{\gamma_1}-1\right)+\frac{M}{N}\widetilde{\gamma_1}\left(\widetilde{\gamma_2}-1\right)\right)\sigma^2 + \frac{L\eta^2}{2}\frac{1}{N}\widetilde{\gamma_1}\widetilde{\gamma_2}\sigma^2 - \frac{\eta}{2}\widetilde{\gamma_1}\widetilde{\gamma_2}\mathbb{E}\left\|\nabla f\left(w(k)\right)\right\|^2,$$

- where

$$P = 1 - \frac{L^2\eta^2\gamma_{i,k}^j(\gamma_{i,k}^j-1)}{2} - \frac{L^2\eta^2\widetilde{\boldsymbol{\gamma_2}}^2\gamma_{i,k}^j(\gamma_{i,k}-1)}{2} - L\eta\gamma_{i,k}\gamma_{i,k}^j.$$

Outline

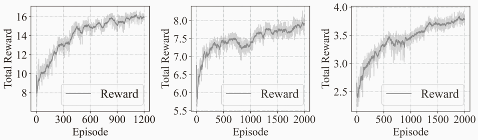
北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

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

Settings:

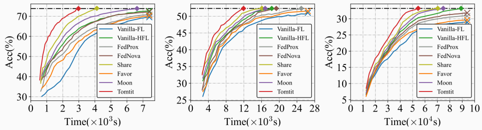
- Testbed: Edges are 5 (default) or 10, and devices are 20, 50 (default) and 100.
- Datasets: MNIST, Cifar-10, and Cifar-100.
- Baselines: Vanilla-FL, Vanilla-HFL, FedProx, FedNova, Share, Moon
- Heterogeneity:
  1. Different adapter settings for each device.
  2. Sample bandwidth from MobiPerf and apply to the edge.
  3. The data set is segmented in label non-IID and Dirichlet non-IID.

The performance of DRL training:



(a) MNIST (b) CIFAR-10 (c) CIFAR-100

Figure: The reward of training the MARL agent



(a) MNIST (b) CIFAR-10 (c) CIFAR-100

Figure: Accuracy v.s. time of different FL methods

Evaluation

The performance with different threshold time:

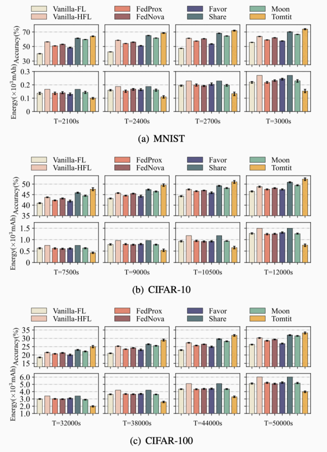


Figure: Performance under different training time

The performance with different non-IID levels:

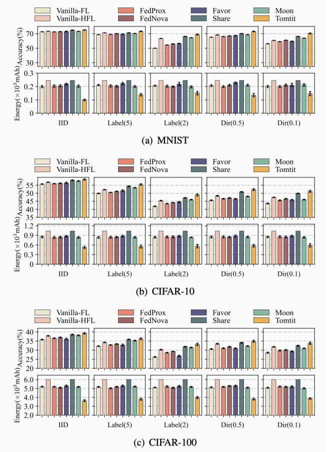


Figure: Performancesumption under different non-IID levels

Evaluation

Scalability:

- Change various scale configurations about the number of edges and devices.

TABLE I
PERFORMANCE AT DIFFERENT SCALES

Scale [Edges, Devices]	Data	*Vanilla-FL* Acc (%)	*Vanilla-FL* Energy (mAh)	*Vanilla-HFL* Acc (%)	*Vanilla-HFL* Energy (mAh)	*FedProx* Acc (%)	*FedProx* Energy (mAh)	*FedNova* Acc (%)	*FedNova* Energy (mAh)	*Favor* Acc (%)	*Favor* Energy (mAh)	*Share* Acc (%)	*Share* Energy (mAh)	*Moon* Acc (%)	*Moon* Energy (mAh)	*Tomtit* Acc (%)	*Tomtit* Energy (mAh)
[5, 20]	MNIST	51.8	218.3	58.9	268.7	55.3	225.3	57.4	226.8	53.1	209.5	62.2	268.7	60.9	228.1	**64.5**	**148.3**
	CIFAR-10	43.3	1258.8	46.0	1453.2	45.7	1240.3	45.8	1251.6	44.9	1289.2	47.8	1453.2	46.1	1247.4	**48.4**	**781.5**
	CIFAR-100	24.5	5008.3	28.4	6093.6	27.5	5109.7	27.9	5153.6	25.1	5220.7	29.2	6093.6	29.0	5111.6	**31.5**	**3844.5**
[5, 50]	MNIST	55.7	220.3	63.8	271.0	59.6	217.5	62.1	232.1	57.6	244.7	70.2	271.0	66.7	229.6	**73.8**	**152.7**
	CIFAR-10	46.5	1270.4	48.8	1494.8	47.5	1240.0	48.1	1249.3	47.4	1306.8	50.9	1494.8	49.4	1258.8	**52.9**	**758.6**
	CIFAR-100	26.3	5112.6	30.3	6014.8	28.6	5217.5	29.3	5087.4	26.8	5248.8	32.0	6014.8	31.4	5176.8	**33.2**	**3985.6**
[5, 100]	MNIST	48.1	231.8	54.2	270.8	52.1	218.9	52.9	206.9	50.8	228.7	56.3	270.8	55.7	222.5	**56.3**	**179.5**
	CIFAR-10	40.2	1249.2	44.1	1437.6	42.5	1286.3	43.4	1256.7	42.4	1251.2	**45.8**	1437.6	43.5	1248.5	45.2	**795.4**
	CIFAR-100	22.1	5023.1	27.0	5983.7	25.2	5154.2	26.1	5169.3	22.9	5201.5	28.1	5983.7	27.8	5281.4	**30.2**	**4021.0**
[10, 100]	MNIST	48.1	231.8	54.9	270.8	52.1	218.9	52.9	206.9	50.8	228.7	57.0	270.8	55.7	222.5	**58.6**	**161.2**
	CIFAR-10	40.2	1249.2	44.3	1437.6	42.5	1286.3	43.4	1256.7	42.4	1251.2	45.6	1437.6	43.5	1248.5	**46.9**	**781.8**
	CIFAR-100	22.1	5023.1	27.6	5983.7	25.2	5154.2	26.1	5169.3	22.9	5201.5	28.5	5983.7	27.8	5281.4	**30.6**	**3912.7**

Dynamic system environment:

- change data distribution and CPU utilization of devices.

TABLE II
ROBUSTNESS IN DYNAMIC ENVIRONMENTS

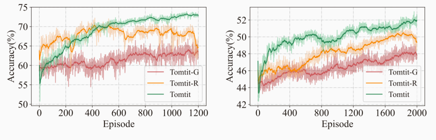
		10%	30%	50%
CPU	MNIST	73.71%(-0.21%)	73.45%(-0.37%)	73.12%(-0.60%)
	CIFAR-10	52.64%(-0.29%)	52.35%(-0.58%)	52.11%(-0.82%)
	CIFAR-100	32.92%(-0.31%)	32.76%(-0.47%)	32.44%(-0.79%)
Data	MNIST	73.15%(-0.67%)	72.91%(-0.91%)	72.50%(-1.21%)
	CIFAR-10	52.27%(-0.66%)	51.68%(-1.25%)	51.49%(-1.44%)
	CIFAR-100	32.85%(-0.38%)	32.11%(-1.12%)	31.70%(-1.53%)
CPU+Data	MNIST	72.54%(-1.28%)	72.05%(-1.77%)	71.86%(-1.94%)
	CIFAR-10	52.03%(-0.90%)	51.30%(-1.63%)	50.96%(-1.97%)
	CIFAR-100	32.49%(-0.74%)	31.88%(-1.35%)	31.24%(-1.99%)

The impact of agent design:

- 1.Tomtit-G: Concatenating observations as the global state;
- 2.Tomtit-R: replace $Q(u) = \Upsilon^u$ with $Q(u) = u$ in the reward.

State dimension:

- Change the dimensionality of PCA of the state.



(a) Accuracy of training MNIST (b) Accuracy of training CIFAR-10

Figure: Training the DRL agent

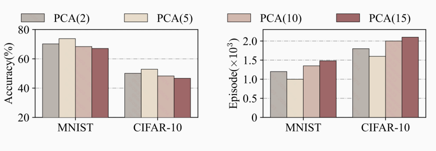


Figure: Impact of different principal component

Outline

Conclusion

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

Conclusion

- We propose a hierarchical federated fine-tuning system, which can achieve higher model accuracy and lower energy consumption when fine-tuning models.
- We design a distributed algorithm based on MARL to determine the aggregation frequency of edges and devices, and prove convergence.
- We conduct extensive experiments comparing with the state-of-the-arts in a real system.

Thank you for your attention!

Questions?