FedProphet: Memory-Efficient Federated Adversarial Training via Robust and Consistent Cascade Learning

Minxue Tang[*1] Yitu Wang[*1] Jingyang Zhang[1] Louis DiValentin[2] Aolin Ding[2] Amin Hass[2]
Yiran Chen[1] Hai "Helen" Li[1]
[1]Duke University, Dept. of Electrical and Computer Engineering
[2]Accenture Cyber Labs

MLSys

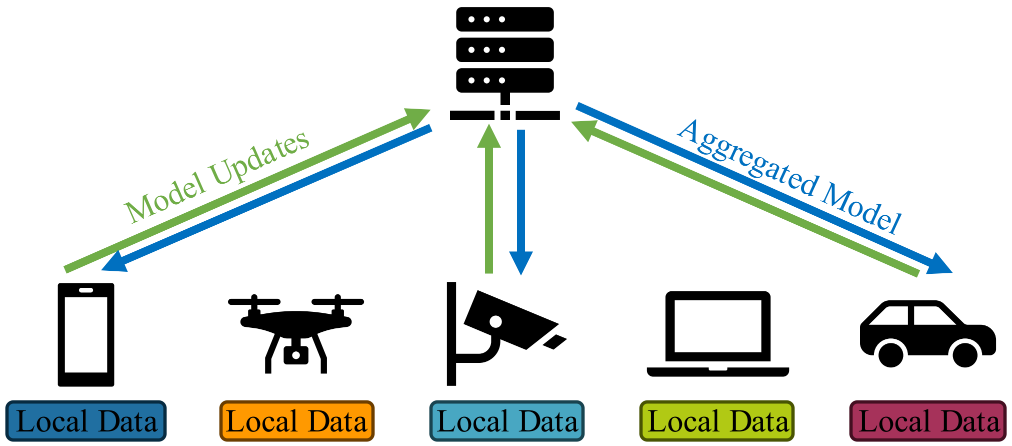
accenture Duke

Background: Federated Learning

- Update Rule of Federated Learning
  - Partial participation
  - Multi-step local SGD
  - Central Aggregation with Average

$$w_{t+1} = w_t - \eta_t \sum_{k \in \mathbb{K}_t} p_k \tilde{g}_{k,t}$$

$$\tilde{g}_{k,t} = \sum_{j=0}^{\tau_k - 1} \mathbb{E}_{x \sim p_{x,k}} \left[\nabla l_{k,t}(x; w_{k,t,j}) \right]$$



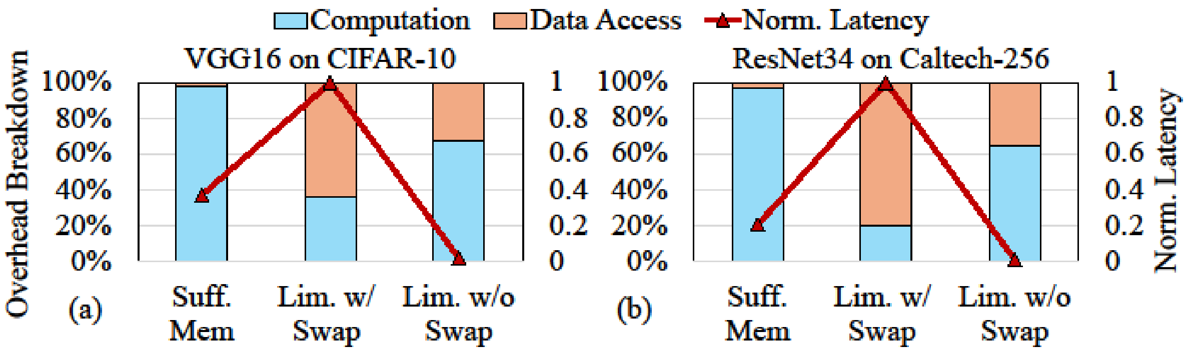
Duke

Motivations

- Federated learning can provide privacy guarantee but cannot provide robustness guarantee against adversarial examples.
- Adversarial training can provide robustness enhancement but requiring more computational resources.

$$\boxed{\min_{w} \max_{\|\delta\| \leq \epsilon} l(x+\delta; w)}$$

Dataset	CIFAR10		Caltech256	
Model Size	Clean Acc.	Adv. Acc.	Clean Acc.	Adv. Acc.
FAT-Large	79.74%	56.76%	46.56%	17.76%
FAT-Small	66.57%	54.33%	25.64%	13.49%
FedRolex-AT	67.14%	54.13%	30.18%	11.78%



Duke

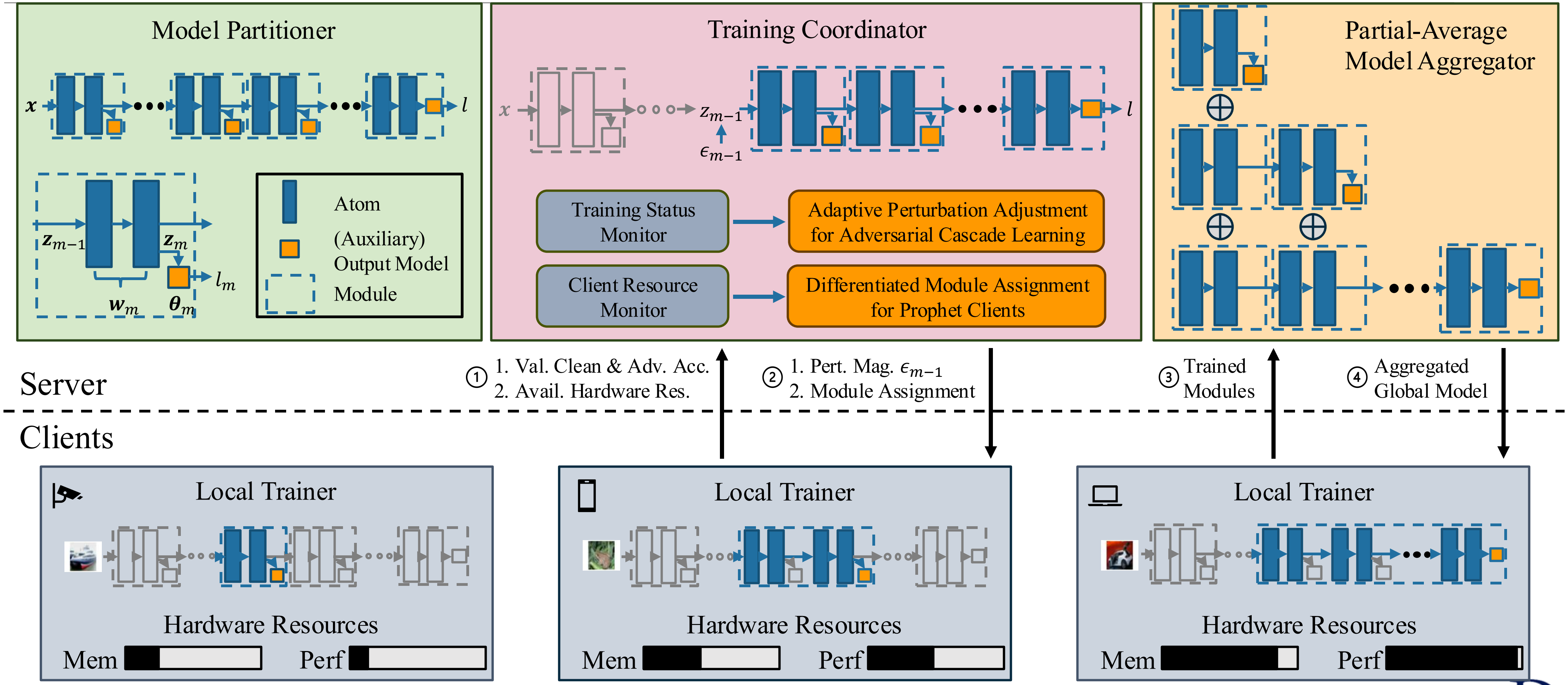
Motivation

- Previous memory-efficient federated learning methods have large objective inconsistency incurred by systematic heterogeneity.
  - To tackle the insufficient computational resources on some clients, previous methods usually allow them to train small models or small parts of the global model.
- Objective inconsistency causes poor convergence.

$$\boxed{\xi_t^2 = \left\| \nabla l_{k,t}(x; w_t) - \nabla l(x; w_t) \right\|^2}$$

Duke

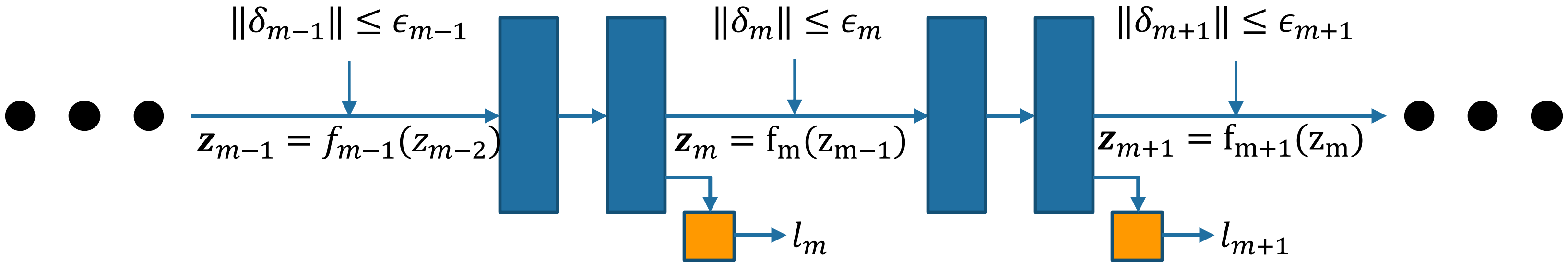
System Framework



Duke

Client: Local Trainer
Adversarial Cascade Learning

- Guarantee the joint robustness.



Sufficient condition for joint robustness

$$\boxed{\max_{\|\delta_{m-1}\| \leq \epsilon_{m-1}} \|f_m(z_{m-1} + \delta_{m-1}) - f_m(z_{m-1})\| \leq \epsilon_m}$$

- Solution 1: Adding regularization on $\|f_m(z_{m-1} + \delta_{m-1}) - f_m(z_{m-1})\|$ directly

$$l_m^{adv} = l_m(z_{m-1}) + \mu_m \max_{\|\delta_{m-1}\| \leq \epsilon_{m-1}} \|f_m(z_{m-1} + \delta_{m-1}) - f_m(z_{m-1})\|^2$$

  - Drawbacks: doubles the batch size and increases the memory requirement.

Client: Local Trainer

Adversarial Cascade Learning with Strong Convexity Regularization

- Solution 2: Making the loss strongly convex in z_m:

$$l_m^{adv} = \max_{\|\delta_{m-1}\| \le \epsilon_{m-1}} [l_m(z_{m-1} + \delta_{m-1}) + \frac{\mu_m}{2} \|f_m(z_{m-1} + \delta_{m-1})\|^2]$$

$$\Rightarrow \max_{\|\delta_{m-1}\| \le \epsilon_{m-1}} \|f_m(z_{m-1} + \delta_{m-1}) - f_m(z_{m-1})\| \le \frac{g_m}{\mu_m} + \sqrt{\frac{2c_m}{\mu_m} + \frac{g_m^2}{\mu_m^2}}$$

  - Use a single linear layer as the auxiliary output model to guarantee the convexity
  - Use ℓ_2 regularization to guarantee the μ-strong convexity

Duke

Client: Local Trainer

Robustness-Consistency Relationship

- Object Inconsistency

$$\boxed{\|\nabla_{w_m} l - \nabla_{w_m} l_m\|_2 \leq \left\| \frac{\partial z_m}{\partial w_m} \right\|_2 \sqrt{2(c_m + c_M)(\beta_m + \beta'_m)}.}$$

- β'_m(smoothness of the joint loss) and c_M (sensitivity of the joint loss) are small if we ensure joint robustness
- β_m (smoothness of the module loss) and c_m (sensitivity of the module loss) are small if we ensure module robustness

Duke

Server: Model Partitioner

- All modules must satisfy the memory constraint.
  - Module Size <= Min Reserved Memory
- Greedy partitioning
  - Go through each atom in the forward propagation order
  - Add atoms into the module until reach the memory limits
  - Begin the next module

Algorithm 1: Memory-constrained Model Partition

Require: The "atom" sequence $(a_1 \circ \cdots \circ a_L)$;
  Minimal reserved memory $R_{\min}$

Initialize $\mathbb{M} = \emptyset, m = \emptyset$;

for $i \leq L$ **do**

  if $MemReq(m \cup \{a_i\}) < R_{min}$ **then**

    Append a_i to m;

  else

    Append m to $\mathbb{M}$;

    $m \leftarrow \{a_i\}$;

Append m to $\mathbb{M}$;

Result: Model partition $\mathbb{M}$

Duke

Server: Training Coordinator

Adaptive Perturbation Adjustment

- Adversarial Perturbation Magnitude ϵ_m
  - It is sufficient but not necessary: $\epsilon_m = \max_{z_{m-1}, \|\delta_{m-1}\| \le \epsilon_{m-1}} \|f_m(z_{m-1} + \delta_{m-1}) - f_m(z_{m-1})\|$
  - $\epsilon_m^{(t)} = \alpha_m^{(t)} \mathbb{E}_{z_{m-1}} \left[\max_{\|\delta_{m-1}\| \le \epsilon_{m-1}^*} \|f_m(z_{m-1} + \delta_{m-1}) - f_m(z_{m-1})\| \right]$
- Adaptive adjustment of $\alpha_m^{(t)}$

$$\boxed{\alpha_m^{(t)} = \begin{cases} \alpha_m^{(t-1)} + \Delta\alpha, \text{if } \dfrac{C_{m+1}^{(t)}}{A_{m+1}^{(t)}} > (1+\gamma)\dfrac{C_m^*}{A_m^*}; \\ \alpha_m^{(t-1)} - \Delta\alpha, \text{if } \dfrac{C_{m+1}^{(t)}}{A_{m+1}^{(t)}} < (1-\gamma)\dfrac{C_m^*}{A_m^*}; \\ \alpha_m^{(t-1)}, \text{elsewhere} \end{cases}}$$

Duke

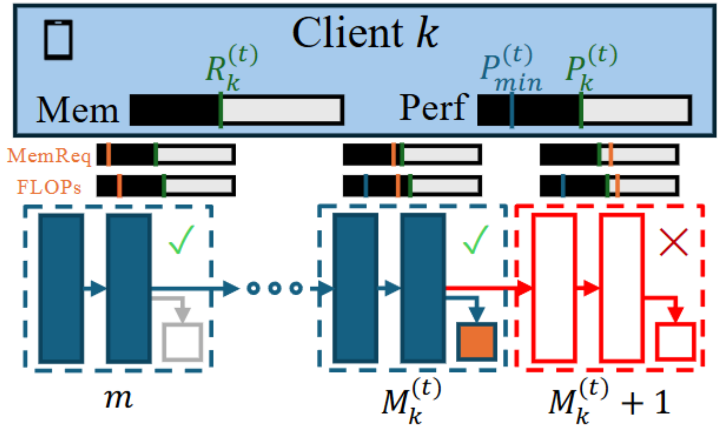
Server: Training Coordinator

Differentiated Module Assignment

- Train more modules on resource-sufficient clients
  - The combined modules fit the real-time available computational resources on each device:

$$\mathrm{MemReq}(m \circ m+1 \circ \cdots \circ M_k^{(t)}) \le R_k^{(t)}.$$

$$\mathrm{FLOPs}(m \circ m+1 \circ \cdots \circ M_k^{(t)}) \le \frac{P_k^{(t)}}{P_{\min}^{(t)}} \mathrm{FLOPs}(m).$$



- Devices with more computational resources become the prophet: train more modules to see what will happen in the future training stage and help reduce the objective inconsistency.

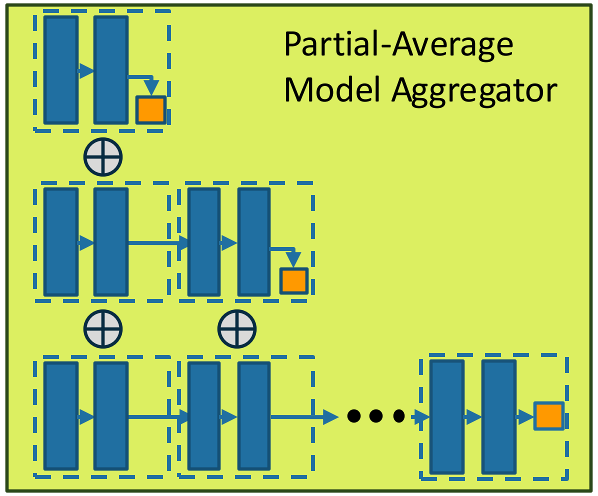
Duke

Server: Partial-Average Model Aggregator

- Each parameter is averaged only among clients who trained this parameter in this communication round.

$$\mathbf{w}_n^{(t+1)} = \frac{\sum_{k \in \mathbb{S}_n^{(t)}} q_k \mathbf{w}_{n,k}^{(t,E)}}{\sum_{k \in \mathbb{S}_n^{(t)}} q_k}, \quad \mathbb{S}_n^{(t)} = \{k : M_k^{(t)} \geq n\},$$

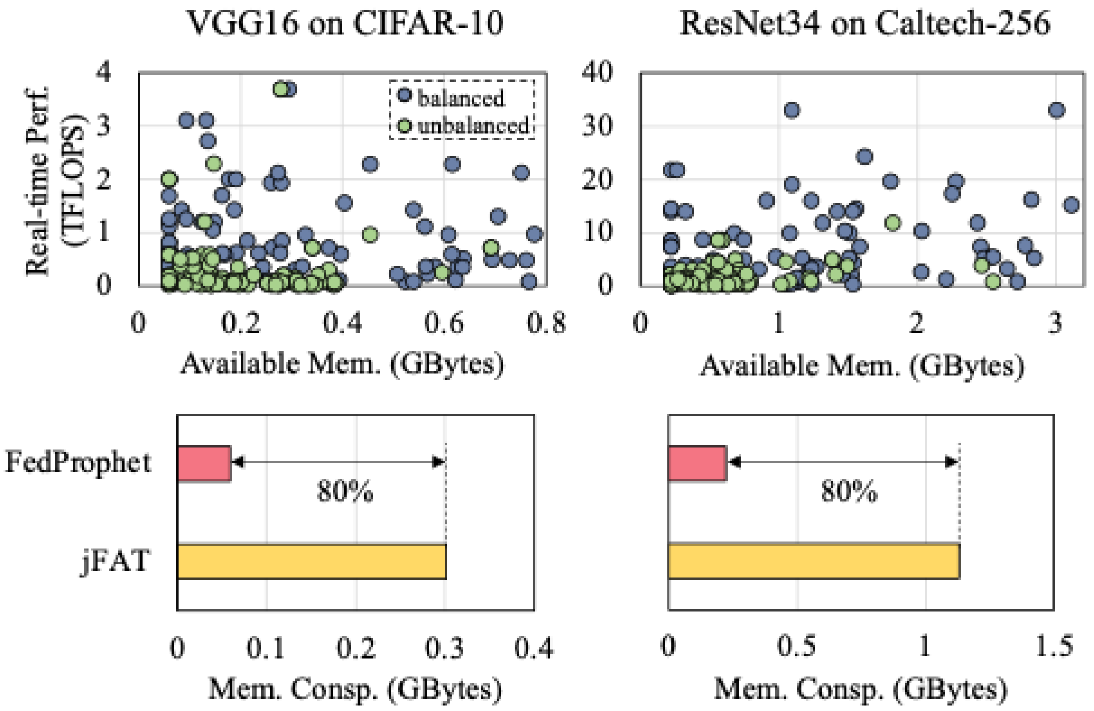
$$\boldsymbol{\theta}_n^{(t+1)} = \frac{\sum_{k \in \mathbb{K}_n^{(t)}} q_k \boldsymbol{\theta}_{n,k}^{(t,E)}}{\sum_{k \in \mathbb{K}_n^{(t)}} q_k}, \quad \mathbb{K}_n^{(t)} = \{k : M_k^{(t)} = n\}.$$



Duke

Experiments: Hardware Sampling

- We sample the devices from pools of devices



(a) VGG16 with $R_{\min} = 60$ MB.

Module	Layer	Mem. Req.	FLOPs
1	Conv 1 Conv 2	55.8 MB	2.6 G
2	Conv 3 Conv 4 Conv 5	46.1 MB	4.9 G
3	Conv 6 Conv 7 Conv 8	50.4 MB	6.0 G
4	Conv 9	34.7 MB	2.4 G
5	Conv 10	33.1 MB	2.4 G
6	Conv 11 Conv 12	59.3 MB	1.2 G
7	Conv 13 Linear 1 Linear 2 Linear 3	36.1 MB	0.6 G

(b) ResNet34 with $R_{\min} = 224$ MB.

Module	Layer/Block	Mem. Req.	FLOPs
1	Conv	148.6 MB	3.9 G
2	BasicBlock 1	130.2 MB	7.5 G
3	BasicBlock 2	130.2 MB	7.5 G
4	BasicBlock 3 BasicBlock 4	197.9 MB	13.3 G
5	BasicBlock 5 BasicBlock 6 BasicBlock 7 BasicBlock 8	221.6 MB	28.1 G
6	BasicBlock 9 BasicBlock 10 BasicBlock 11 BasicBlock 12 BasicBlock 13	206.5 MB	37.1 G
7	BasicBlock 14 BasicBlock 15 BasicBlock 16 Linear	204.0 MB	20.6 G

Duke

Empirical Results

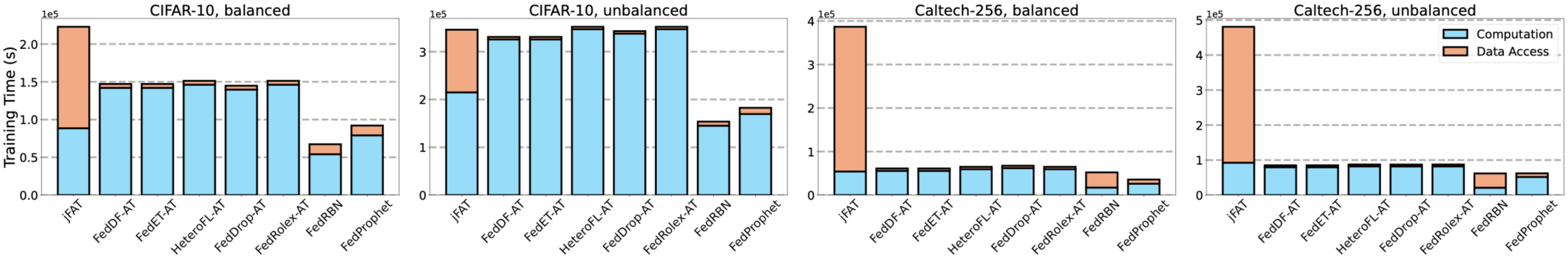
- Higher accuracy and robustness
  - Comparable to joint training

Dataset	CIFAR-10 (32 × 32)						Caltech-256 (224 × 224)					
Sys. Hetero.	balanced			unbalanced			balanced			unbalanced		
Method	Clean Acc.	PGD Acc.	AA Acc.	Clean Acc.	PGD Acc.	AA Acc.	Clean Acc.	PGD Acc.	AA Acc.	Clean Acc.	PGD Acc.	AA Acc.
jFAT	79.74%	56.76%	55.01%	79.74%	56.76%	55.01%	46.56%	19.76%	18.36%	46.56%	19.76%	18.36%
FedDF-AT	47.77%	24.88%	18.72%	48.16%	25.39%	18.34%	6.74%	4.83%	4.10%	11.78%	0.09%	0%
FedET-AT	40.73%	7.29%	5.12%	34.91%	8.74%	5.54%	11.48%	2.76%	2.44%	16.49%	1.92%	1.73%
HeteroFL-AT	51.63%	39.36%	38.47%	55.25%	43.05%	41.96%	27.80%	8.70%	8.15%	9.43%	3.04%	2.87%
FedDrop-AT	65.92%	54.21%	53.23%	63.26%	53.21%	52.61%	27.10%	11.87%	10.05%	11.68%	6.54%	5.20%
FedRolex-AT	67.14%	54.13%	53.51%	66.44%	53.25%	52.00%	30.18%	11.78%	9.84%	12.51%	5.80%	4.81%
FedRBN	**84.81%**	42.88%	39.82%	**86.70%**	42.99%	39.85%	**78.38%**	3.14%	0%	**78.81%**	1.43%	0%
FedProphet	77.79%	**59.22%**	**57.89%**	76.47%	**59.51%**	**58.64%**	47.07%	**19.10%**	**18.11%**	43.39%	**14.93%**	**14.41%**

Duke

Empirical Results

- Less training time
  - Avoid memory swapping and synchronization time



Conclusions

- We propose consistent and robust adversarial cascade learning with strong convexity regularization to reduce the memory requirement for federated adversarial training.
- We propose a server coordinator, with adaptive perturbation adjustment to balance the utility and robustness, and differentiated module assignment to further reduce the objective inconsistency.
- FedProphet maintains almost the same accuracy and robustness as joint federated adversarial training, while reducing 80% memory or achieving up to 11x speedup in training time.

FedProphet: Memory-Efficient Federated Adversarial Training via Robust and Consistent Cascade Learning

Minxue Tang[*1] Yitu Wang[*1] Jingyang Zhang[1] Louis DiValentin[2] Aolin Ding[2] Amin Hass[2]
Yiran Chen[1] Hai “Helen” Li[1]
[1]Duke University, Dept. of Electrical and Computer Engineering
[2]Accenture Cyber Labs

MLSys