Decentralized Federated Learning with Model Caching on Mobile Agents

Xiaoyu Wang[1], **Guojun Xiong**[2], **Houwei Cao**[3], **Jian Li**[2], **Yong Liu**[1]

[1]New York University
[2]Stony Brook University
[3]New York Institute of Technology
{wang.xiaoyu, yongliu}@nyu.edu, {guojun.xiong, jian.li.3}@stonybrook.edu, hcao02@nyit.edu

Abstract

Federated Learning (FL) trains a shared model using data and computation power on distributed agents coordinated by a central server. Decentralized FL (DFL) utilizes local model exchange and aggregation between agents to reduce the communication and computation overheads on the central server. However, when agents are mobile, the communication opportunity between agents can be sporadic, largely hindering the convergence and accuracy of DFL. In this paper, we propose **Cached Decentralized Federated Learning** (`Cached-DFL`) to investigate delay-tolerant model spreading and aggregation enabled by model caching on mobile agents. Each agent stores not only its own model, but also models of agents encountered in the recent past. When two agents meet, they exchange their own models as well as the cached models. Local model aggregation utilizes all models stored in the cache. We theoretically analyze the convergence of `Cached-DFL`, explicitly taking into account the model staleness introduced by caching. We design and compare different model caching algorithms for different DFL and mobility scenarios. We conduct detailed case studies in a vehicular network to systematically investigate the interplay between agent mobility, cache staleness, and model convergence. In our experiments, `Cached-DFL` converges quickly, and significantly outperforms DFL without caching.

Code —
https://github.com/ShawnXiaoyuWang/Cached-DFL
Extended version — https://arxiv.org/abs/2408.14001

Introduction

Federated Learning on Mobile Agents

Federated learning (FL) is a type of distributed machine learning (ML) that prioritizes data privacy (McMahan et al. 2017). The traditional FL involves a central server that connects with a large number of agents. The agents retain their data and do not share them with the server. During each communication round, the server sends the current global model to the agents, and a small subset of agents are chosen to update the global model by running stochastic gradient descent (SGD) (Robbins and Monro 1951) for multiple iterations on their local data. The central server then aggregates the updated parameters to obtain the new global model. FL naturally complements emerging Internet-of-Things (IoT) systems, where each IoT device not only can sense its surrounding environment to collect local data, but also is equipped with computation resources for local model training, and communication interfaces to interact with a central server for model aggregation. Many IoT devices are mobile, ranging from mobile phones, autonomous cars/drones, to self-navigating robots. In recent research efforts on smart connected vehicles, there has been a focus on integrating vehicle-to-everything (V2X) networks with Machine Learning (ML) tools and distributed decision making (Barbieri et al. 2022), particularly in the area of computer vision tasks such as traffic light and signal recognition, road condition sensing, intelligent obstacle avoidance, and intelligent road routing, etc. With FL, vehicles locally train deep ML models and upload the model parameters to the central server. This approach not only reduces bandwidth consumption, as the size of model parameters is much smaller than the size of raw image/video data, but also leverages computing power on vehicles, and protects user privacy.

However, FL on mobile agents still faces communication and computation challenges. The movements of mobile agents, especially at high speed, lead to fast-changing channel conditions on the wireless connections between mobile agents and the central server, resulting in high latency in FL (Niknam, Dhillon, and Reed 2020). Battery-powered mobile agents also have limited power budget for long-range wireless communications. Non-i.i.d data distributions on mobile agents make it difficult for local models to converge. As a result, FL on mobile agents to obtain an optimal global model remains an open challenge. Decentralized FL (DFL) has emerged as a potential solution where local model aggregations are conducted between neighboring mobile agents using local device-to-device (D2D) communications with high bandwidth, low latency and low power consumption (Martínez Beltrán et al. 2023). Preliminary studies have demonstrated that DFL algorithms have the potential to significantly reduce the high communication costs associated with centralized FL. However, blindly applying model aggregation algorithms, such as FedAvg (McMahan et al. 2017), developed for centralized FL to DFL cannot achieve fast convergence and high model accuracy (Liu et al. 2022).

Delay-Tolerant Model Communication and Aggregation Through Caching

D2D model communication between a pair of mobile agents is possible only if they are within each other's transmission ranges. If mobile agents only meet with each others sporadically, there will not be enough model aggregation opportunity for fast convergence. In addition, with non-i.i.d data distributions on agents, if an agent only meets with agents from a small cluster, there is no way for the agent to interact with models trained by data samples outside of its cluster, leading to disaggregated local models that cannot perform well on the global data distribution. It is therefore essential to achieve **fast and even** model spreading using limited D2D communication opportunities among mobile agents. A similar problem was studied in the context of Mobile Ad hoc Network (MANET), where wireless communication between mobile nodes are sporadic. The efficiency of data dissemination in MANET can be significantly improved by *Delay-Tolerant Networking (DTN)* (Fall 2003; Burleigh et al. 2003): a mobile node caches data it received from nodes it met in the past; when meeting with a new node, it not only transfers its own data, but also the cached data of other nodes. Essentially, node mobility forms a new "communication" channel through which cached data are transported through node movement in physical space. It is worth noting that, due to multi-hop caching-and-relay, DTN transmission incurs longer delay than D2D direct transmission. Data staleness can be controlled by caching and relay algorithms to match the target application's delay tolerance such as Li et al. (2023).

Motivated by DTN, we propose delay-tolerant DFL communication and aggregation enabled by model caching on mobile agents. To realize DTN-like model spreading, each mobile agent stores not only its own local model, but also local models received from other agents in the recent history. Whenever it meets another agent, it transfers its own model as well as the cached models to the agent through high-speed D2D communication. Local model aggregation on an agent works on all its cached models, mimicking a local parameter server. Compared with DFL without caching, DTN-like model spreading can push local models faster and more evenly to the whole network; aggregating all cached models can facilitate more balanced learning than pairwise model aggregation. While DFL model caching sounds promising, it also faces a new challenge of *model staleness*: a cached model from an agent is not the current model on that agent, with the staleness determined by the mobility patterns, as well as the model spreading and caching algorithms. Using stale models in model aggregation may slow down or even deviate model convergence.

The key challenge we want to address in this paper is how to design cached model spreading and aggregation algorithms to achieve fast convergence and high accuracy in DFL on mobile agents. Towards this goal, we make the following contributions:

1. We develop `Cached-DFL`, a new DFL framework that utilizes model caching on mobile agents to realize *delay-tolerant* model communication and aggregation;

Notation	Description
N	Number of agents
T	Number of global epochs
$[N]$	Set of integers $\{1, ..., N\}$
K	Number of local updates
$x_i(t)$	Model in the t^{th} epoch on agent i
$x(t)$	Global Model in the t^{th} epoch, $x(t) = \mathbb{E}_{i\in[N]}[x_i(t)]$
$x_i(t,k)$	Model initialized from x_t, after k-th local update on agent i
$\tilde{x}_i(t)$	Model $x_i(t)$ after local updates
$\mathcal{D}^i$	Dataset on the i-th agent
α	Aggregation weight
$t-\tau$	Staleness
τ_{max}	Tolerance of staleness in cache
$\|\|\cdot\|\|$	All the norms in the paper are l_2-norms

Table 1: Notations and Terminologies.

2. We theoretically analyze the convergence of aggregation with cached models, explicitly taking into account the model staleness;
3. We design and compare different model caching algorithms for different DFL and mobility scenarios.
4. We conduct a detailed case study on vehicular network to systematically investigate the interplay between agent mobility, cache staleness, and convergence of model aggregation. Our experimental results demonstrate that our `Cached-DFL` converges quickly and significantly outperforms DFL without caching.

Mobile DFL with Model Caching

Global Training Objective

Similar to the standard FL problem, the overall objective of mobile DFL is to learn a single global statistical model from data stored on tens to potentially millions of mobile agents. The overall goal is to find the optimal model weights $x^* \in \mathbb{R}^d$ to minimize the global loss function:

$$\min_x F(x), \text{ where } F(x) = \frac{1}{N}\sum_{i\in[N]} \mathbb{E}_{z^i\sim\mathcal{D}^i} f(x; z^i), \quad (1)$$

where N denotes the total number of mobile agents, and each agent has its own local dataset, i.e., $\mathcal{D}^i \neq \mathcal{D}^j, \forall i \neq j$. And z^i is sampled from the local data $\mathcal{D}^i$.

DFL Training with Local Model Caching

All agents participate in DFL training over T global epochs. At the beginning of the t^{th} epoch, agent i's local model is $x_i(t)$. After K steps of SGD to solve the following optimization problem with a regularized loss function:

$$\min_x \mathbb{E}_{z^i\sim D^i} f(x; z^i) + \frac{\rho}{2}||x - x_i(t)||^2,$$

agent i obtains an updated local model $\tilde{x}_i(t)$. Meanwhile, during the t^{th} epoch, driven by their mobility patterns, each

Algorithm 1: **Cached Decentralized Federated Learning** (`Cached-DFL`)

Input: Global epochs T, local updates K, initial models $\{x_i(0)\}_{i=1}^N$, staleness tolerance $\tau_{\max}$

```
1: function LocalUpdate($x_i(t)$)
2:     Initialize: $x_i(t,0) = x_i(t)$
3:     Define: $g_{x(t)}(x;z) = f(x;z) + \frac{\rho}{2}\|x - x(t)\|^2$
4:     for $k = 1, 2, \dots, K$ do
5:         Randomly sample $z_k^i \sim \mathcal{D}^i$
6:         $x_i(t,k) = x_i(t,k-1) - \eta \nabla g_{x_i(t)}(x_i(t,k-1); z_k^i)$
7:     end for
8:     return $\tilde{x}_i(t) = x_i(t,K)$
9: end function

10: function ModelAggregation($\mathcal{C}_i(t)$)
11:     $x_i(t+1) = \sum_{j\in\mathcal{C}_i(t)} \alpha_j \tilde{x}_j(\tau)$
12:     return $x_i(t+1)$
13: end function
    Main Process:
14: for $t = 0, 1, \dots, T-1$ do
15:     for $i = 1, 2, \dots, N$ do
16:         $\tilde{x}_i(t) \leftarrow$ LocalUpdate($x_i(t)$)
17:         $\mathcal{C}_i(t) \leftarrow$ CacheUpdate($\mathcal{C}_i(t-1), \tau_{\max}$)
18:         $x_i(t+1) \leftarrow$ ModelAggregation($\mathcal{C}_i(t)$)
19:     end for
20: end for
```

Output: $\{x_i(T)\}_{i=1}^N$

agent meets and exchanges models with other agents. Other than its own model, agent i also stores models it received from other agents encountered in the recent history in its local cache $\mathcal{C}_i(t)$. When two agents meet, they not only exchange their own local models, but also share their cached models with each others to maximize the efficiency of DTN-like model spreading. The models received by agent i will be used to update its model cache $\mathcal{C}_i(t)$, using different cache update algorithms, such as LRU update method (Algorithm 2) or Group-based LRU update method, which will be described in details later. As the cache size of each agent is limited, it is important to design an efficient cache update rule in order to maximize the caching benefit.

After cache updating, each agent conducts local model aggregation using all the cached models with customized aggregation weights $\{\alpha_j \in (0,1)\}$ to get the updated local model $x_i(t+1)$ for epoch $t+1$. In our simulation, we take the aggregation weight as $\alpha_j = (n_j / \sum_{j\in\mathcal{C}_i(t)} n_j)$, where n_j is the number of samples on agent j.

The whole process repeats until the end of T global epochs. The detailed algorithm is shown in Algorithm 1. z_k^i are randomly drawn local data samples on agent i for the k-th local update, and η is the learning rate.

Remark 1. *Note the N agents communicate with each others in a mobile D2D network. D2D communication can only happen between an agent and its neighbors within a short range (e.g. several hundred meters). Since agent locations are constantly changing (for instance, vehicles continuously move along the road network of a city), D2D network topology is dynamic and can be sparse at any given epoch. To ensure the eventual model convergence, the union graph of D2D networks over multiple epochs should be strongly-connected for efficient DTN-like model spreading. We also assume D2D communications are non-blocking, carried by short-distance high-throughput communication methods such as mmWave or WiGig, which has enough capacity to complete the exchange of cached models before the agents go out of each other's communication ranges.*

Remark 2. *Intuitively, comparing to the DFL without cache (e.g. DeFedAvg (Sun, Li, and Wang 2022)), where each agent can only get new model by averaging with another model, `Cached-DFL` uses more models (delayed versions) for aggregation, thus utilizes more underlying information from datasets on more agents. Although `Cached-DFL` introduces stale models, it can benefit model convergence, especially in highly heterogeneous data distribution scenarios. Overall, our `Cached-DFL` framework allows each agent to act as a local proxy for Centralized FL with delayed cached models, thus speedup the convergence especially in highly heterogeneous data distribution scenarios, that are challenging for the traditional DFL to converge.*

Remark 3. *As mentioned above, our approach inevitably introduces stale models. Intuitively, larger staleness results in greater error in the global model. For the cached models with large staleness $t-\tau$, we could set a threshold τ_{max} to kick old models out of model spreading, which is described in the cache update algorithms. The practical value for τ_{max} should be related to the cache capacity and communication frequency between agents. In our experimental results, we choose τ_{max} to be 10 or 20 epochs. Results and analysis about the effect of different τ_{max} can be found in experimental results section.*

Convergence Analysis

We now theoretically investigate the impact of caching, especially the staleness of cached models, on DFL model convergence. We introduce some definitions and assumptions.

Definition 1 (Smoothness)**.** *A differentiable function f is L-smooth if for $\forall x, y$, $f(y) - f(x) \le \langle \nabla f(x), y - x\rangle + \frac{L}{2}||y - x||^2$, where $L > 0$.*

Definition 2 (Bounded Variance)**.** *There exists constant $\varsigma > 0$ such that the global variability of the local gradient of the loss function is bounded $||\nabla F_j(x) - \nabla F(x)||^2 \le \varsigma^2, \forall j \in [N], x \in \mathbb{R}^d$.*

Theorem 1. *Assume that F is L-smooth and convex, and each agent executes K local updates before meeting and exchanging models, after that, then does model aggregation. We also assume bounded staleness $\tau < \tau_{max}$, as the kick-out threshold. Furthermore, we assume, $\forall x \in \mathbb{R}^d, i \in [N]$, and $\forall z \sim \mathcal{D}^i, ||\nabla f(x;z)||^2 \le V, ||\nabla g_{x'}(x;z)||^2 \le V, \forall x' \in \mathbb{R}^d$. For any small constant $\epsilon > 0$, if we take $\rho > 0$, and satisfying $-(1+2\rho+\epsilon)V + (\rho^2 - \frac{\rho}{2})||x(t,k-1) - x(t)||^2 \ge$*

$0, \forall x(t,k-1), x(t)$, *after* T *global epochs,* `Cached-DFL` *converges to a critical point:*

$$\min_{t=0}^{T-1} \mathbb{E}||\nabla F(x(t))||^2 \leq \frac{\tau_{max}\mathbb{E}[F(x(0)) - F(x_{M(T)}(T))]}{\epsilon\eta C_1 KT}$$
$$+ \mathcal{O}(\frac{\eta\rho K^2}{\epsilon C_1}) \leq \mathcal{O}(\frac{\tau_{max}}{\epsilon\eta C_1 KT}) + \mathcal{O}(\frac{\eta\rho K^2}{\epsilon C_1}). \quad (2)$$

Proof Sketch

We now highlight the key ideas and challenges behind our convergence proof.

Step 1: Similar to Theorem 1 in Xie, Koyejo, and Gupta (2019), we bound the expected cost reduction after K steps of local updates on the j-th agent, $\forall j \in [N]$, as

$$\mathbb{E}[F(\tilde{x}_j(t)) - F(x_j(t))] = \mathbb{E}[F(x_j(t,K)) - F(x_j(t,0))]$$
$$\leq -\eta\epsilon \sum_{k=0}^{K-1} \mathbb{E}||\nabla F(x_j(t,k))||^2 + \eta^2\mathcal{O}(\rho K^3 V).$$

Step 2: For any epoch t, we find the index $M(t)$ of the agent whose model is the "worst", i.e., $M(t) = \arg\max_{j\in[N]}\{F(x_j(t))\}$, and the "worst" model on all agents over the time period of $[t-\tau_{max}+1, t]$ as

$$\mathcal{T}(t, \tau_{max}) = \arg\max_{t\in[t-\tau_{max}+1,t]}\{F(x_{M(t)}(t))\}.$$

Step 3: We bound the cost reduction of the "worst" model at epoch $t+1$ from the "worst" model in the time period of $[t-\tau_{max}+1, t]$, i.e., the worst possible model that can be stored in some agent's cache at time t, as:

$$\mathbb{E}[F\left(x_{M(t+1)}(t+1)\right) - F\left(x_{M(\mathcal{T}(t,\tau_{max}))}(\mathcal{T}(t,\tau_{max}))\right)]$$
$$\leq -\epsilon\eta C_1 K \min_{\tau=t}^{t-\tau_{max}+1} ||\nabla F(x(\tau))||^2 + \eta^2\mathcal{O}(\rho K^3 V). \quad (3)$$

Step 4: We iteratively construct a time sequence $\{T'_0, T'_1, T'_2, ..., T'_{N_T}\} \subseteq \{0, 1, ..., T-1\}$ in the backward fashion so that

$$\begin{aligned} T'_{N_T} &= T-1; \\ T'_i &= \mathcal{T}(T'_{i+1}, \tau_{max}) - 1, \quad 1 \leq i \leq N_T - 1; \\ T'_0 &= 0. \end{aligned}$$

Step 5: Applying inequality (3) at all time instances $\{T'_0, T'_1, T'_2, ..., T'_{N_T}\}$, after T global epochs, we have,

$$\min_{t=0}^{T-1} \mathbb{E}[||\nabla F(x(t))||^2] \leq \frac{1}{N_T}\sum_{t=T'_0}^{T'_{N_T}} \min_{\tau=t}^{t-\tau_{max}+1} ||\nabla F(x(\tau))||^2$$
$$\leq \frac{1}{\epsilon\eta C_1 K N_T}\sum_{t=T'_0}^{T'_{N_T}} \mathbb{E}[F(x_{M(\mathcal{T}(t,\tau_{max}))}(\mathcal{T}(t,\tau_{max})))$$
$$- F(x_{M(t+1)}(t+1))] + \mathcal{O}(\frac{\eta\rho K^2 V}{\epsilon C_1})$$
$$\leq \frac{\mathbb{E}[F(x(0)) - F(x_{M(T)}(T))]}{\epsilon\eta C_1 K N_T} + \mathcal{O}(\frac{\eta\rho K^2 V}{\epsilon C_1})$$
$$\leq \mathcal{O}(\frac{\tau_{max}}{\epsilon\eta C_1 KT}) + \mathcal{O}(\frac{\eta\rho K^2}{\epsilon C_1}). \quad (4)$$

Step 6: With the results in (4) and by leveraging Theorem 1 in Yang, Fang, and Liu (2021), `Cached-DFL` converges to a critical point after T global epochs.

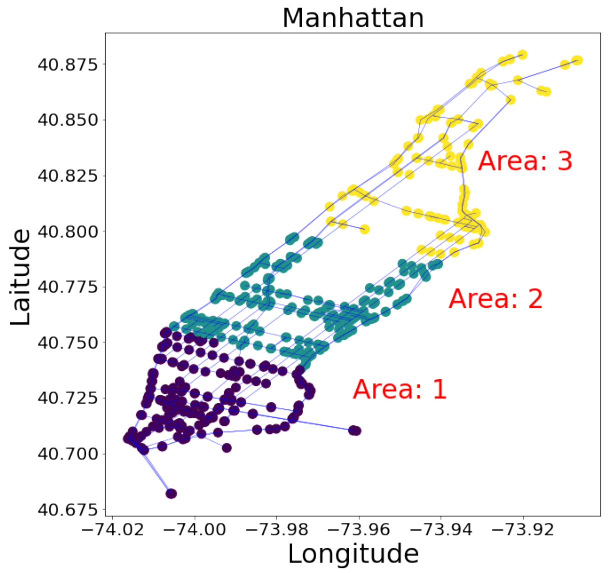

Figure 1: Manhattan Mobility Model Map. The dots represent the intersections while the edges between nodes represent roads in Manhattan.

Experiments

We implement `Cached-DFL` with PyTorch (Paszke et al. 2017) on Python3. Further details about experiments, hyperparameters, and additional results are provided in our technical report Wang et al. (2024).

Datasets. We conduct experiments using three standard FL benchmark datasets: MNIST (Deng 2012), FashionMNIST (Xiao, Rasul, and Vollgraf 2017), CIFAR-10 (Krizhevsky 2009) on $N = 100$ vehicles, with CNN, CNN and ResNet-18 (He et al. 2016) as models respectively. We evaluate three different data distribution settings: *non-i.i.d, i.i.d, and Dirichlet*. In *extreme non-i.i.d*, we use a setting similar to Su, Zhou, and Cui (2022), data points in training set are sorted by labels and then evenly divided into 200 shards, with each shard containing 1–2 labels out of 10 labels. Then 200 shards are randomly assigned to 100 vehicles unevenly: 10% vehicles receive 4 shards, 20% vehicles receive 3 shards, 30% vehicles receive 2 shards and the rest 40% receive 1 shard. For *i.i.d*, we randomly allocate all the training data points to 100 vehicles. For *Dirichlet distribution*, we follow the setting in Xiong et al. (2024), to take a heterogeneous allocation by sampling $p_i \sim Dir_N(\pi)$, where π is the parameter of Dirichlet distribution. We take $\pi = 0.5$ in our following experiments.

Evaluation Setup. The baseline algorithm is DeFedAvg (Sun, Li, and Wang 2022), which implements simple decentralized federated optimization. For convenience, we name DeFedAvg as DFL in the following results. We set batch size to 64 in all experiments. For MNIST and FashionMNIST, we use 60k data points for training and 10k data points for

Algorithm 2: LRU Model Cache Update (LRU Update)

```
Input: Current cache $\mathcal{C}_i(t)$, agent $j$'s cache $\mathcal{C}_j(t)$, model $x_j(t)$ from agent $j$, current time $t$, cache size $\mathcal{C}_{\text{max}}$, staleness tolerance $\tau_{\text{max}}$
     Main Process:
 1: for each $x_k(\tau) \in \mathcal{C}_i(t)$ or $\mathcal{C}_j(t)$ do
 2:     if $t - \tau \geq \tau_{\text{max}}$ then
 3:         Remove $x_k(\tau)$ from the respective cache ($\mathcal{C}_i(t)$ or $\mathcal{C}_j(t)$)
 4:     end if
 5: end for
 6: Add or replace $x_j(t)$ into $\mathcal{C}_i(t)$
 7: for each $x_k(\tau) \in \mathcal{C}_j(t)$ do
 8:     if $x_k(\tau) \notin \mathcal{C}_i(t)$ then
 9:         Add $x_k(\tau)$ into $\mathcal{C}_i(t)$
10:     else
11:         Retrieve $x_k(\tau') \in \mathcal{C}_i(t)$
12:         if $\tau > \tau'$ then
13:             Replace $x_k(\tau')$ with $x_k(\tau)$ in $\mathcal{C}_i(t)$
14:         end if
15:     end if
16: end for
17: Sort models in $\mathcal{C}_i(t)$ in descending order of $\tau$
18: Retain only the first $\mathcal{C}_{\text{max}}$ models in $\mathcal{C}_i(t)$
19: return $\mathcal{C}_i(t+1)$
Output: $\mathcal{C}_i(t+1)$
```

testing. For CIFAR-10, we use 50k data points for training and 10k data points for testing. Different from training set partition, we do not split the testset. For MNIST and FashionMNIST, we test local models of 100 vehicles on the 10k data points of the whole test set and get the average test accuracy for the evaluation metric. What's more, for CIFAR-10, due to the computing overhead, we sample 1,000 data points from test set for each vehicle and use the average test accuracy of 100 vehicles as the evaluation metric. For all the experiments, we train for 1,000 global epochs, and implement early stop when the average test accuracy stops increasing for at least 20 epochs. For MNIST and Fashion-MNIST experiments, we use 10 compute nodes, each with 10 CPUs, to simulate DFL on 100 vehicles. CIFAR-10 results are obtained from 1 compute node with 5 CPUs and 1 A100 NVIDIA GPU.

Optimization Method. We use SGD as the optimizer and set the initial learning rate $\eta = 0.1$, and use learning rate scheduler named **ReduceLROnPlateau** from PyTorch, to automatically adjust the learning rate for each training.

Mobile DFL Simulation. Manhattan Mobility Model maps are derived from real Manhattan road data (INRIX (INRIX 2024)), as shown in Fig. 1. Following Bai, Sadagopan, and Helmy (2003), vehicles move along a grid of horizontal and vertical streets, turning left, right, or going straight at intersections according to specified probabilities (e.g., 0.5 to continue straight and 0.1667 per road among three options). As in Su, Zhou, and Cui (2022), each vehicle is equipped with DSRC and mmWave, can communicate within 100 m, and travels at 13.89 m/s. We set the number of local updates to

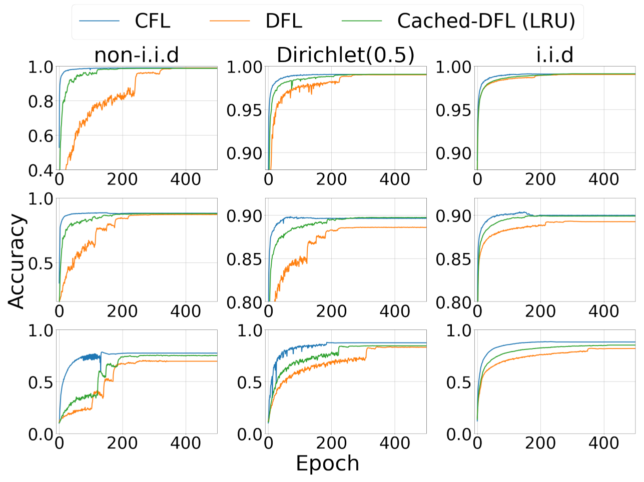

Figure 2: `Cached-DFL` vs. DFL without Caching across Different Datasets: The first row presents results for MNIST, the second row for FashionMNIST, and the third row for CIFAR-10.

$K = 10$, with a 120-second interval between global epochs. During each epoch, vehicles train their models while moving and exchange models when they encounter each other.

LRU Cache Update. Algorithm 2, which we name as LRU method for convenience, is the basic cache update method we proposed. Basically, the LRU updating rule aims to fetch the most recent models and keep as many of them in the cache as possible. At lines 12 and 17, the metric for making choice among models is the timestamp of models, which is defined as the epoch time when the model was received from the original vehicle, rather than received from the cache. What's more, to fully utilize the caching mechanism, a vehicle not only fetches other vehicles' own trained models, but also models in their caches. For instance, at epoch t, vehicle i can directly fetch model $x_j(t)$ from vehicle j, at line 6, and also fetch models $x_k(\tau) \in \mathcal{C}_j(t)$ from the cache of vehicle j, at lines 7-16. This way, each vehicle can not only directly fetch its neighbors' models, but also indirectly fetch models of its neighbors' neighbors, thus boosting the spreading of the underlying data information from different vehicles, and improving the DFL convergence speed, especially with heterogeneous data distribution. Additionally, at lines 1-5, before updating cache, models with staleness $t - \tau \geq \tau_{max}$ will be removed from each vehicle's cache.

Experimental Results

Caching vs. Non-caching. To evaluate the performance of `Cached-DFL`, we compare the DFL with LRU Cache, Centralized FL (CFL) and DFL on MNSIT, FashionMNIST, CIFAR-10 with three distributions: non-i.i.d, i.i.d, Dirichlet with $\pi = 0.5$. For LRU update, we take the cache size as 10 for MNIST and FashionMNIST, and 3 for CIFAR-10 and $\tau_{max} = 5$ based on practical considerations. Given a speed of $13.89m/s$ and and a communication distance of $100m$, the communication window of two agents driving in opposite directions could be limited. Additionally, the above

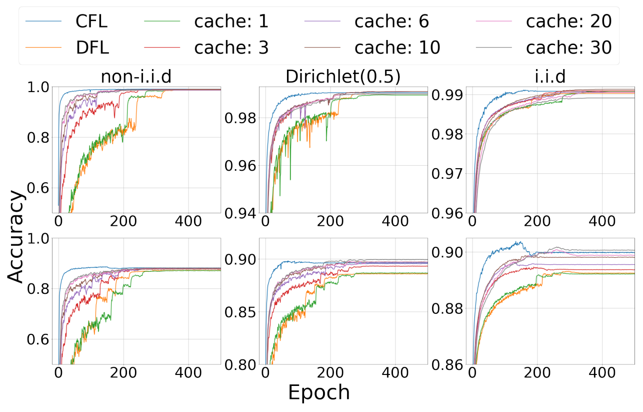

Figure 3: Cached-DFL with LRU at Different Cache Sizes: The first row presents results for MNIST, while the second row corresponds to FashionMNIST.

τ_{max}	1	2	3	4	5	10	20
30s	0.9	1.7	2.7	3.9	5.2	14.9	44.8
	0.0	0.5	1.0	1.6	2.2	5.4	11.6
60s	1.7	3.8	6.5	10.4	14.1	43.1	90.3
	0.0	0.6	1.2	1.8	2.4	5.6	9.8
120s	3.7	9.9	18.5	31.4	35.2	90.2	98.5
	0.0	0.6	1.3	1.9	1.5	4.7	5.2

Table 2: Average number and average age of cached models with different τ_{max} (columns) and different epoch times: 30s, 60s, 120s (rows). Each row has two sub-rows: the first shows the average number of cached models, and the second shows their average age.

cache sizes were chosen by considering the size of chosen models and communication overhead. From the results in Fig. 2, we can see that `Cached-DFL` boosts the convergence and outperforms non-caching method (DFL) and gains performance much closer to CFL in all the cases, especially in non-i.i.d. scenarios.

Impact of Cache Size. Then we evaluate the performance gains with different cache sizes from 1 to 30 and $\tau_{max} = 10$, on MNIST and FashionMNIST in Fig. 3. LRU can benefit more from larger cache sizes, especially in non-i.i.d scenarios, as aggregation with more cached models gets closer to CFL and training with global data distribution.

Impact of Model Staleness. One drawback of model caching is introducing stale models into aggregation, so it is very important to choose a proper staleness tolerance τ_{max}. First, we statistically calculate the relation between the average number and the average age of cached models at different τ_{max} from 1 to 20, when epoch time is 30s, 60s, 120s, with unlimited cache size, in Table 2. We can see that, with the fixed epoch time, as the τ_{max} increases, the average number and average age of cached models increase, approximately linearly. It's not hard to understand, as every epoch each agent can fetch a limited number of models directly from other agents. Increasing the staleness tolerance τ_{max} will increase the number of cached models, as well as the age of cached models. What's more, we can see that the communication frequency or the moving speed of agents will also impact the average age of cached models, as the faster an agent moves, the more models it can fetch within each epoch, which we will further discuss later.

Mobility's Impact on Convergence. Fig. 4 compares DFL and LRU at different $\tau_{\max}$ on MNIST under non-i.i.d and i.i.d settings. Here we pick the epoch time 30s for a clear view. Under non-i.i.d, a larger $\tau_{\max}$ initially speeds up convergence by allowing more cached models (confirming our earlier findings), as it allows for more cached models which bring more benefits than harm to the training with non-i.i.d. Under i.i.d, however, more cached models bring no gain and staleness hinders convergence. Moreover, focusing on the final phase (bottom of each figure), higher $\tau_{\max}$ reduces final accuracy in both scenarios due to staleness. Despite this, with high $\tau_{\max}$, LRU can still match or outperform DFL in non-i.i.d settings.

Vehicle mobility directly determines the frequency and efficiency of cache-based model spreading and aggregation. So we evaluate the performance of `Cached-DFL` at different vehicle speeds. We fix cache size at 10, $\tau_{max} = 10$ with non-i.i.d. data distribution. We take the previous speed $v = v_0 = 13.89$ m/s and $K = 30$ local updates as the base, named as speedup x1. To speedup, v increases while K reduces to keep the fair comparison under the same wall clock. For instance, for speedup x3, $v = 3v_0$ and $K = 10$. Results in Fig. 5 show that when the mobility speed increases, although the number of local updates decreases, the spread of all models in the whole vehicle network is boosted, thus disseminating local models more quickly among all vehicles leading to faster model convergence.

Grouped Mobility Patterns and Data Distributions. In practice, vehicle mobility patterns and local data distributions may naturally form groups. For example, a vehicle may mostly move within its home area, and collect data specific to that area. Vehicles within the same area meet with each others frequently, but have very similar data distributions. A fresh model of a vehicle in the same area is not as valuable as a slightly older model of a vehicle from another area. So model caching should not just consider model freshness, but should also take into account the coverage of group-based

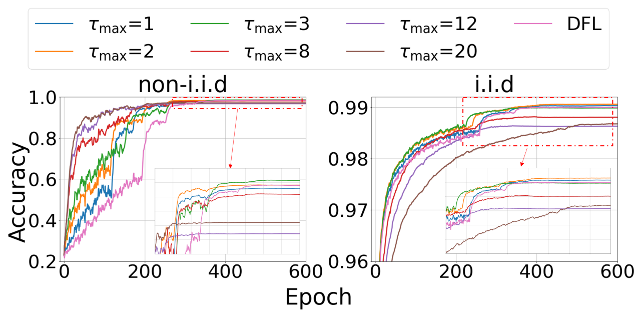

Figure 4: Impact of τ_{max} on Model Convergence for MNIST.

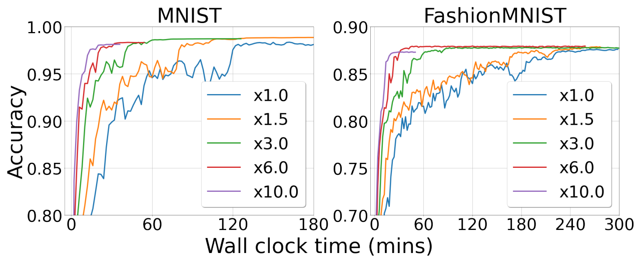

Figure 5: Convergence at Different Mobility Speeds under Non-IID Data Distribution.

data distributions. For those scenarios, we develop a Group-based (GB) caching algorithm. More specifically, knowing that there are m distribution groups, one should maintain balanced presences of models from different groups. One straightforward extension of the LRU caching algorithm is to partition the cache into m sub-caches, one for each group. Each sub-cache is updated by local models from its associated group, using the LRU policy. Due to limited space, the detailed algorithm is presented in our technical report Wang et al. (2024).

We now conduct a case study for group-based cache update. As shown in Fig. 1, the whole Manhattan road network is divided into 3 areas, downtown, mid-town and uptown. Each area has 30 area-restricted vehicles that randomly moves within that area, and 3 or 4 free vehicles that can move into any area. We set 4 different area-related data distributions: Non-overlap, 1-overlap, 2-overlap, 3-overlap. n-overlap means the number of shared label classes between areas is n. We use the same non-i.i.d shards method in the previous section to allocate data points to the vehicles in the same area. On each vehicle, we evenly divide the cache for the three areas. We evaluate our proposed GB cache method on FashionMNIST. As shown in Fig. 6, while vanilla LRU converges faster at the very beginning, it cannot outperform DFL at last. However, the GB cache update method can solve the problem of LRU update and outperform DFL under different overlap settings.

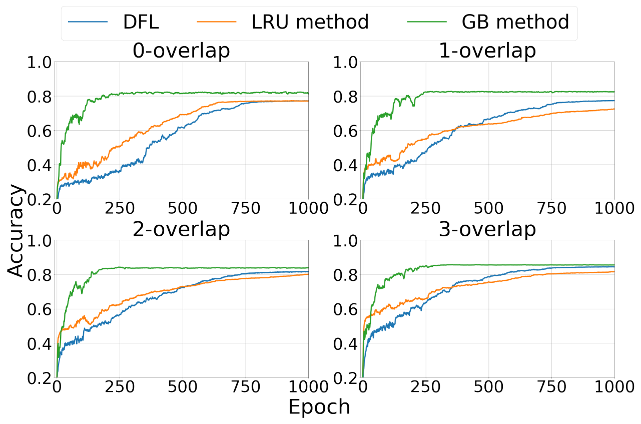

Figure 6: Group-based LRU Cache Update Performance under Different Data Distribution Overlaps on FashionMNIST.

Discussions

In general, the convergence rate of `Cached-DFL` outperforms non-caching method (DFL), especially for non-i.i.d. distributions. Larger cache size and smaller τ_{max} make `Cached-DFL` closer to the performance of CFL. Empirically, there is a trade-off between the age and number of cached models, it is critical to choose a proper τ_{max} to control the staleness. What's more, the choice of τ_{max} should take into account the diversity data distributions on agents. In general, with non-i.i.d data distributions, the benefits of increasing the number of cached models can outweigh the damages caused by model staleness; while when data distributions are close to i.i.d, it is better to use a small number of fresh models than a large number of stale models. Similarly conclusions can also be drawn from the results of area-restricted vehicles. What's more, in a system of moving agents, the mobility will also have big impact on the training, usually higher speed and communication frequency improve the model convergence and accuracy.

Related Work

Decentralized FL (DFL) has been increasingly applied in vehicular networks, leveraging existing frameworks like vehicle-to-vehicle (V2V) communication (Yuan et al. 2024). V2V FL facilitates knowledge sharing among vehicles and has been explored in various studies (Samarakoon et al. 2019; Pokhrel and Choi 2020; Yu et al. 2020; Chen et al. 2021; Barbieri et al. 2022; Su, Zhou, and Cui 2022). Samarakoon et al. (2019) studied optimized joint power and resource allocation for ultra-reliable low-latency communication (URLLC) using FL. Su, Zhou, and Cui (2022) introduced DFL with Diversified Data Sources to address data diversity issues in DFL, improving model accuracy and convergence speed in vehicular networks. None of the previous studies explored model caching on vehicles. Convergence of asynchronous federated optimization was studied in Xie, Koyejo, and Gupta (2019). Their analysis focused on pairwise model aggregation between an agent and the parameter server, does not cover decentralized model aggregation with stale cached models in our proposed framework.

Conclusion & Future Work

In this paper, we developed `Cached-DFL`, a novel decentralized Federated Learning framework that leverages on model caching on mobile agents for fast and even model spreading. We theoretically analyzed the convergence of `Cached-DFL`. Through extensive case studies in a vehicle network, we demonstrated that `Cached-DFL` significantly outperforms DFL without model caching, especially for agents with non-i.i.d data distributions. We employed only simple model caching and aggregation algorithms in the current study. We will investigate more refined model caching and aggregation algorithms customized for different agent mobility patterns and non-i.i.d. data distributions.

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) grants 2148309, 2315614 and 2337914, and was supported in part by funds from OUSD R&E, NIST, and industry partners as specified in the Resilient & Intelligent NextG Systems (RINGS) program. This work was also supported in part through the NYU IT High Performance Computing resources, services, and staff expertise. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

Bai, F.; Sadagopan, N.; and Helmy, A. 2003. IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of RouTing protocols for Adhoc NeTworks. In *IEEE INFOCOM 2003*, volume 2, 825–835.

Barbieri, L.; Savazzi, S.; Brambilla, M.; and Nicoli, M. 2022. Decentralized federated learning for extended sensing in 6G connected vehicles. *Vehicular Communications*, 33: 100396.

Burleigh, S.; Hooke, A.; Torgerson, L.; Fall, K.; Cerf, V.; Durst, B.; Scott, K.; and Weiss, H. 2003. Delay-tolerant networking: an approach to interplanetary internet. *IEEE Communications Magazine*, 41(6): 128–136.

Chen, J.-H.; Chen, M.-R.; Zeng, G.-Q.; and Weng, J.-S. 2021. BDFL: A byzantine-fault-tolerance decentralized federated learning method for autonomous vehicle. *IEEE Transactions on Vehicular Technology*, 70(9): 8639–8652.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6): 141–142.

Fall, K. 2003. A delay-tolerant network architecture for challenged internets. SIGCOMM '03, 27–34. New York, NY, USA.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

INRIX, I. 2024. INRIX documentation. Available at https://docs.inrix.com/. Accessed: January 26, 2025.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, University of Tront*.

Li, C.; Wang, X.; Zong, T.; Cao, H.; and Liu, Y. 2023. Predictive edge caching through deep mining of sequential patterns in user content retrievals. *Computer Networks*, 233: 109866.

Liu, J.; Huang, J.; Zhou, Y.; Li, X.; Ji, S.; Xiong, H.; and Dou, D. 2022. From distributed machine learning to federated learning: A survey. *Knowledge and Information Systems*, 64(4): 885–917.

Martínez Beltrán, E. T.; Pérez, M. Q.; Sánchez, P. M. S.; Bernal, S. L.; Bovet, G.; Pérez, M. G.; Pérez, G. M.; and Celdrán, A. H. 2023. Decentralized Federated Learning: Fundamentals, State of the Art, Frameworks, Trends, and Challenges. *IEEE Communications Surveys & Tutorials*, 25(4): 2983–3013.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Niknam, S.; Dhillon, H. S.; and Reed, J. H. 2020. Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine*, 58(6): 46–51.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Pokhrel, S. R.; and Choi, J. 2020. A decentralized federated learning approach for connected autonomous vehicles. In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 1–6. IEEE.

Robbins, H.; and Monro, S. 1951. A stochastic approximation method. *The annals of mathematical statistics*, 400–407.

Samarakoon, S.; Bennis, M.; Saad, W.; and Debbah, M. 2019. Distributed federated learning for ultra-reliable low-latency vehicular communications. *IEEE Transactions on Communications*, 68(2): 1146–1159.

Su, D.; Zhou, Y.; and Cui, L. 2022. Boost decentralized federated learning in vehicular networks by diversifying data sources. In *2022 IEEE 30th International Conference on Network Protocols (ICNP)*, 1–11. IEEE.

Sun, T.; Li, D.; and Wang, B. 2022. Decentralized federated averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wang, X.; Xiong, G.; Cao, H.; Li, J.; and Liu, Y. 2024. Decentralized Federated Learning with Model Caching on Mobile Agents. arXiv:2408.14001.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747.

Xie, C.; Koyejo, S.; and Gupta, I. 2019. Asynchronous Federated Optimization. *CoRR*, abs/1903.03934.

Xiong, G.; Yan, G.; Wang, S.; and Li, J. 2024. DePRL: Achieving Linear Convergence Speedup in Personalized Decentralized Learning with Shared Representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 16103–16111.

Yang, H.; Fang, M.; and Liu, J. 2021. Achieving Linear Speedup with Partial Worker Participation in Non-IID Federated Learning. *Proceedings of ICLR*.

Yu, Z.; Hu, J.; Min, G.; Xu, H.; and Mills, J. 2020. Proactive content caching for internet-of-vehicles based on peer-to-peer federated learning. In *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, 601–608. IEEE.

Yuan, L.; Wang, Z.; Sun, L.; Philip, S. Y.; and Brinton, C. G. 2024. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*.