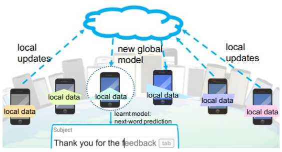
FedAST: Federated Asynchronous Simultaneous Training

Baris Askin Pranay Sharma Carlee Joe-Wong Gauri Joshi
Carnegie Mellon University

Motivation

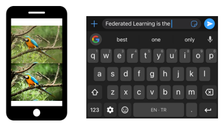
Federated Learning (FL)

- Train ML models with distributed data
- Heterogeneous data across clients
- Need for client data privacy



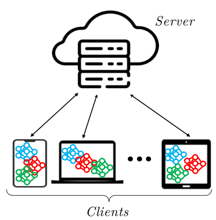
FL with Multiple Models

- Devices need federated training of multiple ML models
- E.g., smartphones use next-word predictor and image enhancer



Problem

How can we efficiently train models for multiple tasks in a federated setting using a shared pool of clients?



- M tasks and M models
- N clients
- **Goal:** For each objective $m \in \{1, \dots, M\}$

$$\min_{x_m \in \mathbb{R}^{d_m}} f_m(x_m)$$

where $f_m(x_m) = \frac{1}{N}\sum_{i=1}^{N} f_{m,i}(x_m)$: global loss function for model m,

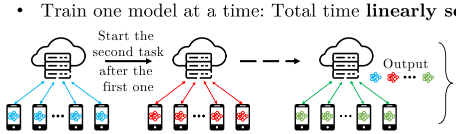
$f_{m,i}(x_m)$: local loss function for model m at client i.

Baselines

Clients are compute- and memory-limited

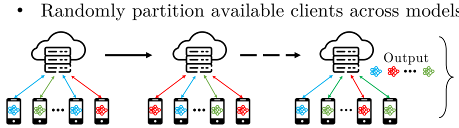
1. Sequential Training:

- Train one model at a time: Total time **linearly scales with M**

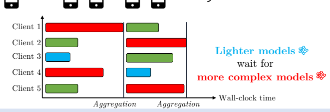


2. Federated Synchronous Simultaneous Training[2](Sync-ST):

- Train all models simultaneously
- Randomly partition available clients across models every round



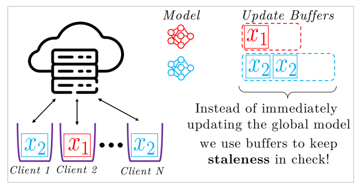
- **Straggler Problem!**

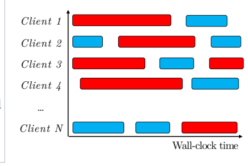


Algorithm

FedAST: Federated Asynchronous Simultaneous Training[1]

Assume we have two models and N clients.





1. The server initializes the training by sending local training requests.

2. Clients asynchronously run *local training* (SGD steps) and send updates to the server.

3. The server buffers the received updates. For each received update, a new training request is assigned to a *randomly selected client.*

4. *Aggregation* when a buffer is full in the server:

$$x_m^{t+1} \leftarrow x_m^t + \frac{1}{\textit{Buffer Size}} \sum_{\Delta \in \textit{Buffer}} \Delta$$

Theoretical Guarantees

Assumptions: For all clients and tasks;

1. Local loss functions are smooth
2. Stochastic gradients have bounded variance
3. Stochastic gradients are unbiased estimators of full gradient
4. The data heterogeneity across clients is bounded
5. All client updates are completed within a finite number of server rounds

Convergence Theorem: Every model m converges to a stationary point of global loss function f_m :

$$\frac{1}{T_m}\sum_{t=0}^{T_m-1} \mathbb{E}||\nabla f_m(x_m^t)||^2 \leq \underbrace{\mathcal{O}\left(\frac{1}{\sqrt{b_m T_m}}\right)}_{\text{Sync. FL error}} + \underbrace{\mathcal{O}\left(\frac{R_m^2}{b_m T_m}\right)}_{\text{Async. aggregation error}}$$

where T_m : #global rounds, b_m : buffer size, R_m: #active local trainings for model m

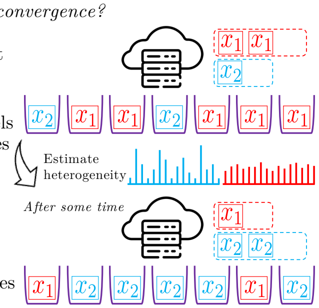
Compared to works with the same convergence rate

Method	Setting	Non-standard Assumptions	Multiple Local SGD Steps	Buffer
[3]	Single-model	Bounded Gradient Uniform Updates	✓	✓
[4]	Single-model	—	✗	✗
FedAST	Multi/single-model	—	✓	✓

Dynamic Resource Allocation

How much resource should we allocate to each task for fast convergence?

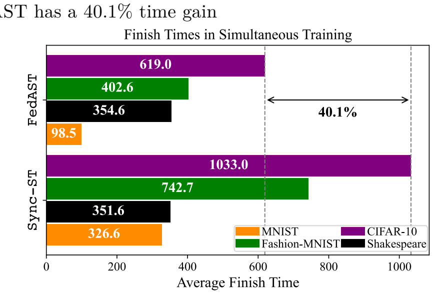
- Tasks have different inter-client data heterogeneity levels
- Estimate the heterogeneity levels using the latest received updates
- Allocate **more clients** and a **larger buffer** to tasks with **high data heterogeneity**
- *Periodically* adjusts the resources based on the needs of tasks!



Experiments

FedAST vs. Sync. Federated Simultaneous Training

- Train 4 models simultaneously
- Simulated wall clock times until the convergence
- Sync-ST suffers from straggler effect
- FedAST has a 40.1% time gain



Takeaways

For Federated Learning with multiple models, we propose;

- Asynchronous and simultaneous algorithm: Solves straggler problem
- Buffers: Solves staleness problem
- Dynamic Resource Allocation: Faster convergence
- Theoretical and experimental superiority

askinb.github.io for the full paper & code!

References

[1] Askin, et al., "FedAST: Federated Asynchronous Simultaneous Training," UAI, 2024.
[2] Bhuyan, et al. "Multi-model federated learning with provable guarantees," EAI ValueTools, 2022.
[3] Nguyen, et al. "Federated learning with buffered asynchronous aggregation." AISTATS, 2022.
[4] Koloskova, et al. "Sharper convergence guarantees for asynchronous SGD for distributed and federated learning." NeurIPS, 2022.

Funding to attend this conference was provided by the CMU GSA/Provost Conference Funding.