Fed-Exp

Federated Algorithm With An Exponential Weighted Average Approach

K. Sai Anuroop

170030035, Final Year, Dept. of Computer Science and Engineering

Supervised by: **B. N. Bharath**, Dept. of Electrical Engineering

Indian Institute of Technology Dharwad

April 15, 2021

Previous Work in a Nutshell (BTP - I)

- Considered a specific FL problem under a non-I.I.D. data setting
- Provided a sound theoretical framework for the proposed algorithm based on a novel Bayesian approach
- Introduced a new complexity measure as a consequence of the PAC bound
- Submitted a paper titled *"Federated Algorithm With Bayesian Approach: Omni-Fedge"* to IEEE ICASSP 2021, which is **accepted** for presentation at the conference

Previous Work in a Nutshell (BTP - I)

- Considered a specific FL problem under a non-I.I.D. data setting
- Provided a sound theoretical framework for the proposed algorithm based on a novel Bayesian approach
- Introduced a new complexity measure as a consequence of the PAC bound
- Submitted a paper titled *"Federated Algorithm With Bayesian Approach: Omni-Fedge"* to IEEE ICASSP 2021, which is **accepted** for presentation at the conference

Previous Work in a Nutshell (BTP - I)

- Considered a specific FL problem under a non-I.I.D. data setting
- Provided a sound theoretical framework for the proposed algorithm based on a novel Bayesian approach
- Introduced a new complexity measure as a consequence of the PAC bound
- Submitted a paper titled *"Federated Algorithm With Bayesian Approach: Omni-Fedge"* to IEEE ICASSP 2021, which is **accepted** for presentation at the conference

Previous Work in a Nutshell (BTP - I)

- Considered a specific FL problem under a non-I.I.D. data setting
- Provided a sound theoretical framework for the proposed algorithm based on a novel Bayesian approach
- Introduced a new complexity measure as a consequence of the PAC bound
- Submitted a paper titled *"Federated Algorithm With Bayesian Approach: Omni-Fedge"* to IEEE ICASSP 2021, which is **accepted** for presentation at the conference

Online Learning: Background

- Online learning algorithms are suitable for modern applications since they provide an efficient solution for large-scale problems
- These algorithms process one sample at a time with an update per iteration that is often computationally cheap and easy to implement
- On-line algorithms **do not** require any distributional assumption: their analysis assumes an adversarial scenario
- Note that this is in stark contrast to our previous work, as there is no notion of generalisation in the Online Learning scenario

Online Learning: Background

- Online learning algorithms are suitable for modern applications since they provide an efficient solution for large-scale problems
- These algorithms process one sample at a time with an update per iteration that is often computationally cheap and easy to implement
- On-line algorithms **do not** require any distributional assumption: their analysis assumes an adversarial scenario
- Note that this is in stark contrast to our previous work, as there is no notion of generalisation in the Online Learning scenario

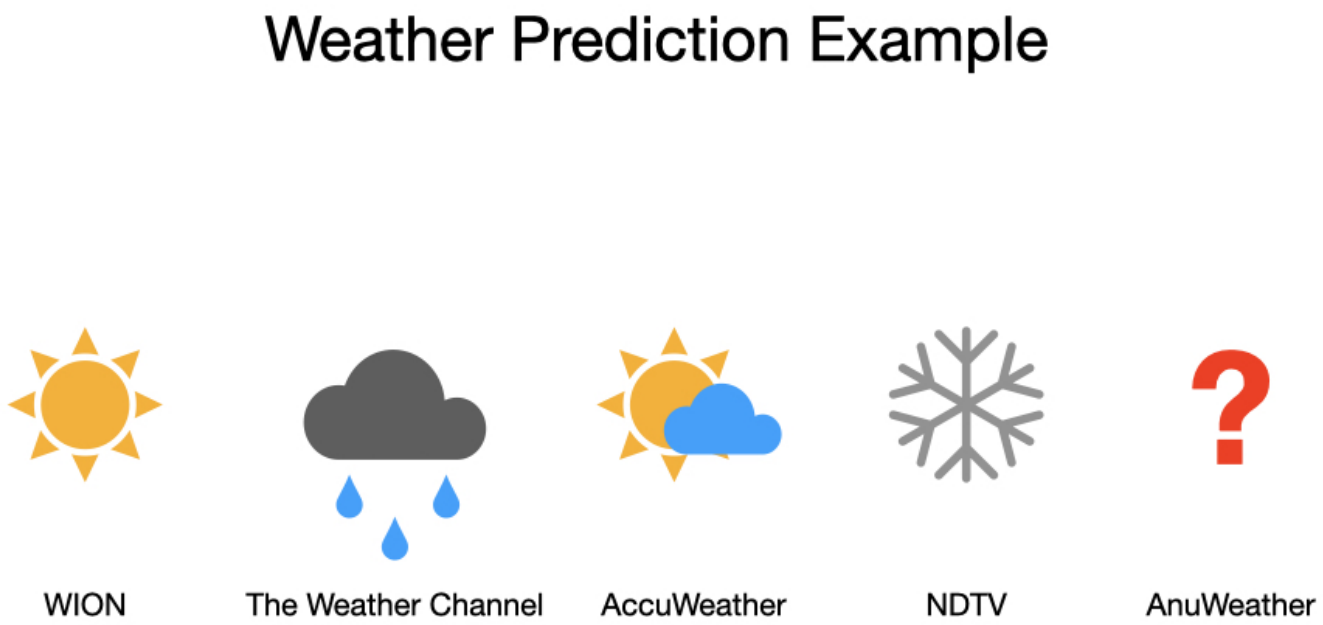
Online Learning: Background

- Online learning algorithms are suitable for modern applications since they provide an efficient solution for large-scale problems
- These algorithms process one sample at a time with an update per iteration that is often computationally cheap and easy to implement
- On-line algorithms **do not** require any distributional assumption: their analysis assumes an adversarial scenario
- Note that this is in stark contrast to our previous work, as there is no notion of generalisation in the Online Learning scenario

Online Learning: Background

- Online learning algorithms are suitable for modern applications since they provide an efficient solution for large-scale problems
- These algorithms process one sample at a time with an update per iteration that is often computationally cheap and easy to implement
- On-line algorithms **do not** require any distributional assumption: their analysis assumes an adversarial scenario
- Note that this is in stark contrast to our previous work, as there is no notion of generalisation in the Online Learning scenario

Online Learning: Example



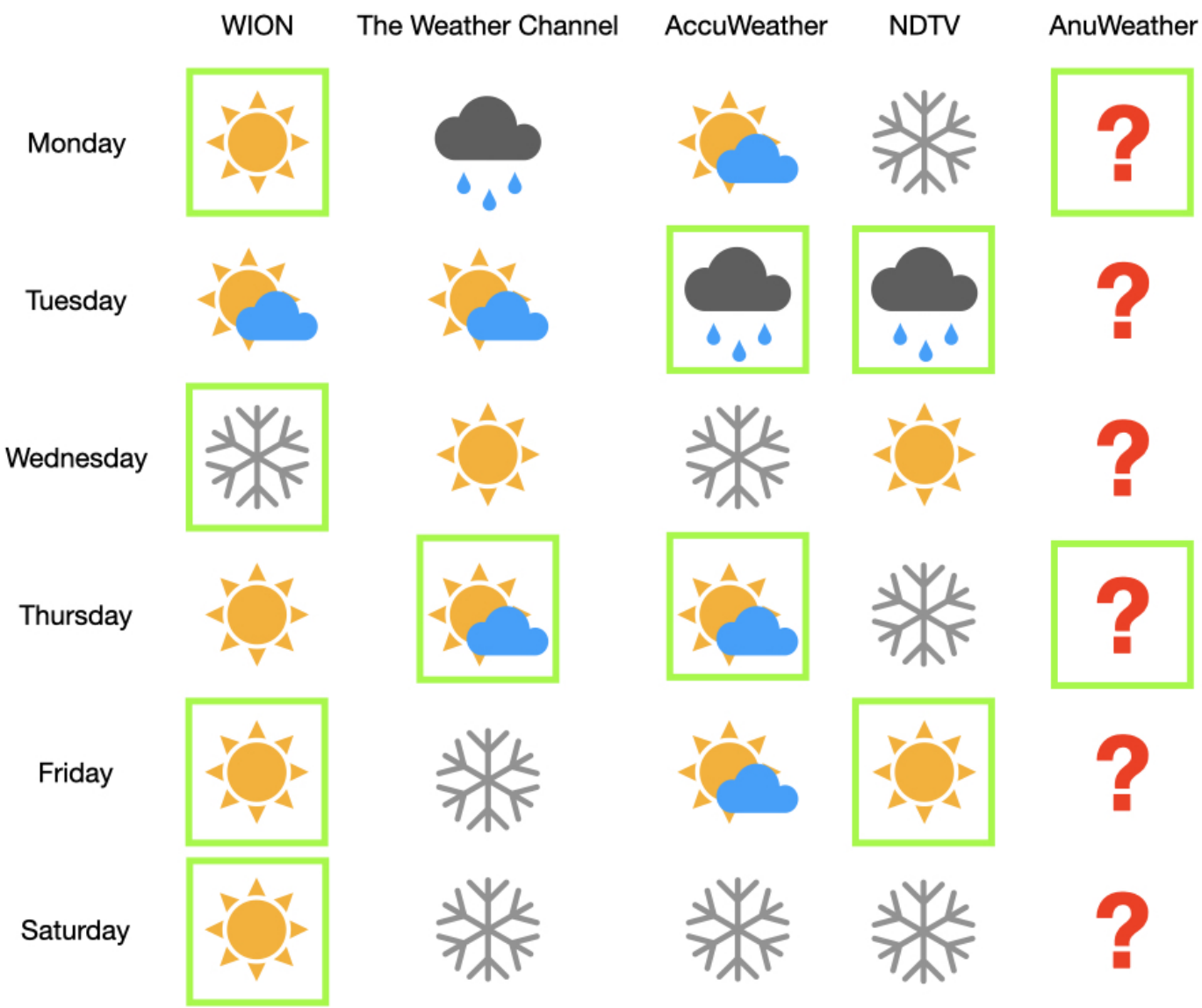
Online Learning: Example

Online Learning: Example

Online Learning: Example

y_t

Online Learning: Example



Regret

- The objective in this setting is to minimize the regret R_T, which compares the cumulative loss of the algorithm to that of the best expert in *hindsight* after T rounds of prediction:

$$R_T = \sum_{t=1}^{T} \mathcal{L}(\hat{y}_t, y_t) - \min_{i=1}^{N} \sum_{t=1}^{T} \mathcal{L}(\hat{y}_{i,t}, y_t)$$

Regret

- The objective in this setting is to minimize the regret R_T, which compares the cumulative loss of the algorithm to that of the best expert in *hindsight* after T rounds of prediction:

$$R_T = \sum_{t=1}^{T} \mathcal{L}(\hat{y}_t, y_t) - \min_{i=1}^{N} \sum_{t=1}^{T} \mathcal{L}(\hat{y}_{i,t}, y_t)$$

Our Work: Motivation

- We require a simple method to weigh the neural networks learnt by various nodes in the network
- Exponential weighted average is one such simple method which does the job
- It is in particular suitable for online learning scenario where the neural network needs to adapt itself quickly to the incoming data
- Though FL in online scenario is studied, most of these provide equal weight to the nodes in the network
- Fed-Exp is 'lighter' communication and computation-wise when compared to Omni-Fedge, our previous work

Our Work: Motivation

- We require a simple method to weigh the neural networks learnt by various nodes in the network
- Exponential weighted average is one such simple method which does the job
- It is in particular suitable for online learning scenario where the neural network needs to adapt itself quickly to the incoming data
- Though FL in online scenario is studied, most of these provide equal weight to the nodes in the network
- Fed-Exp is 'lighter' communication and computation-wise when compared to Omni-Fedge, our previous work

Our Work: Motivation

- We require a simple method to weigh the neural networks learnt by various nodes in the network
- Exponential weighted average is one such simple method which does the job
- It is in particular suitable for online learning scenario where the neural network needs to adapt itself quickly to the incoming data
- Though FL in online scenario is studied, most of these provide equal weight to the nodes in the network
- Fed-Exp is 'lighter' communication and computation-wise when compared to Omni-Fedge, our previous work

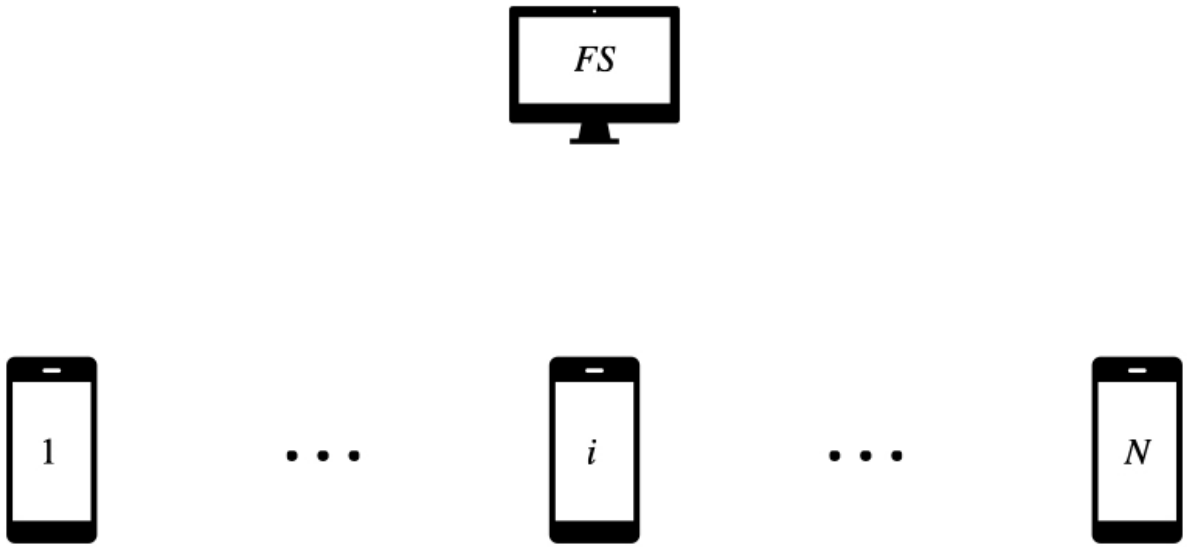
Our Work: Motivation

- We require a simple method to weigh the neural networks learnt by various nodes in the network
- Exponential weighted average is one such simple method which does the job
- It is in particular suitable for online learning scenario where the neural network needs to adapt itself quickly to the incoming data
- Though FL in online scenario is studied, most of these provide equal weight to the nodes in the network
- Fed-Exp is 'lighter' communication and computation-wise when compared to Omni-Fedge, our previous work

Our Work: Motivation

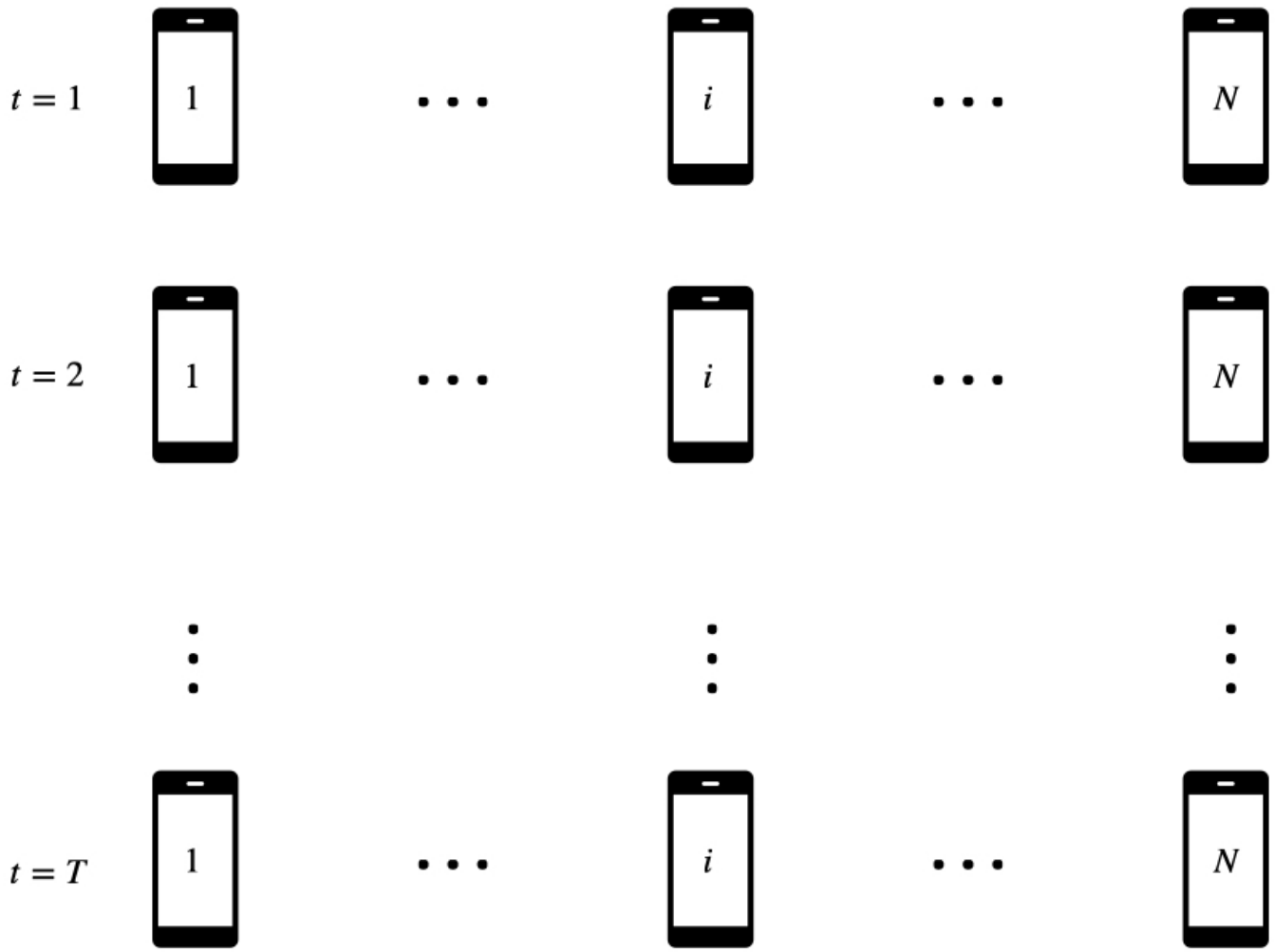
- We require a simple method to weigh the neural networks learnt by various nodes in the network
- Exponential weighted average is one such simple method which does the job
- It is in particular suitable for online learning scenario where the neural network needs to adapt itself quickly to the incoming data
- Though FL in online scenario is studied, most of these provide equal weight to the nodes in the network
- Fed-Exp is 'lighter' communication and computation-wise when compared to Omni-Fedge, our previous work

Our Work: Architecture

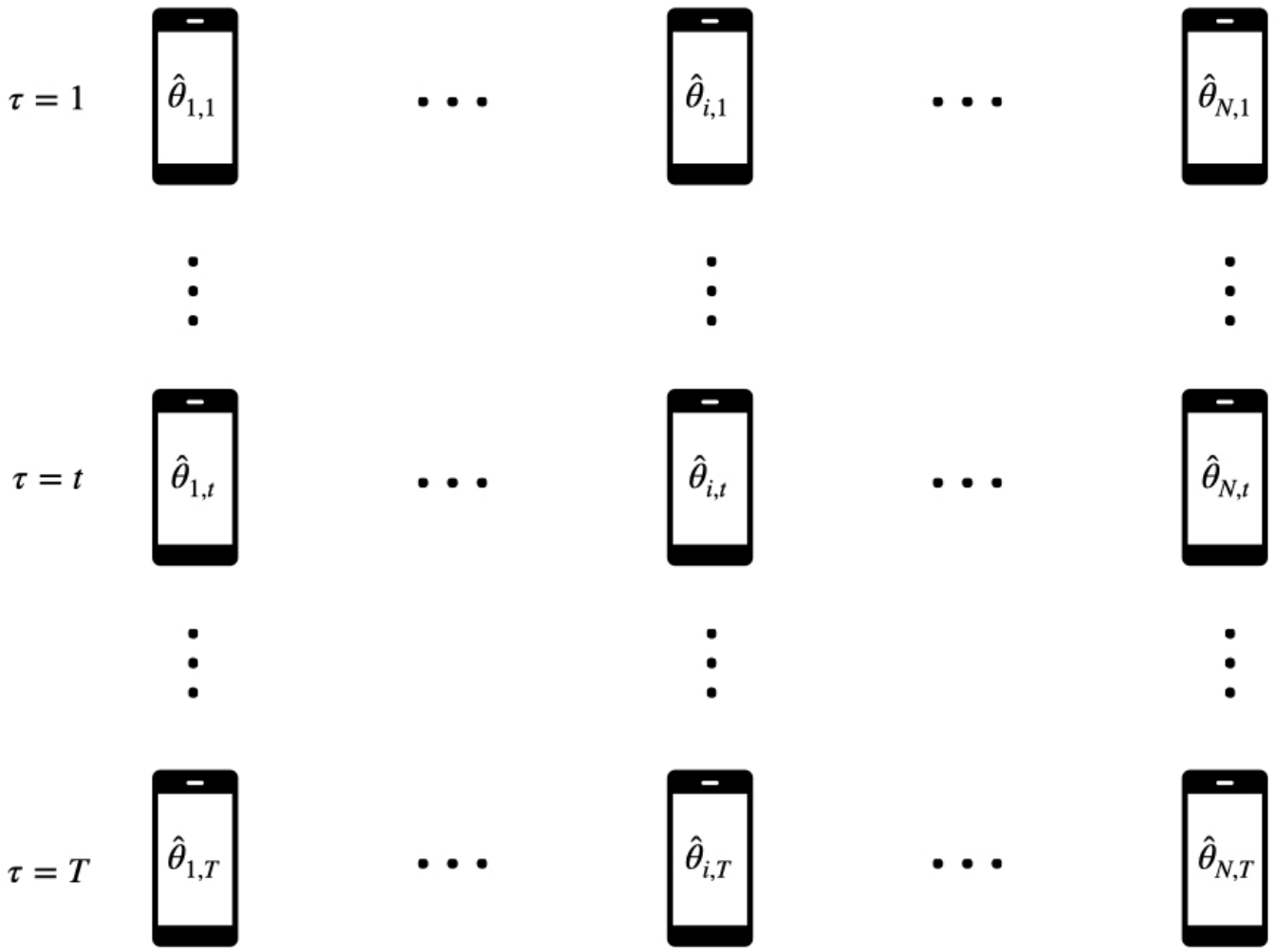


N edge-devices/nodes and one Federating Server (FS)

Our Work: Architecture



Our Work: Architecture



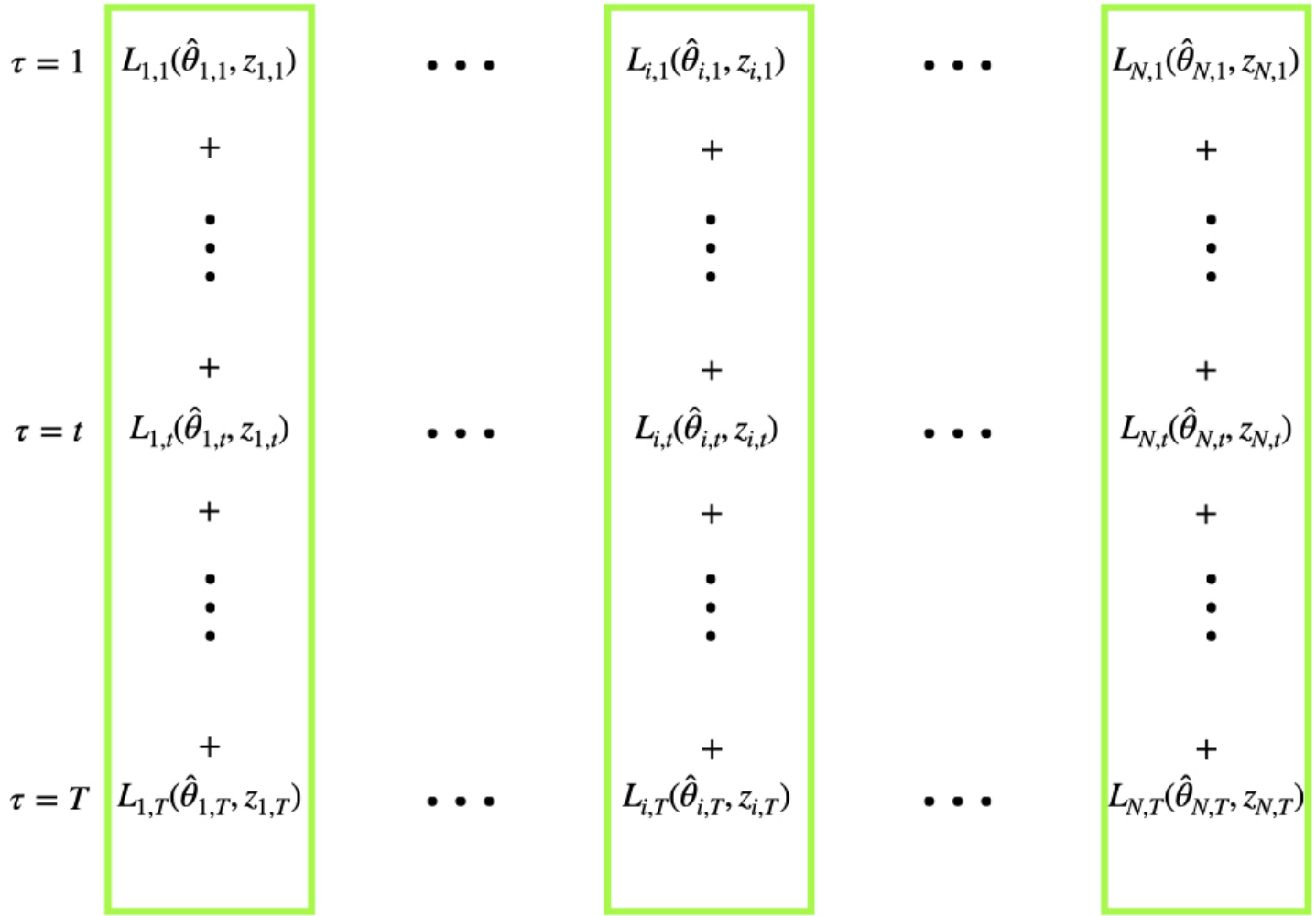
$\hat{\theta}_{i,t}$ denotes the neural network of i^{th} node at time t

Our Work: Architecture

$$
\begin{array}{lccccc}
\tau = 1 & L_{1,1}(\hat{\theta}_{1,1}, z_{1,1}) & \cdots & L_{i,1}(\hat{\theta}_{i,1}, z_{i,1}) & \cdots & L_{N,1}(\hat{\theta}_{N,1}, z_{N,1}) \\
 & + & & + & & + \\
 & \vdots & & \vdots & & \vdots \\
 & + & & + & & + \\
\tau = t & L_{1,t}(\hat{\theta}_{1,t}, z_{1,t}) & \cdots & L_{i,t}(\hat{\theta}_{i,t}, z_{i,t}) & \cdots & L_{N,t}(\hat{\theta}_{N,t}, z_{N,t}) \\
 & + & & + & & + \\
 & \vdots & & \vdots & & \vdots \\
 & + & & + & & + \\
\tau = T & L_{1,T}(\hat{\theta}_{1,T}, z_{1,T}) & \cdots & L_{i,T}(\hat{\theta}_{i,T}, z_{i,T}) & \cdots & L_{N,T}(\hat{\theta}_{N,T}, z_{N,T})
\end{array}
$$

$L_{i,t}(\hat{\theta}_{i,t}, z_{i,t})$ denotes the loss of i^{th} node at time t

Our Work: Architecture

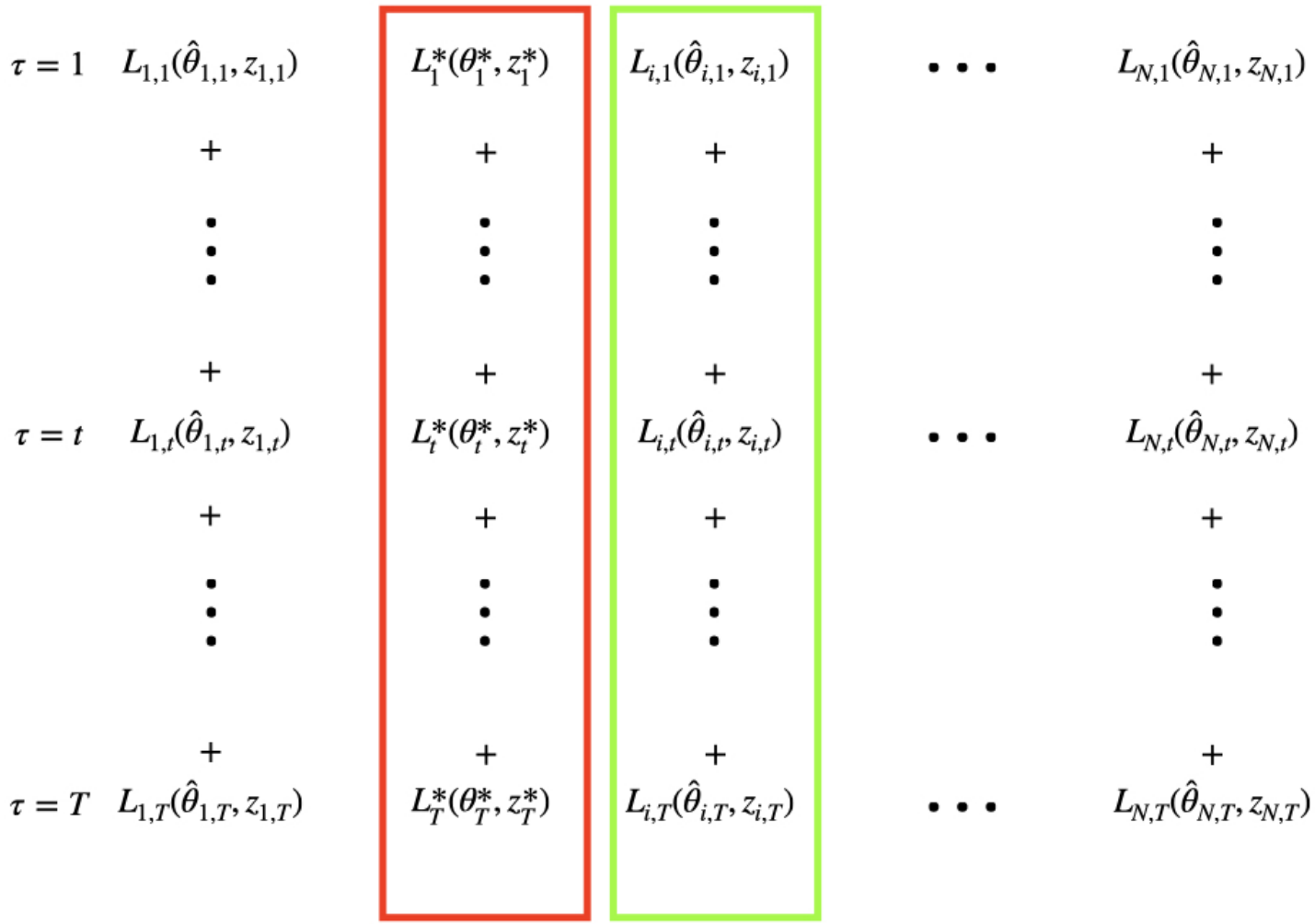


Our Work: Architecture

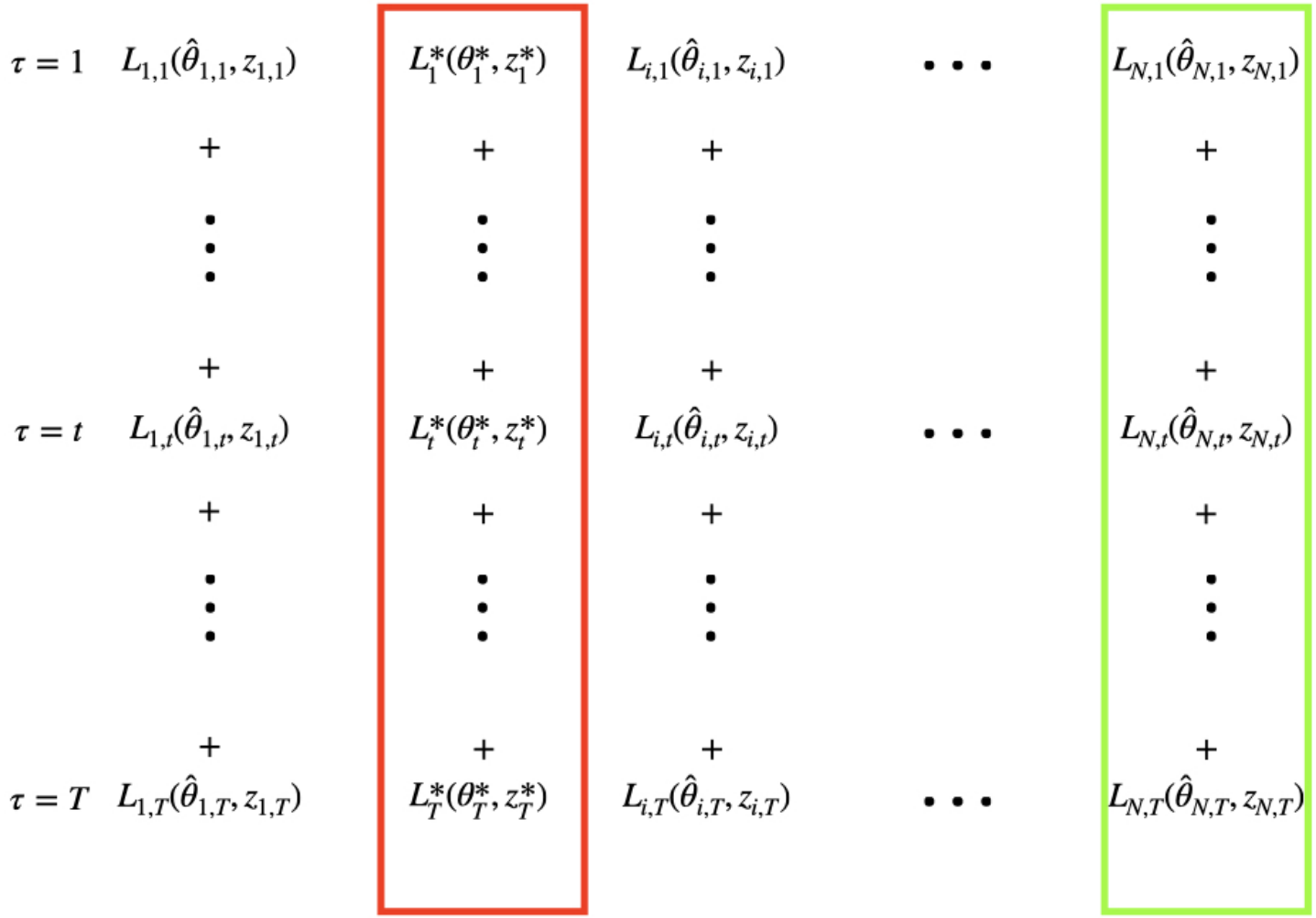
$\tau = 1$	$L_{1,1}(\hat{\theta}_{1,1}, z_{1,1})$	$L_1^*(\theta_1^*, z_1^*)$	$L_{i,1}(\hat{\theta}_{i,1}, z_{i,1})$	$\cdots$	$L_{N,1}(\hat{\theta}_{N,1}, z_{N,1})$
	$+$	$+$	$+$		$+$
	$\vdots$	$\vdots$	$\vdots$		$\vdots$
	$+$	$+$	$+$		$+$
$\tau = t$	$L_{1,t}(\hat{\theta}_{1,t}, z_{1,t})$	$L_t^*(\theta_t^*, z_t^*)$	$L_{i,t}(\hat{\theta}_{i,t}, z_{i,t})$	$\cdots$	$L_{N,t}(\hat{\theta}_{N,t}, z_{N,t})$
	$+$	$+$	$+$		$+$
	$\vdots$	$\vdots$	$\vdots$		$\vdots$
	$+$	$+$	$+$		$+$
$\tau = T$	$L_{1,T}(\hat{\theta}_{1,T}, z_{1,T})$	$L_T^*(\theta_T^*, z_T^*)$	$L_{i,T}(\hat{\theta}_{i,T}, z_{i,T})$	$\cdots$	$L_{N,T}(\hat{\theta}_{N,T}, z_{N,T})$

* denotes that node given by $arg \min_{i=1}^{N} \sum_{\tau=1}^{T} L_{i,t}(\hat{\theta}_{i,t}, z_{i,t})$

Our Work: Architecture



Our Work: Architecture



Our Work: Problem Setting

- There are N edge-devices/nodes and one Federating Server (FS).
- Let $\hat{\theta}_{i,t}$ denote the neural network of i^{th} node at time t and $\mathcal{L}_{i,t}(\hat{\theta}_{j,t}, z_{i,t})$ denote the loss of node i at time t using the neural network of node j at time t, where $z_{i,t}$ denotes the data (or a batch of data points) seen by node i at time t.
- Let $\omega_{ij,t}$ denote the weight given by node i to node j based upon the loss incurred by node i on using the neural network of node j.

Our Work: Problem Setting

- There are N edge-devices/nodes and one Federating Server (FS).
- Let $\hat{\theta}_{i,t}$ denote the neural network of i^{th} node at time t and $\mathcal{L}_{i,t}(\hat{\theta}_{j,t}, z_{i,t})$ denote the loss of node i at time t using the neural network of node j at time t, where $z_{i,t}$ denotes the data (or a batch of data points) seen by node i at time t.
- Let $\omega_{ij,t}$ denote the weight given by node i to node j based upon the loss incurred by node i on using the neural network of node j.

Our Work: Problem Setting

- There are N edge-devices/nodes and one Federating Server (FS).
- Let $\hat{\theta}_{i,t}$ denote the neural network of i^{th} node at time t and $\mathcal{L}_{i,t}(\hat{\theta}_{j,t}, z_{i,t})$ denote the loss of node i at time t using the neural network of node j at time t, where $z_{i,t}$ denotes the data (or a batch of data points) seen by node i at time t.
- Let $\omega_{ij,t}$ denote the weight given by node i to node j based upon the loss incurred by node i on using the neural network of node j.

Our Work: Algorithm

for $t \in \{1, \ldots, T\}$ **do**

- Update $\omega_{ij,t+1} \leftarrow \omega_{ij,t} e^{-\eta \mathcal{L}_{i,t}(\hat{\theta}_{j,t}, z_{i,t})}$, $\forall\ j \in \{1, \ldots, N\}$, where $\eta > 0$ is a hyper-parameter and $\omega_{ij,1} = 1\ \forall\ i, j \in \{1, \ldots, N\}$
- Compute $\Gamma_{i,t+1} = arg\ \min_{\Gamma} \mathcal{L}_{i,t+1}(\Gamma, z_{i,t+1})$ and broadcast to all nodes
- **if** $\left| \mathbb{E}_{\boldsymbol{p}_{i,t}}[(\hat{\theta}_{j,t} - \Gamma_{j,t})^T \nabla \mathcal{L}_{i,t}(\Gamma_{j,t}, z_{i,t})] \right| \leq \frac{c}{\sqrt{T}}$
  - Update $\hat{\theta}_{i,t+1} \leftarrow \frac{\sum_{j \in \mathcal{S}_{t+1}} \omega_{ij,t+1} \Gamma_{j,t+1}}{\sum_{j \in \mathcal{S}_{t+1}} \omega_{ij,t+1}}$ and broadcast to all nodes, where $\mathcal{S}_{t+1} \subseteq \{1, \ldots, N\}$ is randomly chosen and is of cardinality K
- **else**
  - Update $\hat{\theta}_{i,t+1} \leftarrow \Gamma_{j,t+1}$ and broadcast to all nodes, where $\mathcal{S}_{t+1} \subseteq \{1, \ldots, N\}$ is randomly chosen and is of cardinality K

Our Work: Theoretical Guarantees

Definition: Regret of Fed-Exp

$$\mathcal{R}_{i,T}^{Fed-Exp} = \mathbb{E}\Big[\sum_{t=1}^{T} \mathcal{L}_{i,t}(\hat{\theta}_{i,t}, z_{i,t})\Big] - \mathbb{E}[\min_{j\in\{1,\ldots,N\}} (L_{j,T})]$$

Our Work: Theoretical Guarantees

Theorem: Regret Bound of Fed-Exp

$$\mathbb{E}\Big[\sum_{t=1}^{T}\mathcal{L}_{i,t}(\hat{\theta}_{i,t},z_{i,t})\Big]-\mathbb{E}[\min_{j\in\{1,\ldots,N\}}(L_{j,T})]<$$

$$\frac{1}{\eta}\sum_{t=1}^{T}\log\left(\frac{(K-1)^2K}{N}e^{-\eta(L_{min,t}-L_{max,t})}+1\right)+\frac{\eta T}{8}$$

$$+cK\sqrt{T}+\frac{\log N}{\eta}+\left(1-\frac{K}{N}\right)L_T$$

Our Work: Theoretical Guarantees

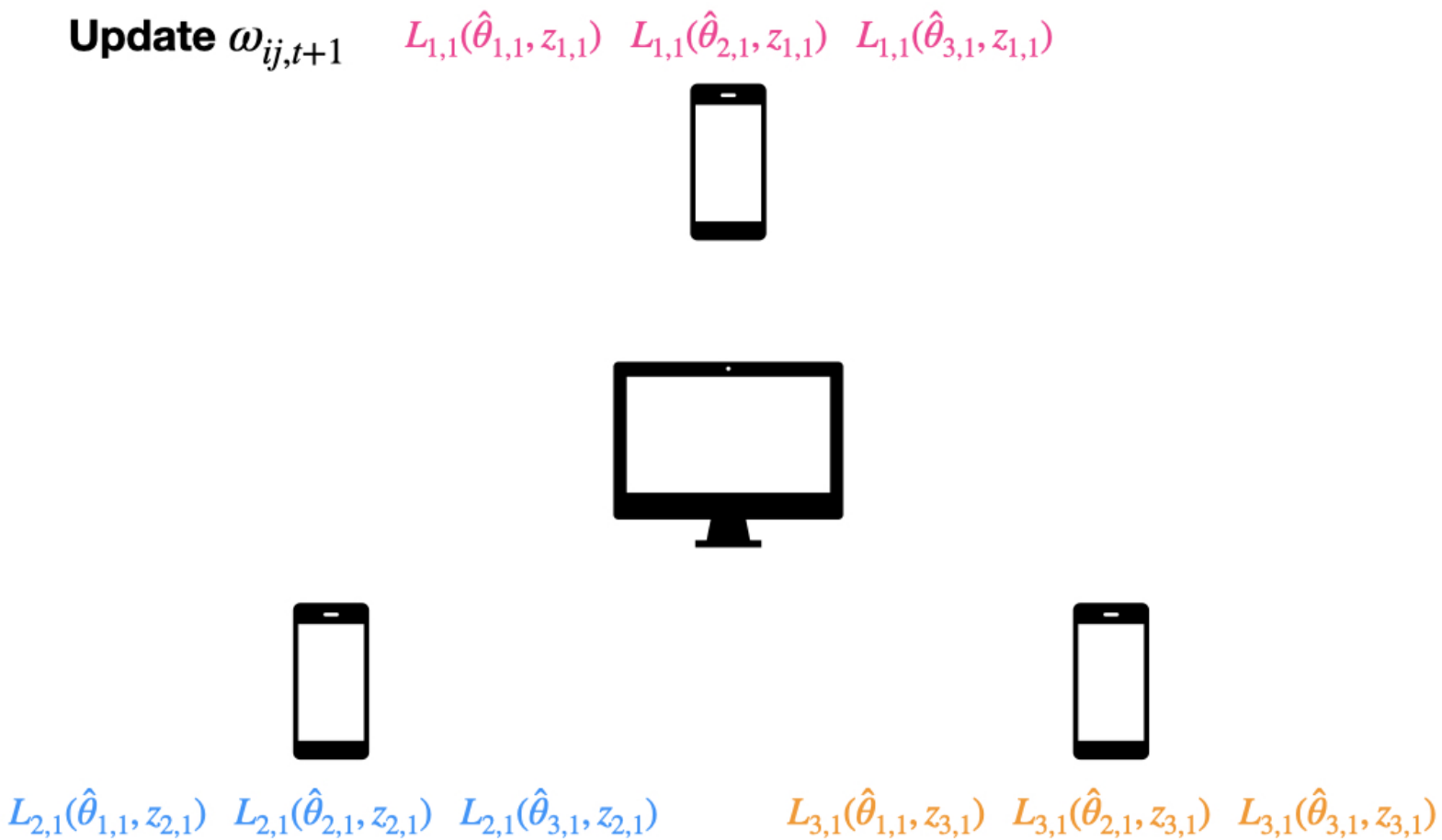
Theorem: Regret Bound of Fed-Exp

$$\mathbb{E}\Big[\sum_{t=1}^{T}\mathcal{L}_{i,t}(\hat{\theta}_{i,t},z_{i,t})\Big]-\mathbb{E}[\min_{j\in\{1,\ldots,N\}}(L_{j,T})]<$$

$$\frac{1}{\eta}\sum_{t=1}^{T}\log\left(\frac{(K-1)^2K}{N}e^{-\eta(L_{min,t}-L_{max,t})}+1\right)+\frac{\eta T}{8}$$

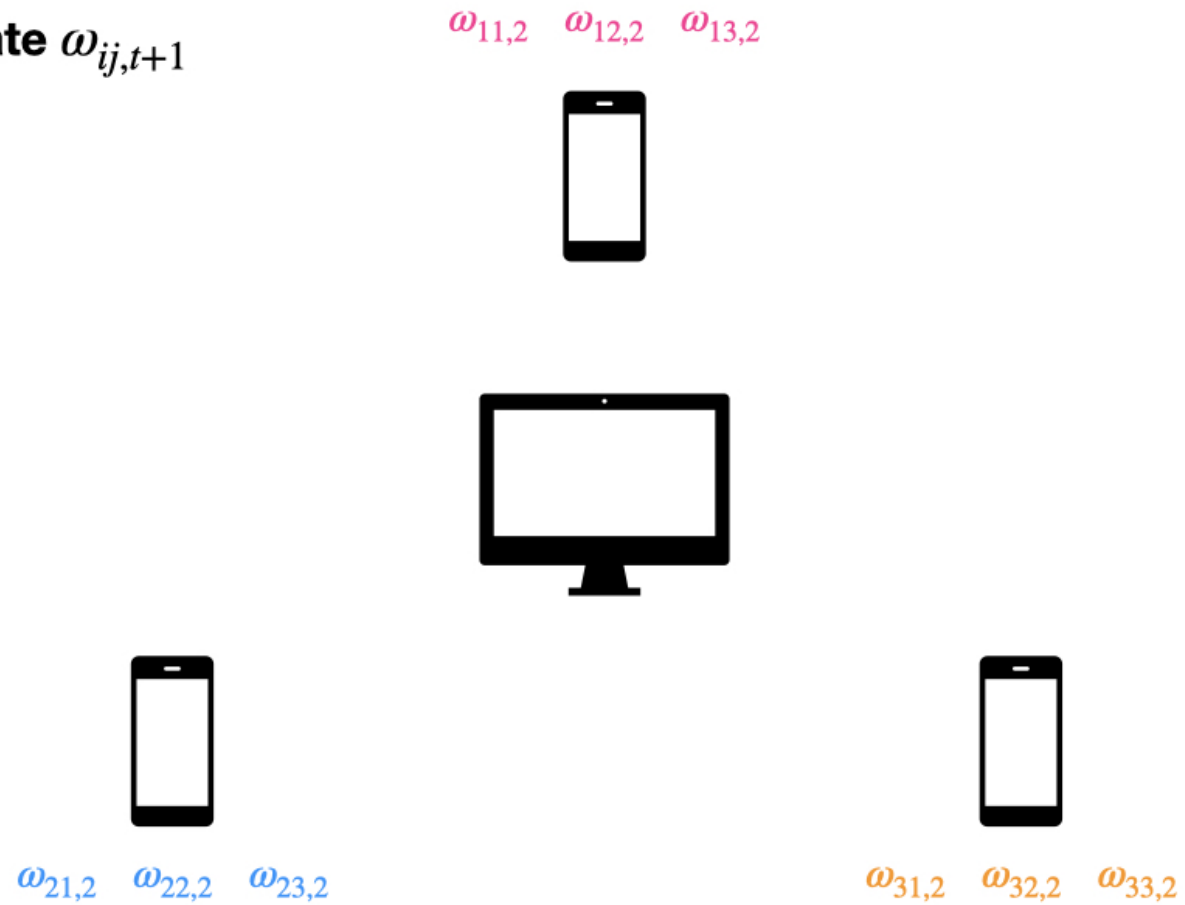
$$+cK\sqrt{T}+\frac{\log N}{\eta}+\left(1-\frac{K}{N}\right)L_T$$

Our Work: Algorithm (Explained)



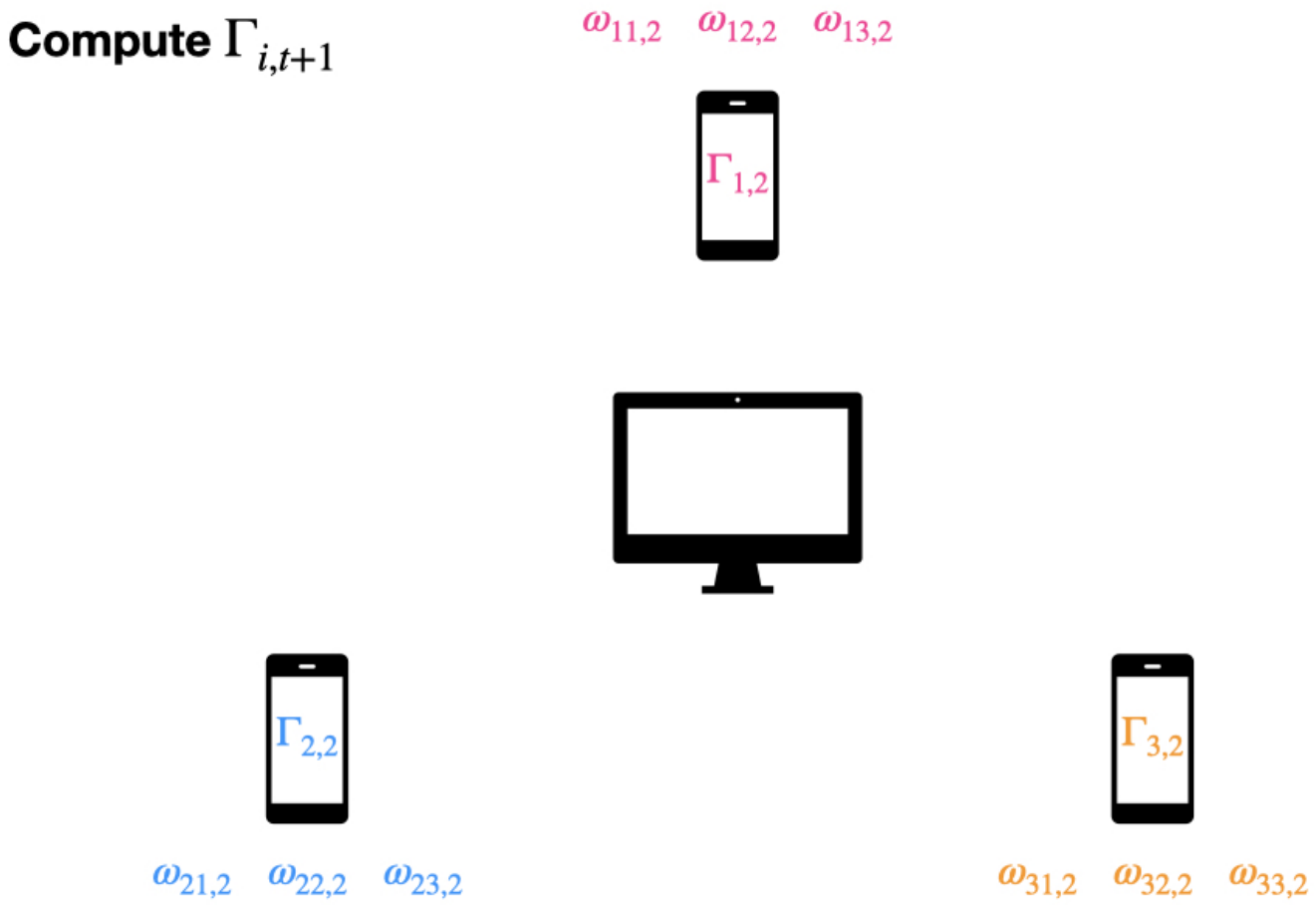
Our Work: Algorithm (Explained)

Update $\omega_{ij,t+1}$

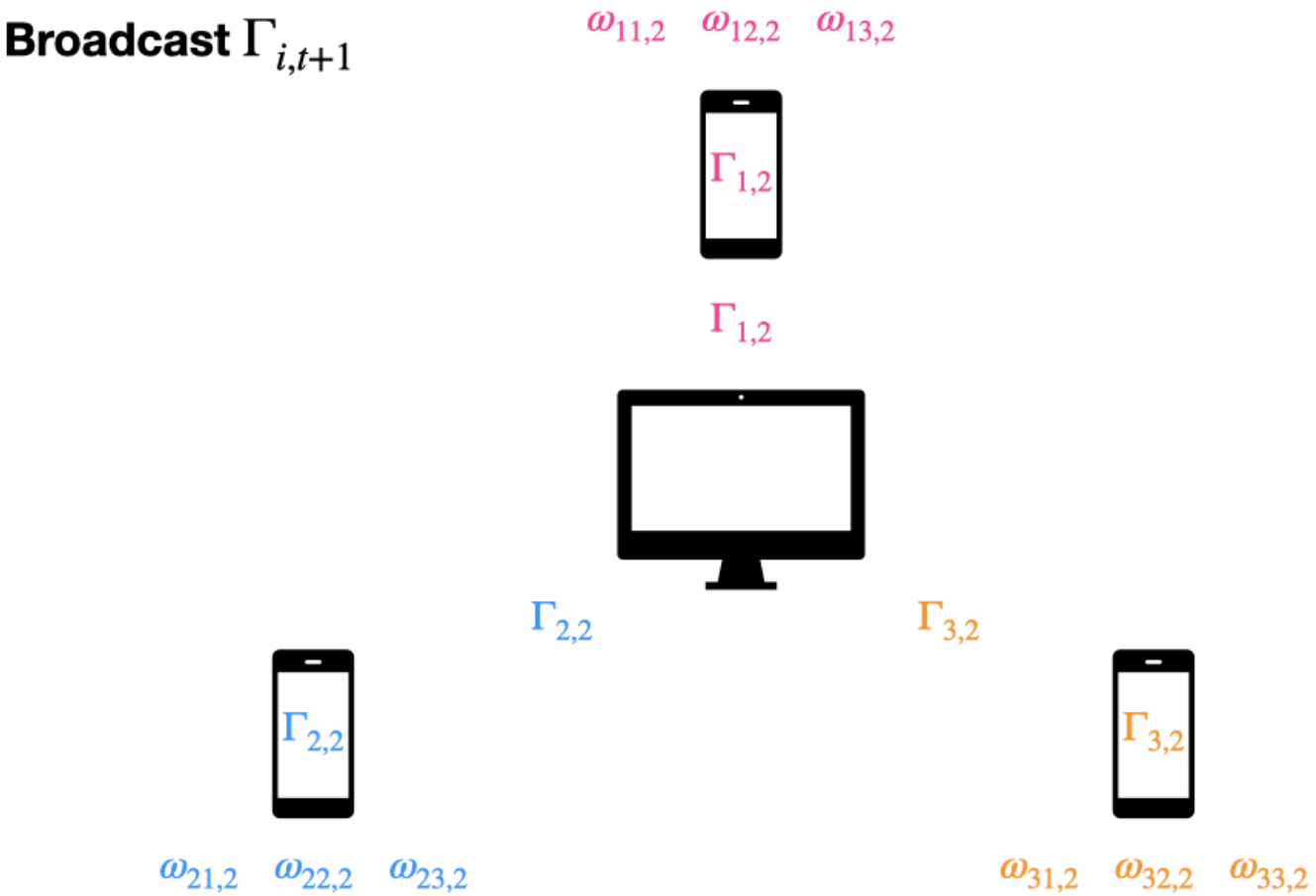


Our Work: Algorithm (Explained)

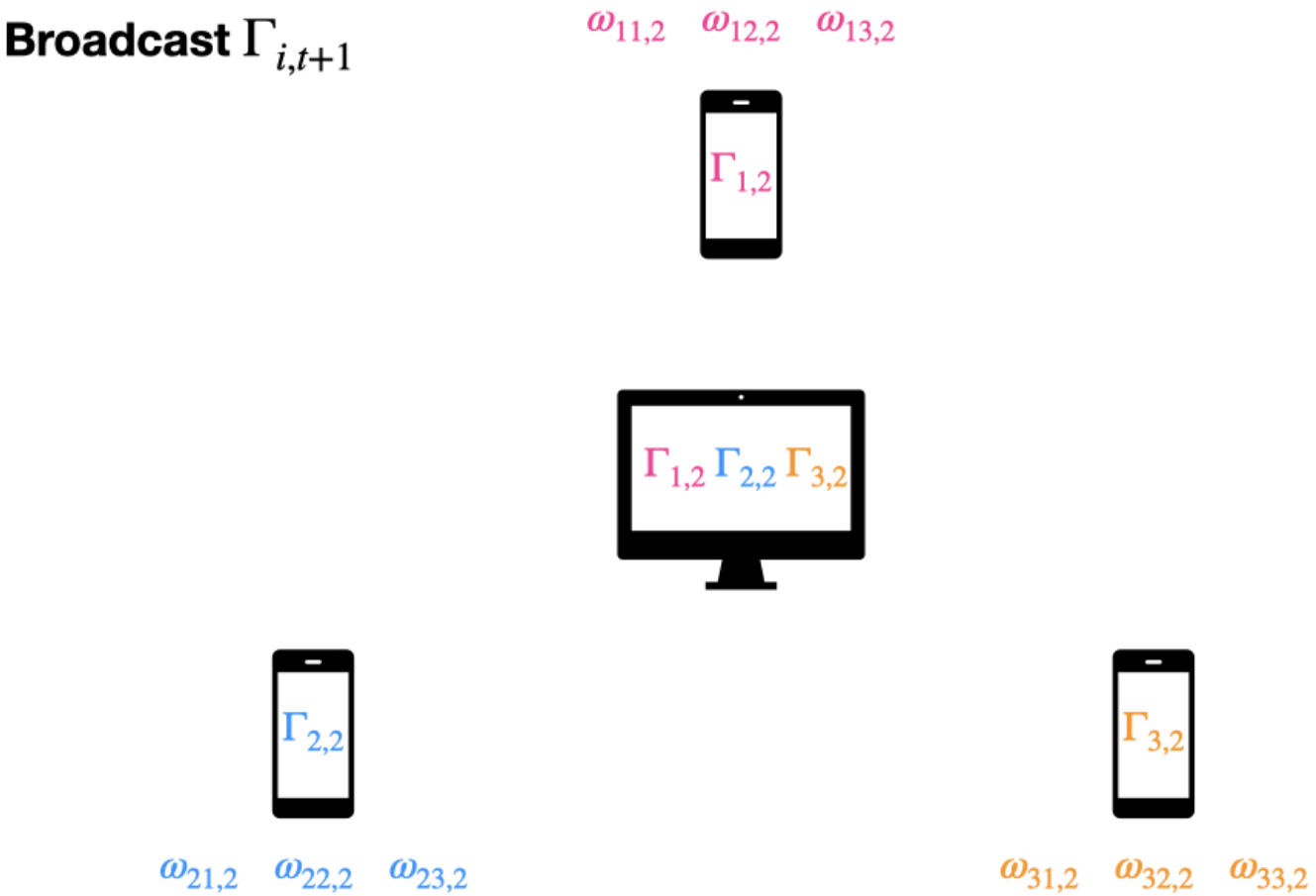
Compute $\Gamma_{i,t+1}$



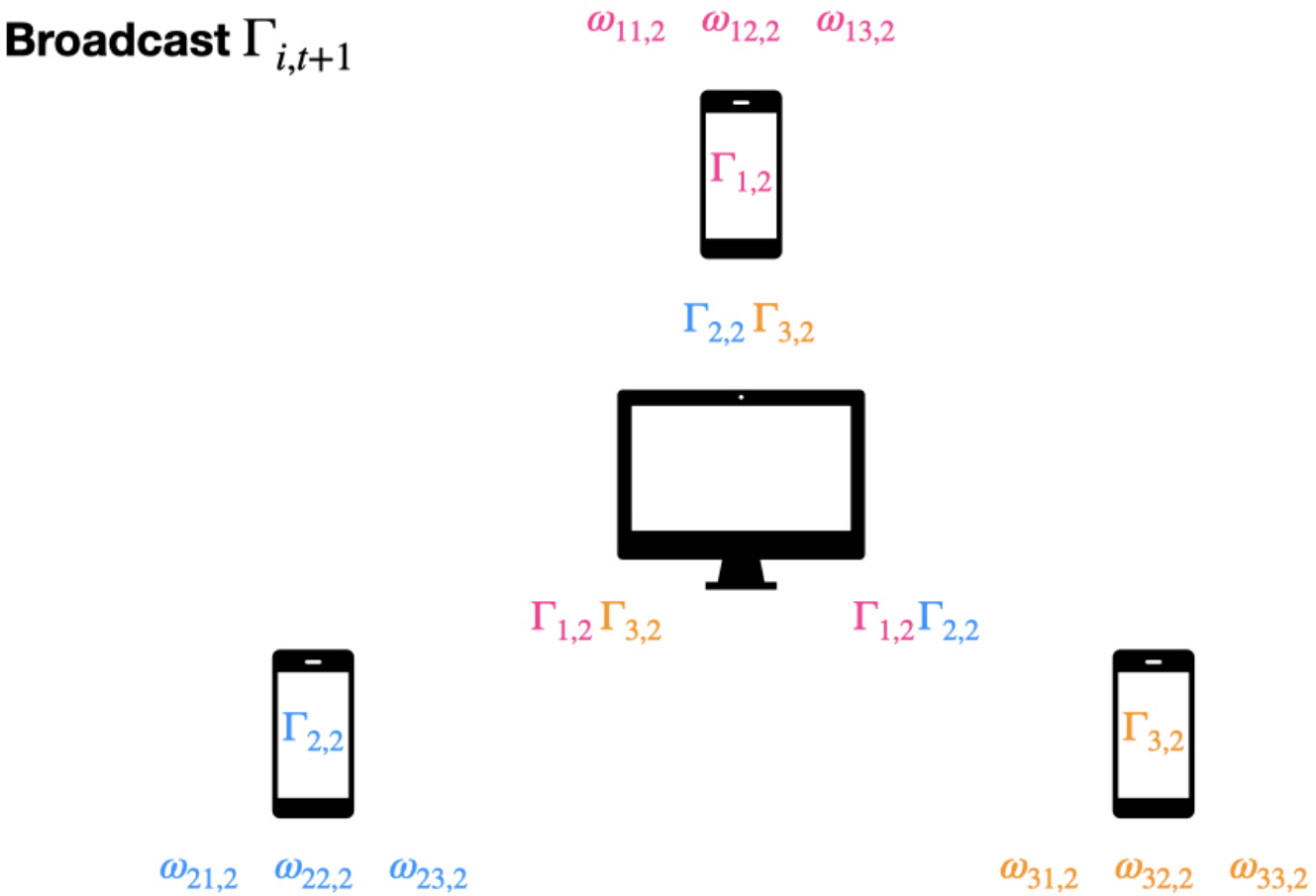
Our Work: Algorithm (Explained)



Our Work: Algorithm (Explained)

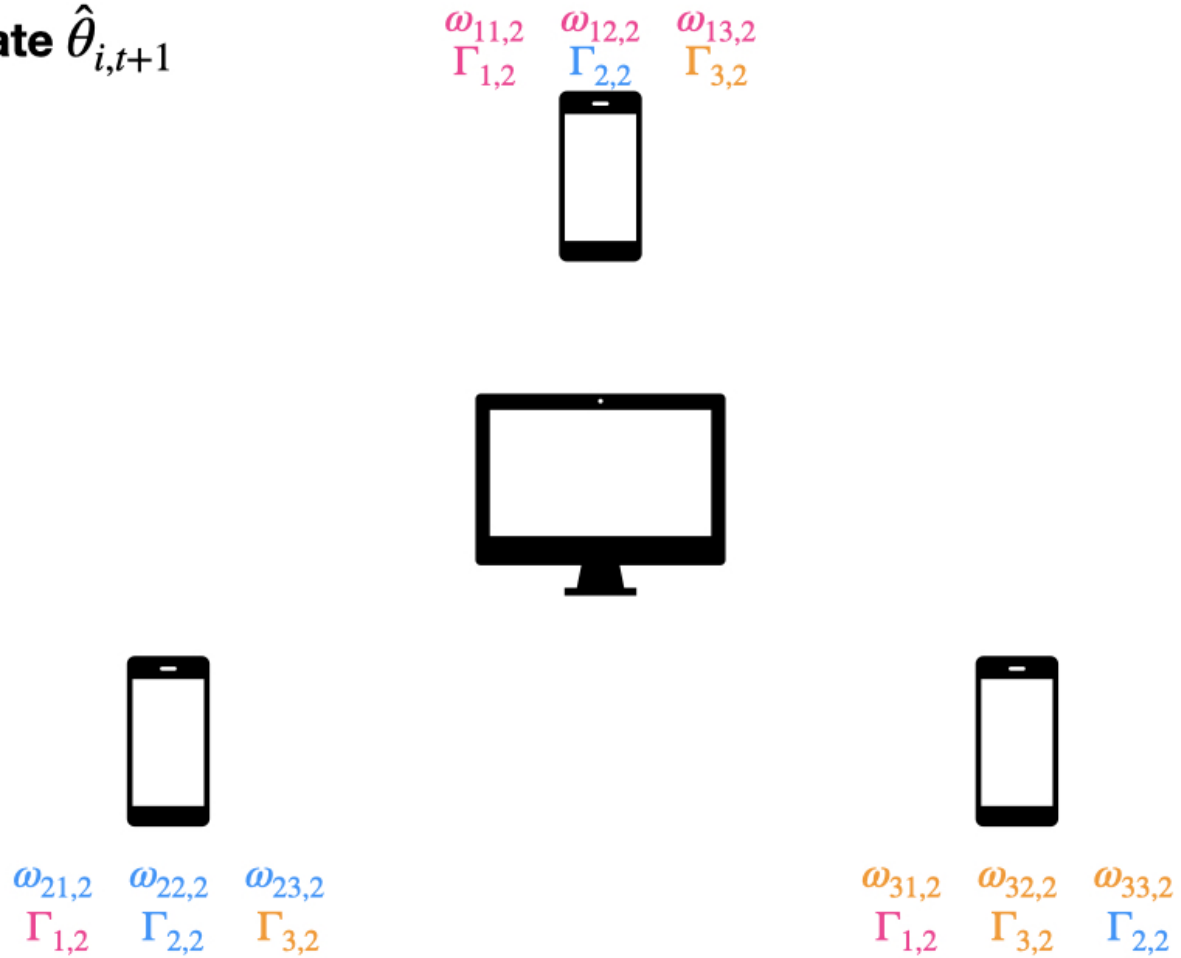


Our Work: Algorithm (Explained)



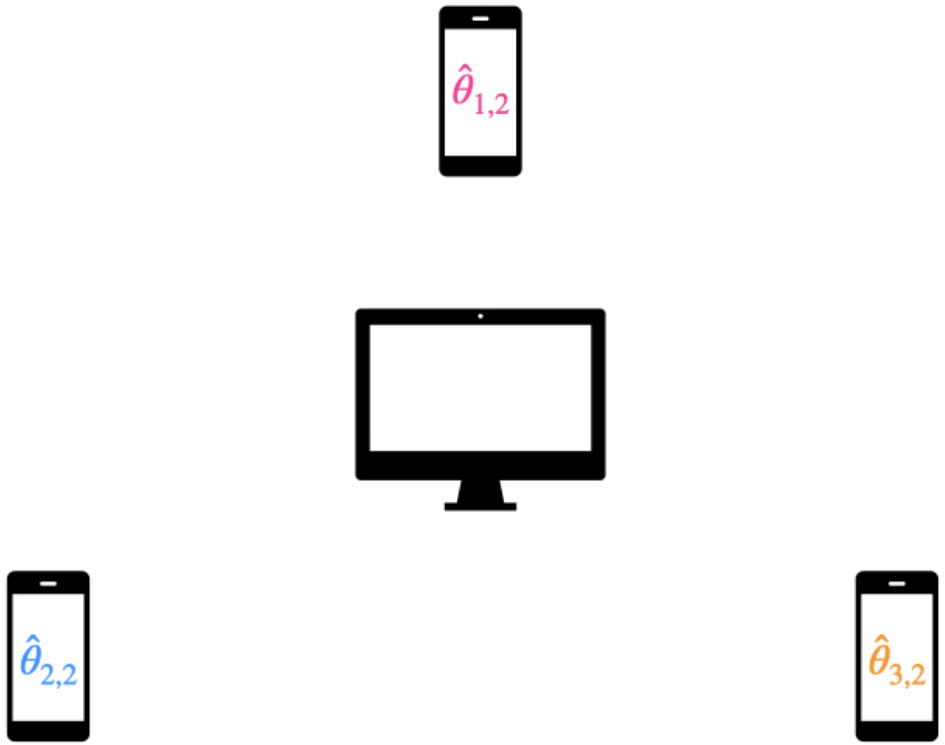
Our Work: Algorithm (Explained)

Update $\hat{\theta}_{i,t+1}$



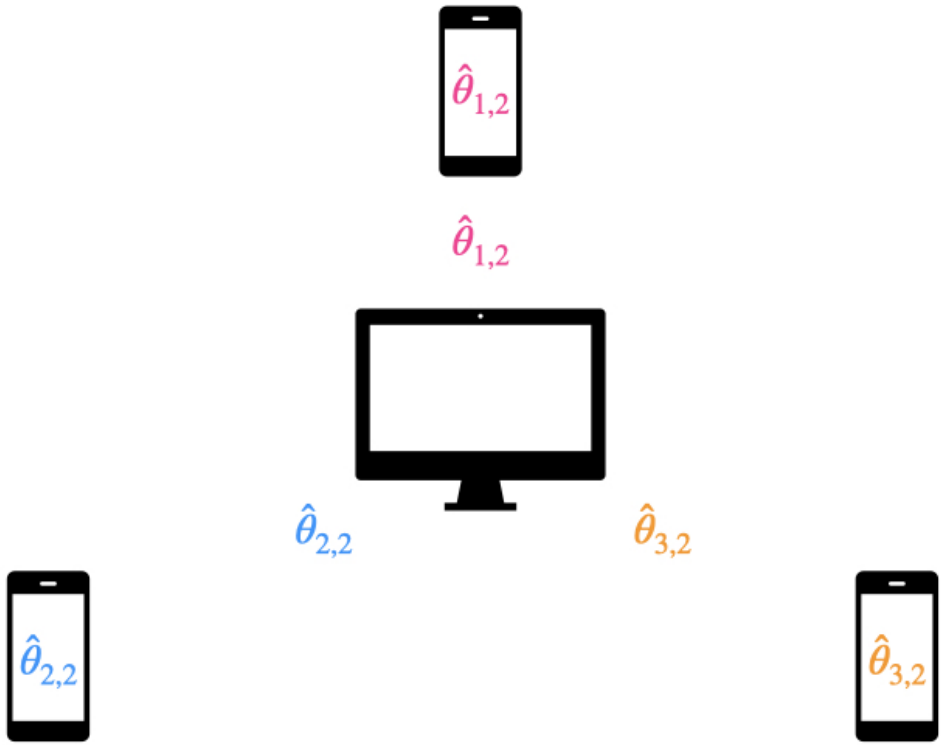
Our Work: Algorithm (Explained)

Update $\hat{\theta}_{i,t+1}$

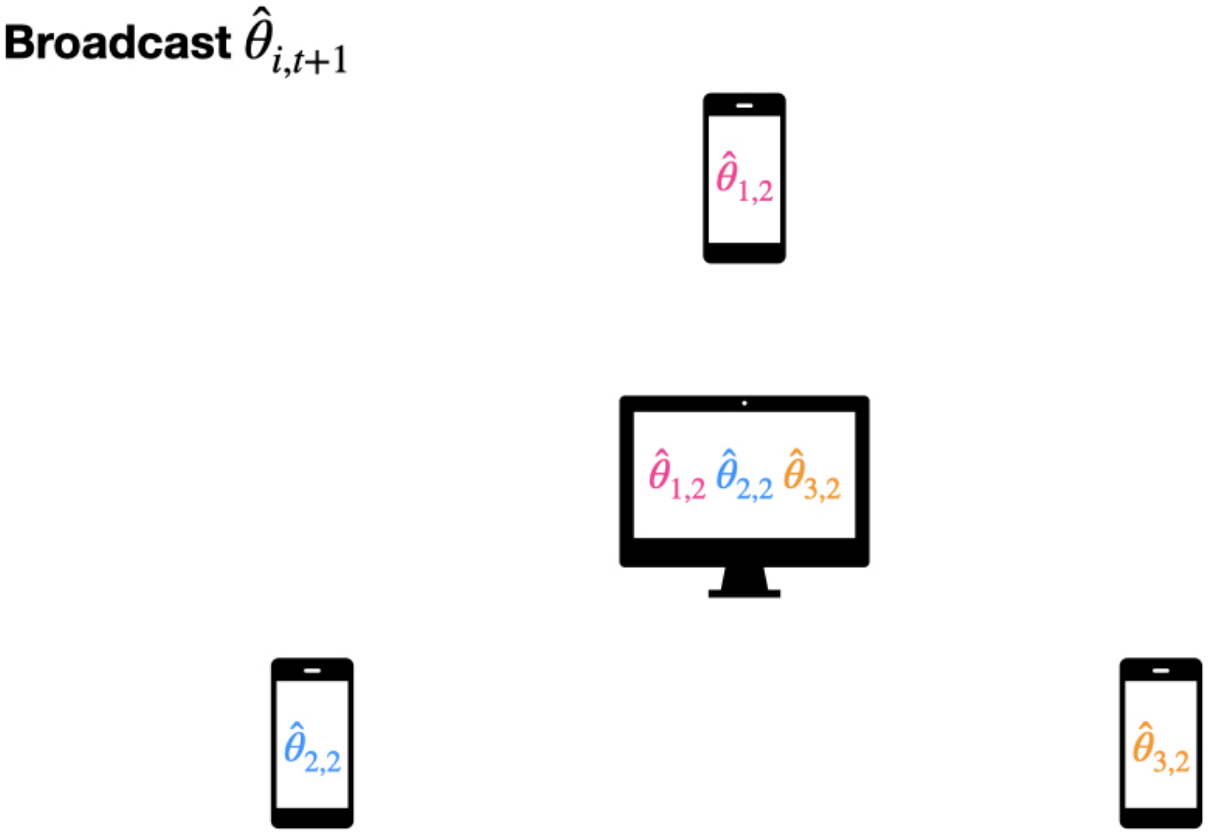


Our Work: Algorithm (Explained)

Broadcast $\hat{\theta}_{i,t+1}$

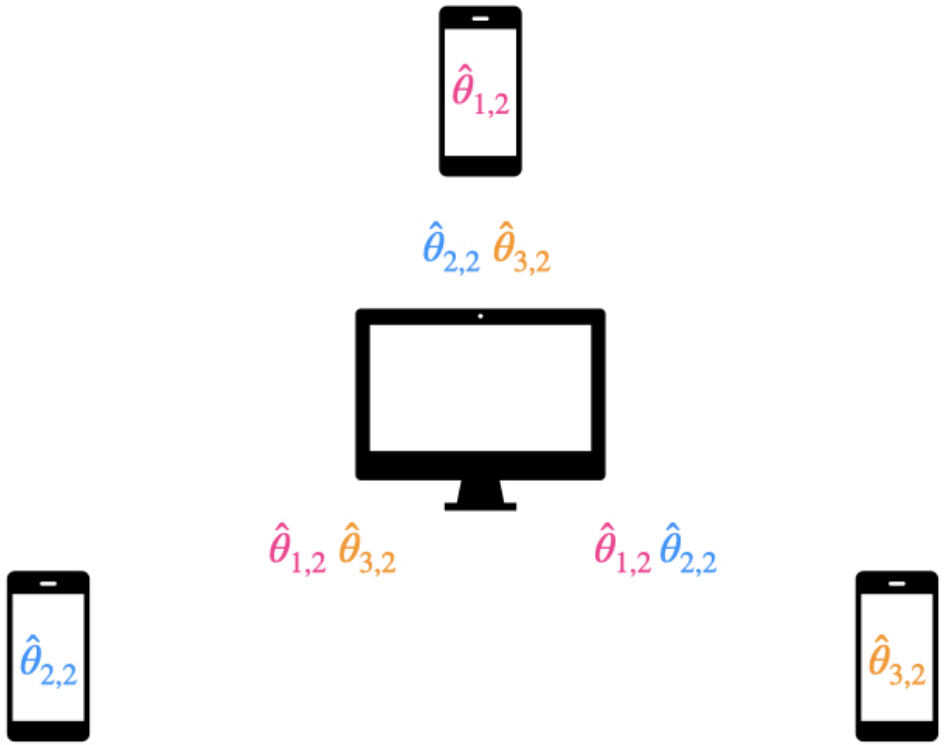


Our Work: Algorithm (Explained)



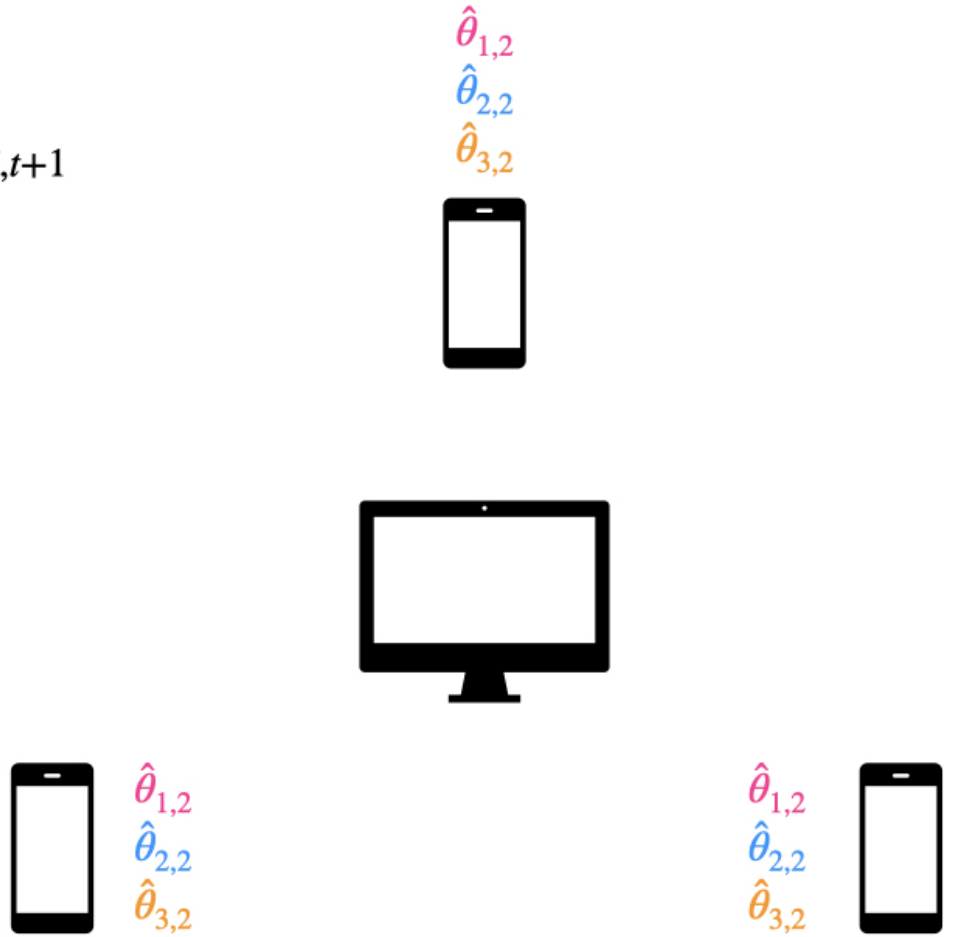
Our Work: Algorithm (Explained)

Broadcast $\hat{\theta}_{i,t+1}$



Our Work: Algorithm (Explained)

Update $\omega_{ij,t+1}$



Our Work: Algorithm (Explained)

Update $\omega_{ij,t+1}$

$L_{1,2}(\hat{\theta}_{1,2}, z_{1,2})$ $L_{1,2}(\hat{\theta}_{2,2}, z_{1,2})$ $L_{1,2}(\hat{\theta}_{3,2}, z_{1,2})$

$L_{2,2}(\hat{\theta}_{1,2}, z_{2,2})$ $L_{2,2}(\hat{\theta}_{2,2}, z_{2,2})$ $L_{2,2}(\hat{\theta}_{3,2}, z_{2,2})$

$L_{3,2}(\hat{\theta}_{1,2}, z_{3,2})$ $L_{3,2}(\hat{\theta}_{2,2}, z_{3,2})$ $L_{3,2}(\hat{\theta}_{3,2}, z_{3,2})$

Our Work: Experimental Results with Fed-Exp

Type of Data	Training samples per node	Testing samples per node	Number of batches	Number of training samples per iteration	Number of testing samples per iteration
MNIST i.i.d.	2488	3112	400	~6.22	~7.78
MNIST n.i.i.d.	799	1000	400	~2	~2.5
FMNIST i.i.d.	3111	3889	400	~7.77	~9.72
FMNIST n.i.i.d.	971	1178	400	~2.42	~2.95

Our Work: Experimental Results with Fed-Exp

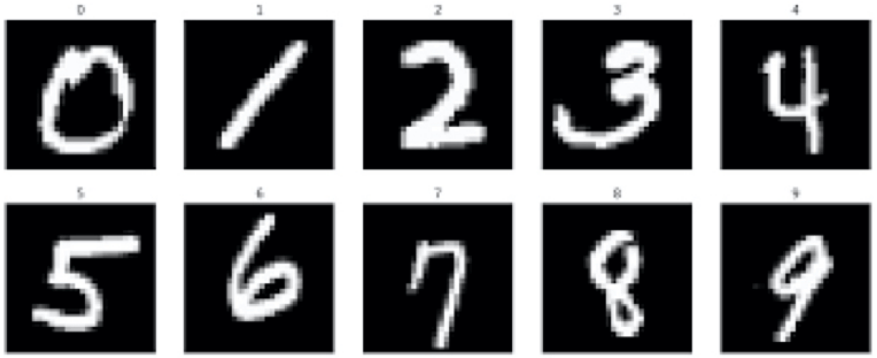


Figure: Sample MNIST and FMNIST images

Our Work: Experimental Results with Fed-Exp

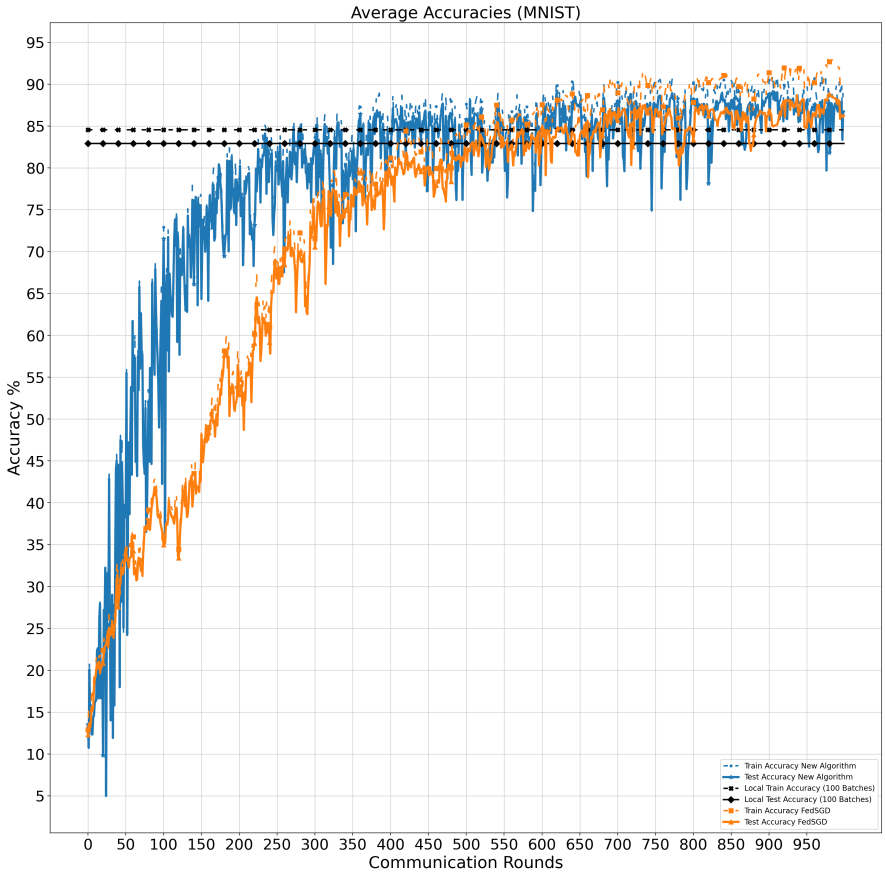


Figure: Results for non-i.i.d. data using MNIST

Our Work: Experimental Results with Fed-Exp

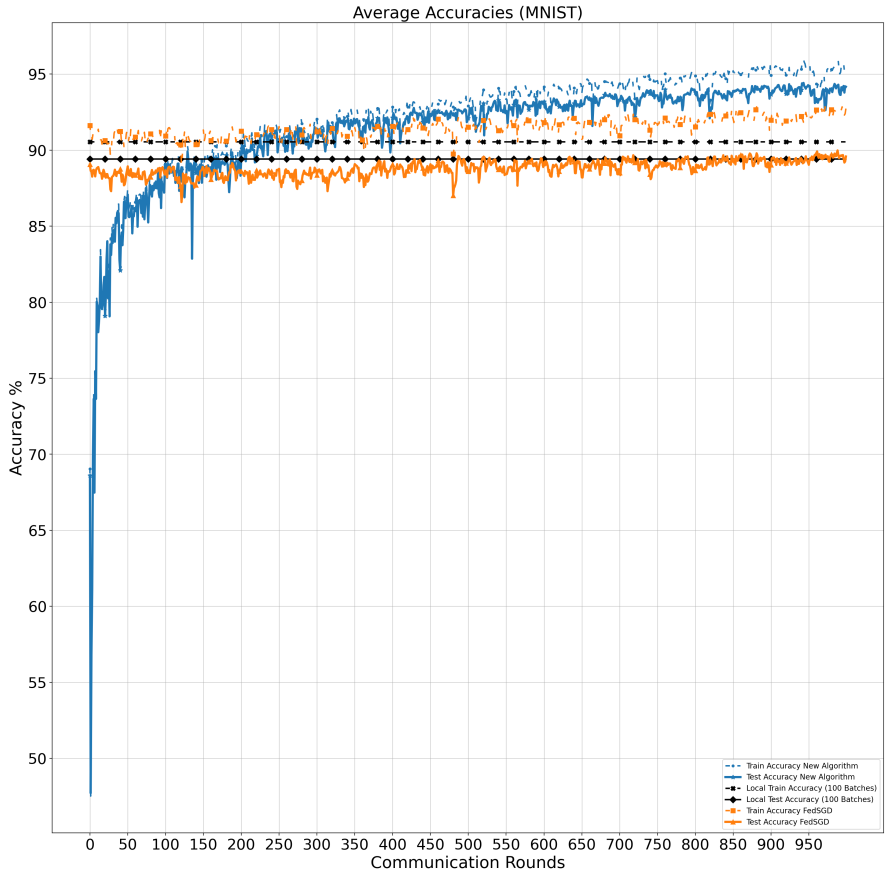


Figure: Results for i.i.d. data using MNIST

Our Work: Experimental Results with Fed-Exp

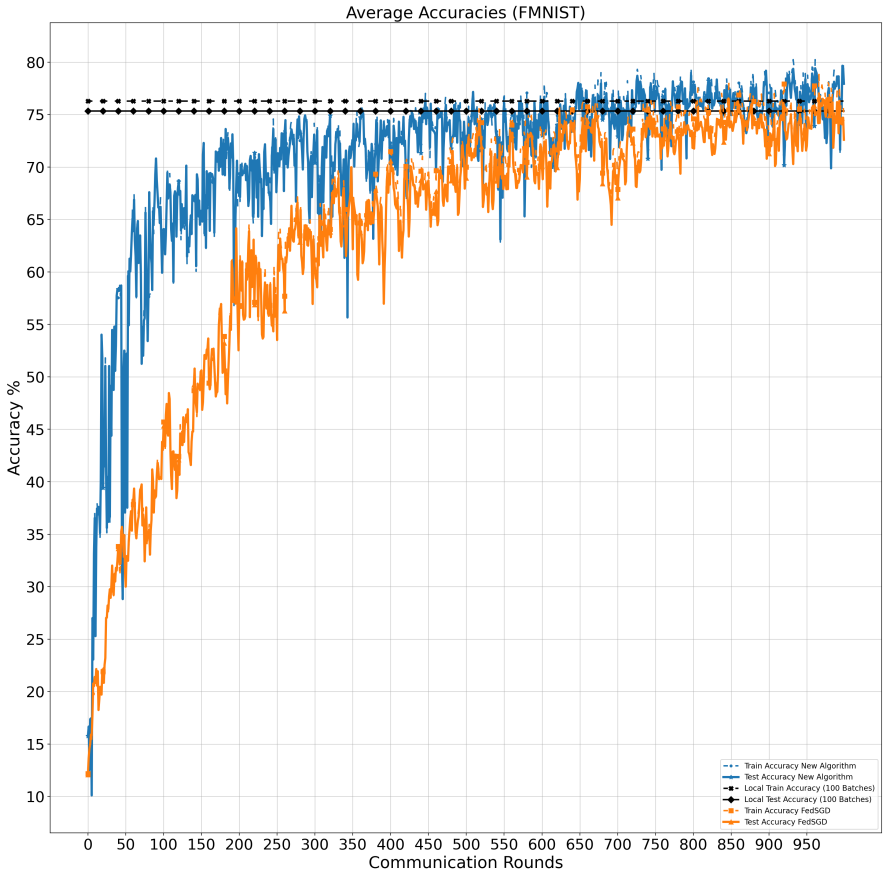


Figure: Results for non-i.i.d. data using FMNIST

Our Work: Experimental Results with Fed-Exp

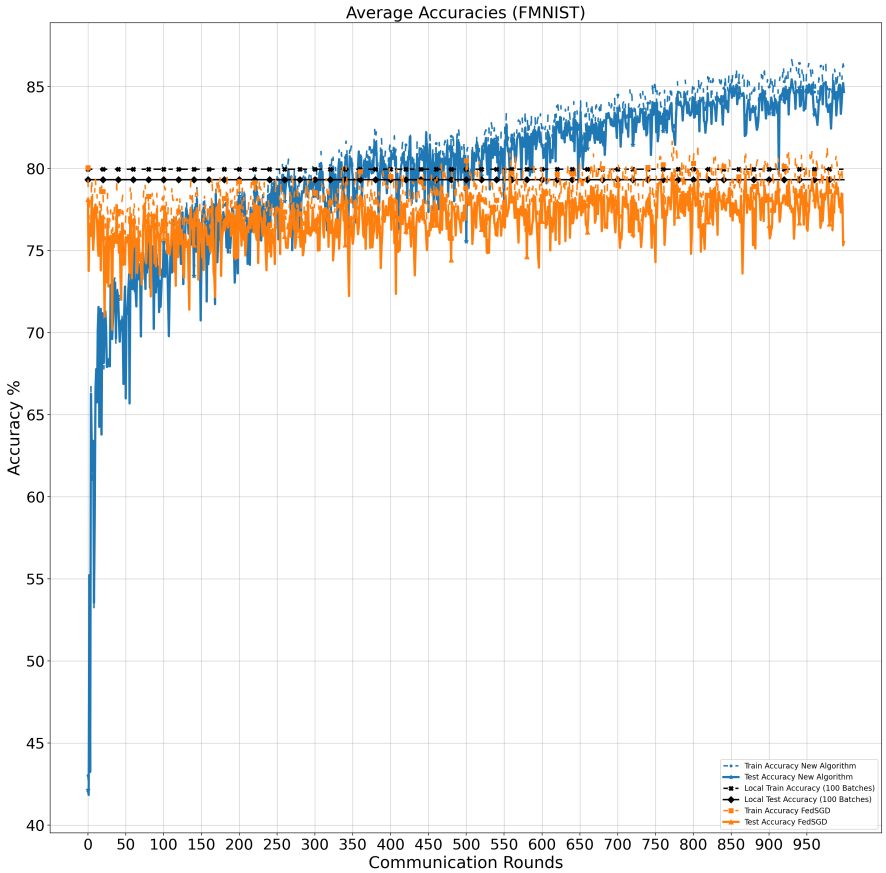


Figure: Results for i.i.d. data using FMNIST

Conclusion

- We proposed a federated algorithm in an online learning scenario and provided a sound theoretical framework on its regret bound.
- We tested the proposed algorithm using MNIST and FMNIST data-sets and observed that the algorithm performs considerably better than FedSGD and local training in both the cases.
- We propose to further improve the regret bound as the current one is not that desirable as we would expect the regret to go down as $\mathcal{O}(\sqrt{T})$. We also wish to study the performance of the proposed algorithm on other data-sets.
- Also, we envisage a large scale implementation of both of the algorithms that we have proposed.

Conclusion

- We proposed a federated algorithm in an online learning scenario and provided a sound theoretical framework on its regret bound.
- We tested the proposed algorithm using MNIST and FMNIST data-sets and observed that the algorithm performs considerably better than FedSGD and local training in both the cases.
- We propose to further improve the regret bound as the current one is not that desirable as we would expect the regret to go down as $\mathcal{O}(\sqrt{T})$. We also wish to study the performance of the proposed algorithm on other data-sets.
- Also, we envisage a large scale implementation of both of the algorithms that we have proposed.

Conclusion

- We proposed a federated algorithm in an online learning scenario and provided a sound theoretical framework on its regret bound.
- We tested the proposed algorithm using MNIST and FMNIST data-sets and observed that the algorithm performs considerably better than FedSGD and local training in both the cases.
- We propose to further improve the regret bound as the current one is not that desirable as we would expect the regret to go down as $\mathcal{O}(\sqrt{T})$. We also wish to study the performance of the proposed algorithm on other data-sets.
- Also, we envisage a large scale implementation of both of the algorithms that we have proposed.

Conclusion

- We proposed a federated algorithm in an online learning scenario and provided a sound theoretical framework on its regret bound.
- We tested the proposed algorithm using MNIST and FMNIST data-sets and observed that the algorithm performs considerably better than FedSGD and local training in both the cases.
- We propose to further improve the regret bound as the current one is not that desirable as we would expect the regret to go down as $\mathcal{O}(\sqrt{T})$. We also wish to study the performance of the proposed algorithm on other data-sets.
- Also, we envisage a large scale implementation of both of the algorithms that we have proposed.

Thank you!