arXiv:2408.14001v2 [cs.LG] 4 Feb 2025

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.

1 Introduction

1.1 Federated Learning on Mobile Agents

Federated learning (FL) is a type of distributed machine learning (ML) that prioritizes data privacy [12]. 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) [16] 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 [2], 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 [13]. 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 [11]. 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 [12], developed for centralized FL to DFL cannot achieve fast convergence and high model accuracy [10].

1.2 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)* [3, 6]: 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. [9].

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;
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.

Table 1: Notations and Terminologies.

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

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.

2 Background

2.1 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:

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

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 $\mathcal{D}^i$.

2.2 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 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 $\mathcal{C}_i(t)$, following 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.

Algorithm 1 Cached Decentralized **F**ederated **L**earning (`Cached-DFL`)

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

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

function ModelAggregation($\mathcal{C}_i(t)$)
  $x_i(t+1) = \sum_{j \in \mathcal{C}_i(t)} \alpha_j \tilde{x}_j(\tau)$
  return $x_i(t+1)$
end function
Main Process:
for $t = 0, 1, \ldots, T-1$ **do**
  for $i = 1, 2, \ldots, N$ **do**
    $\tilde{x}_i(t) \leftarrow$ LocalUpdate($x_i(t)$)
    $\mathcal{C}_i(t) \leftarrow$ CacheUpdate($\mathcal{C}_i(t-1), \tau_{\max}$)
    $x_i(t+1) \leftarrow$ ModelAggregation($\mathcal{C}_i(t)$)
  end for
end for

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

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 [19]), 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.

3 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 $\forall x, y$, $f(y) - f(x) \leq \langle \nabla f(x), y - x \rangle + \frac{L}{2}||y - x||^2$, 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 $||\nabla F_j(x) - \nabla F(x)||^2 \leq \varsigma^2, \forall j \in [N], x \in \mathbb{R}^d$.*

Definition 3 (L-Lipschitz Continuous Gradient)**.** *There exists a constant $L > 0$, such that $||\nabla F(x) - \nabla F(x')|| \leq L'||x - x'||, \forall x, x' \in \mathbb{R}^d$.*

Theorem 4. *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 $\forall z \sim \mathcal{D}^i, ||\nabla f(x; z)||^2 \leq V, ||\nabla g_{x'}(x; z)||^2 \leq V, \forall x' \in \mathbb{R}^d$, and ∇F satisfies L-Lipschitz Continuous Gradient. For any small constant $\epsilon > 0$, if we take $\rho > 0$, and satisfying $-(1 + 2\rho + \epsilon)V + (\rho^2 - \frac{\rho}{2})||x(t, k-1) - x(t)||^2 \geq 0, \forall x(t, k-1), x(t)$, after T global epochs, Algorithm 1 converges to a critical point:*

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

3.1 Proof Sketch

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

Step 1: Similar to Theorem 1 in Xie et al. [21], we bound the expected cost reduction after K steps of local updates on the j-th agent, $\forall j \in [N]$, as

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

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:

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

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,

$$\begin{aligned} \min_{t=0}^{T-1} \mathbb{E}[||\nabla F(x(t))||^2] &\leq \frac{1}{N_T} \sum_{t=T_0'}^{T_{N_T}'} \min_{\tau=t}^{t-\tau_{max}+1} ||\nabla F(x(\tau))||^2 \\ &\leq \frac{1}{\epsilon\eta C_1 K N_T} \sum_{t=T_0'}^{T_{N_T}'} \mathbb{E}[F(x_{M(\mathcal{T}(t,\tau_{max}))}(\mathcal{T}(t,\tau_{max}))) - F(x_{M(t+1)}(t+1))] + \mathcal{O}(\frac{\eta\rho K^2 V}{\epsilon C_1}) \\ &\leq \frac{1}{\epsilon\eta C_1 K N_T} \mathbb{E}[F(x(0)) - F(x_{M(T)}(T)] + \mathcal{O}(\frac{\eta\rho K^2 V}{\epsilon C_1}) \\ &\leq \mathcal{O}(\frac{\tau_{max}}{\epsilon\eta C_1 K T}) + \mathcal{O}(\frac{\eta\rho K^2}{\epsilon C_1}). \end{aligned} \tag{4}$$

Step 6: With the results in (4) and by leveraging Theorem 1 in Yang et al. [23], `Cached-DFL` converges to a critical point after T global epochs.

4 Realization of `DFedCache`

We implement `Cached-DFL`[1] with PyTorch [14] on Python3.

4.1 Datasets

We conduct experiments using three standard FL benchmark datasets: MNIST [5], FashionMNIST [20], CIFAR-10 [8] on $N = 100$ vehicles, with CNN, CNN and ResNet-18 [7] 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 et al. [18], 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. [22], 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.

[1] https://github.com/ShawnXiaoyuWang/Cached-DFL

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 τ_{max}

Main Process:

for each $x_k(\tau) \in \mathcal{C}_i(t)$ or $\mathcal{C}_j(t)$ **do**
  if $t - \tau \geq \tau_{\text{max}}$ **then**
    Remove $x_k(\tau)$ from the respective cache ($\mathcal{C}_i(t)$ or $\mathcal{C}_j(t)$)
  end if
end for
Add or replace $x_j(t)$ into $\mathcal{C}_i(t)$
for each $x_k(\tau) \in \mathcal{C}_j(t)$ **do**
  if $x_k(\tau) \notin \mathcal{C}_i(t)$ **then**
    Add $x_k(\tau)$ into $\mathcal{C}_i(t)$
  else
    Retrieve $x_k(\tau') \in \mathcal{C}_i(t)$
    if $\tau > \tau'$ **then**
      Replace $x_k(\tau')$ with $x_k(\tau)$ in $\mathcal{C}_i(t)$
    end if
  end if
end for
Sort models in $\mathcal{C}_i(t)$ in descending order of τ
Retain only the first $\mathcal{C}_{\text{max}}$ models in $\mathcal{C}_i(t)$
return $\mathcal{C}_i(t+1)$

Output: $\mathcal{C}_i(t+1)$

4.2 Evaluation Setup

The baseline algorithm is DeFedAvg [19], 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 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 FashionMNIST 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.

4.3 Optimization Method

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

4.4 Mobile DFL Simulation

Manhattan Mobility Model maps are generated from the real road data of Manhattan from INRIX[2], which is shown in Fig. 1. We follow the setting from Bai et al. [1]: a vehicle is allowed to move along the grid of

[2]© 2024 INRIX, Inc.,"INRIX documentation," INRIX documentation `https://docs.inrix.com/`.

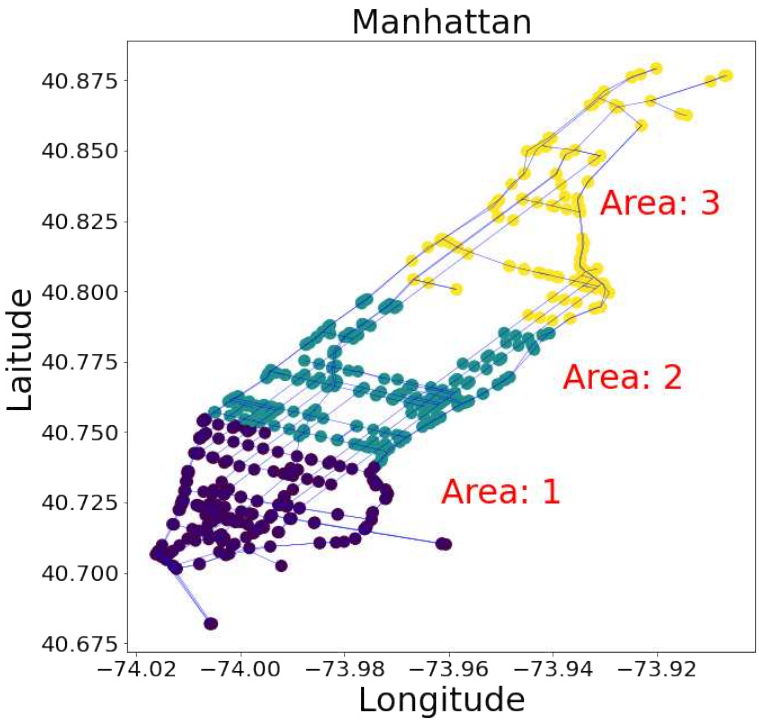

Figure 1: Manhattan Mobility Model Map. The dots represent the intersections while the edges between nodes represent road in Manhattan.

horizontal and vertical streets on the map. At an intersection, the vehicle can turn left, right or go straight. This choice is probabilistic: the probability of moving on the same street is 0.5, and the probabilities of turn into one of the rest roads are equal. For example, at an intersection with 3 more roads to turn into, then the probability to turn into each road is 0.1667. Follow the setting from Su et al. [18]: each vehicle is equipped with DSRC and mmWave communication components, can communicate to other vehicles within the communication range of 100 meters, while the default velocity of each vehicle is 13.89 m/s. In our experiments, we set up the number of local updates $K = 10$, and the interval between each epoch is 120 seconds. Within each global epoch, vehicles train their models on local datasets while moving on the road and updating their cache by uploading and downloading models with other encountered vehicles.

4.5 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.

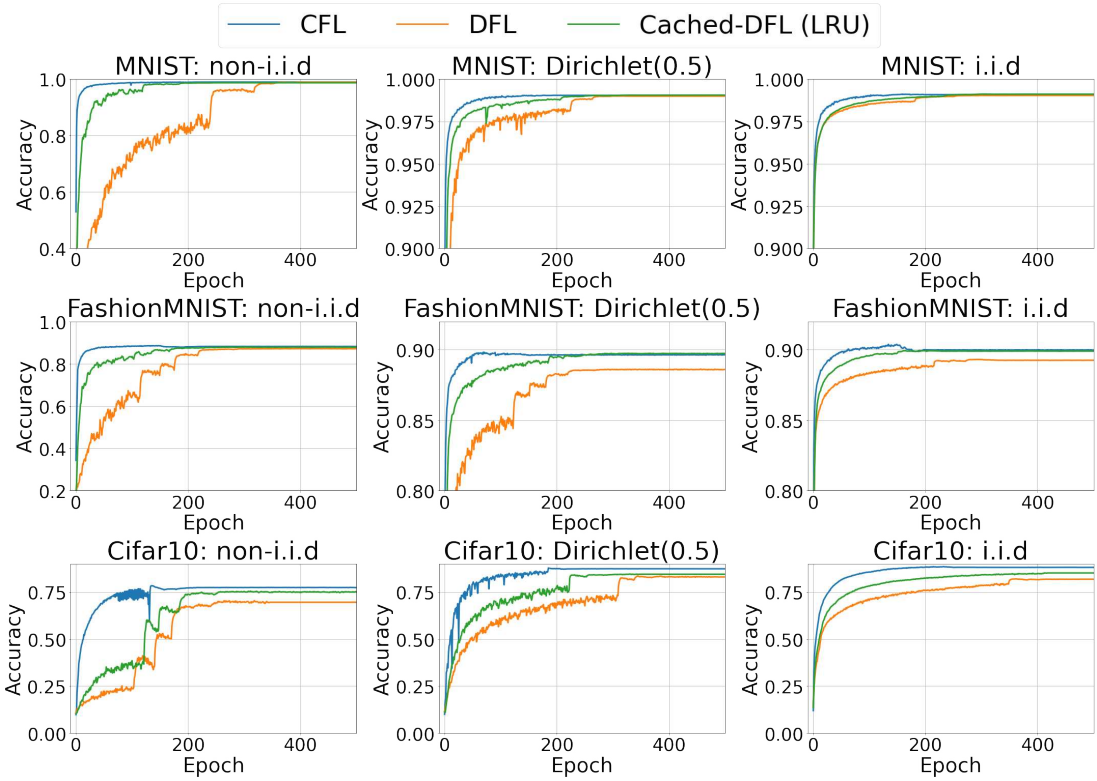

Figure 2: DFL with Caching vs. DFL without Caching.

5 Numerical Results

5.1 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 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.

5.2 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.

Table 2: Average number and average age of cached models with different τ_{max} and different epoch time: 30s, 60s, 120s. Columns for different τ_{max}, and rows for different epoch time. There are two sub-rows for each epoch time, with the first sub-row be the average number of cached models, the second sub-row be their average age.

	τ_{max}	1	2	3	4	5	10	20
30s	num	0.8549	1.6841	2.6805	3.8584	5.2054	14.8520	44.8272
	age	0.0000	0.4904	1.0489	1.6385	2.2461	5.4381	11.5735
60s	num	1.6562	3.8227	6.4792	10.3889	14.0749	43.0518	90.2576
	age	0.0000	0.5593	1.1604	1.8075	2.4396	5.5832	9.7954
120s	num	3.6936	9.9149	18.5016	31.3576	35.1958	90.2469	98.5412
	age	0.0000	0.6189	1.2715	1.9194	1.5081	4.7131	5.2357

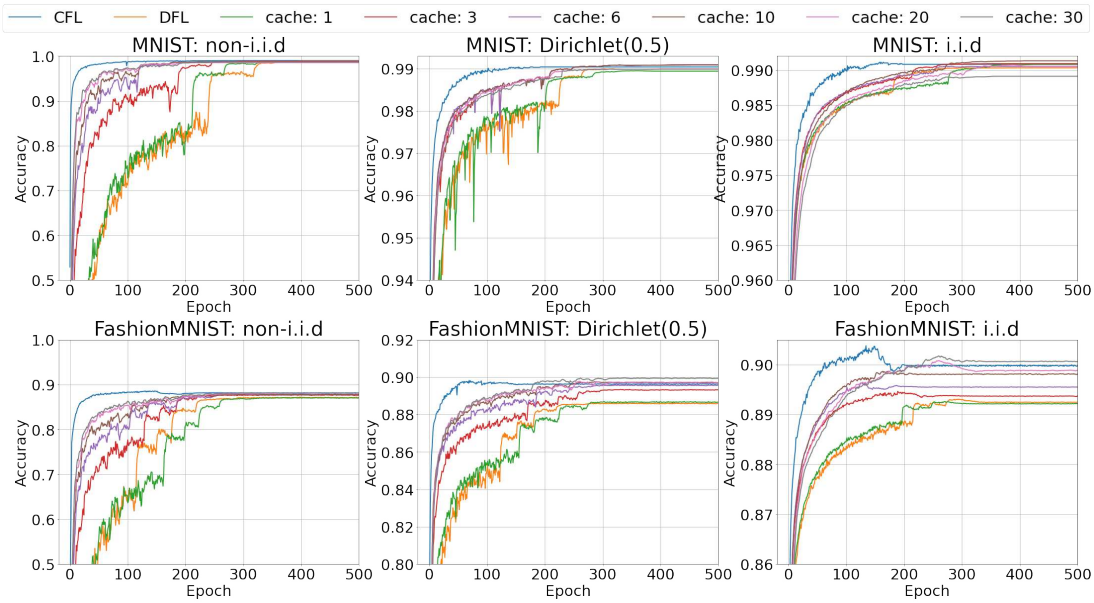

Figure 3: DFL with LRU at Different Cache Sizes.

5.3 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.

We also compare the performance of DFL with LRU at different τ_{max} on MNIST under non-i.i.d and i.i.d in Fig. 4. Here we pick the epoch time 30s for a clear view. First, for non-i.i.d, a larger τ_{max} brings faster convergence at the beginning, which is consistent with our previous conclusion, as it allows for more cached models which bring more benefits than harm to the training with non-i.i.d. However, for i.i.d scenarios, larger τ_{max} stops improving the performance of models, as introducing more cached models

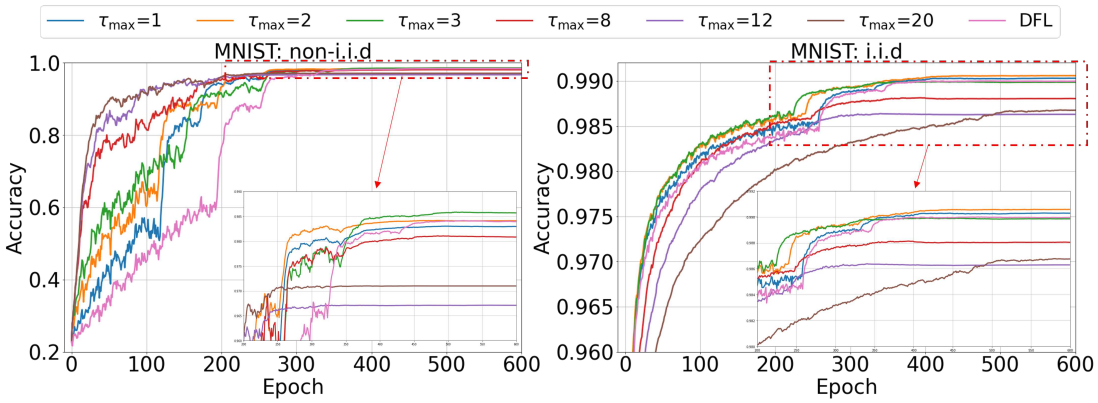

Figure 4: Impact of τ_{max} on Model Convergence.

hardly benefits training in i.i.d, and the model staleness will prevent convergence. What's more, if we zoom in the final converging phase, which is amplified in the bottom of each figure, we observe that larger τ_{max} will reduce the final converged accuracy in both non-i.i.d and i.i.d scenarios, as it introduces more staleness. Even with large τ_{max} and large staleness, LRU method still achieves similar or even better performance than DFL in non-i.i.d. scenarios.

5.4 Mobility's Impact on Convergence

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 our previous speed $v = v_0 = 13.89m/s$ 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.

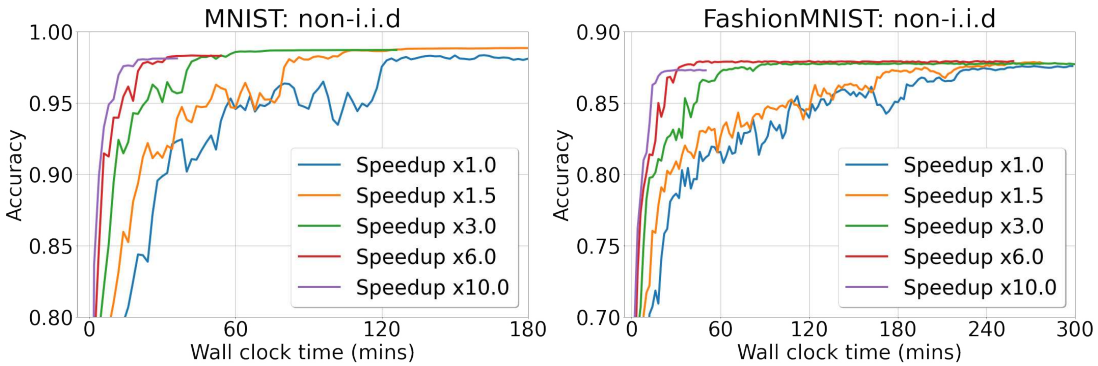

Figure 5: Convergence at Different Mobility Speed.

5.5 Experiments with 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 data distributions. For those scenarios, we develop a Group-based (GB) caching algorithm as Algorithm 3. 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.

Algorithm 3 Group Based Cache Update (GB Update)

Input: Current cache $\mathcal{C}_i(t)$, Cache of device j: $\mathcal{C}_j(t)$, Model $x_j(t)$ from device j, Current time t, Cache size $\mathcal{C}_{max}$, Tolerance of staleness τ_{max}, Device group mapping list $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_n\}$, here $\mathcal{A}_i \in \{1, 2, ..., m\}, \forall i \in [N]$. Group cache size list $\mathcal{R} = \{r_1, r_2, ..., r_m\}$, here $r_{\mathcal{A}_i}$ is number of cache slots reserved for models from devices belong to group $\mathcal{A}_i$.
Here the model $x(i, t')$ is local updated at time t', based on car i's dataset.

function GROUP_PRUNE_CACHE($\mathcal{C}_i(t), \mathcal{R}, \mathcal{A}$):
    Create empty lists of group $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_m$,
    for $x_k(\tau) \in \mathcal{C}_i(t)$ **do**
        Put $x_k(\tau)$ into $\mathcal{L}_{\mathcal{A}_k}$
    end for
    Empty $\mathcal{C}_i(t)$
    for $k = 1, 2, ..., m$ **do**
        Sort models in $\mathcal{L}_k$ by τ descending.
        Take and put first r_k models in $\mathcal{L}_k$ into $\mathcal{C}_i(t)$
    end for
    return $\mathcal{C}_i(t+1)$
end function

Main Process:
for $x_k(\tau) \in \mathcal{C}_i(t)$ or $\mathcal{C}_j(t)$ **do**
    if $t - \tau \geq \tau_{max}$ **then**
        Remove $x_k(\tau)$ from related cache $\mathcal{C}_i(t)$ or $\mathcal{C}_j(t)$
    end if
end for
Add or replace $x_j(t)$ into $\mathcal{C}_i(t)$
for $x_k(\tau) \in \mathcal{C}_j(t)$ **do**
    if $x_k \notin \mathcal{C}_i(t)$ **then**
        Add $x_k(\tau)$ into $\mathcal{C}_i(t)$
    else
        Retrieve $x_k(\tau') \in \mathcal{C}_i(t)$
        if $\tau > \tau'$ **then**
            Replace $x_k(\tau)$ with $x_k(\tau')$ into $\mathcal{C}_i(t)$
        end if
    end if
end for
$\mathcal{C}_i(t+1) \leftarrow$ GROUP_PRUNE_CACHE($\mathcal{C}_i(t), \mathcal{R}, \mathcal{A}$)

Output: $\mathcal{C}_i(t+1)$

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

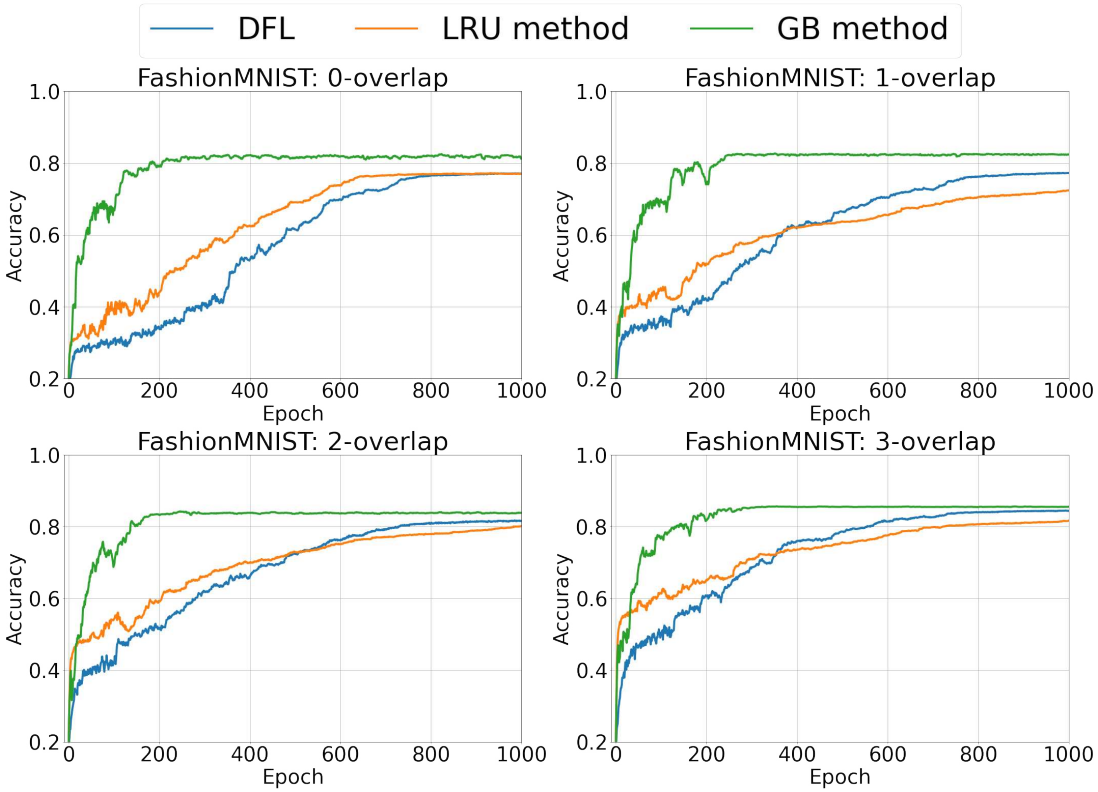

Figure 6: Group-based LRU Cache Update Performance under Different Data Distribution Overlaps.

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.

5.6 Discussions

Decentralized FL (DFL) has been increasingly applied in vehicular networks, leveraging existing frameworks like vehicle-to-vehicle (V2V) communication [25]. V2V FL facilitates knowledge sharing among vehicles and has been explored in various studies [2, 4, 15, 17, 18, 24]. Samarakoon et al. [17] studied optimized joint power and resource allocation for ultra-reliable low-latency communication (URLLC) using FL. Su et al. [18] 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 et al. [21]. 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.

6 Related Work

Decentralized FL (DFL) has been increasingly applied in vehicular networks, leveraging existing frameworks like vehicle-to-vehicle (V2V) communication [25]. V2V FL facilitates knowledge sharing among vehicles and has been explored in various studies [2, 4, 15, 17, 18, 24]. Samarakoon et al. [17] studied optimized joint power and resource allocation for ultra-reliable low-latency communication (URLLC)

using FL.

Su et al. [18] 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 et al. [21]. 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.

7 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.

References

[1] Fan Bai, Narayanan Sadagopan, and Ahmed Helmy. Important: A framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, volume 2, pages 825–835. IEEE, 2003.

[2] Luca Barbieri, Stefano Savazzi, Mattia Brambilla, and Monica Nicoli. Decentralized federated learning for extended sensing in 6g connected vehicles. *Vehicular Communications*, 33:100396, 2022.

[3] Scott Burleigh, Adrian Hooke, Leigh Torgerson, Kevin Fall, Vint Cerf, Bob Durst, Keith Scott, and Howard Weiss. Delay-tolerant networking: an approach to interplanetary internet. *IEEE Communications Magazine*, 41(6):128–136, 2003.

[4] Jin-Hua Chen, Min-Rong Chen, Guo-Qiang Zeng, and Jia-Si Weng. Bdfl: A byzantine-fault-tolerance decentralized federated learning method for autonomous vehicle. *IEEE Transactions on Vehicular Technology*, 70(9):8639–8652, 2021.

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

[6] Kevin Fall. A delay-tolerant network architecture for challenged internets. SIGCOMM '03, page 27–34, New York, NY, USA, 2003.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

[9] Chen Li, Xiaoyu Wang, Tongyu Zong, Houwei Cao, and Yong Liu. Predictive edge caching through deep mining of sequential patterns in user content retrievals. *Computer Networks*, 233:109866, 2023.

[10] Ji Liu, Jizhou Huang, Yang Zhou, Xuhong Li, Shilei Ji, Haoyi Xiong, and Dejing Dou. From distributed machine learning to federated learning: A survey. *Knowledge and Information Systems*, 64(4):885–917, 2022.

[11] Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys & Tutorials*, 25(4):2983–3013, 2023. doi: 10.1109/COMST.2023.3315746.

[12] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[13] Solmaz Niknam, Harpreet S Dhillon, and Jeffrey H Reed. Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine*, 58(6):46–51, 2020.

[14] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[15] Shiva Raj Pokhrel and Jinho Choi. A decentralized federated learning approach for connected autonomous vehicles. In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6. IEEE, 2020.

[16] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[17] Sumudu Samarakoon, Mehdi Bennis, Walid Saad, and Mérouane Debbah. Distributed federated learning for ultra-reliable low-latency vehicular communications. *IEEE Transactions on Communications*, 68(2):1146–1159, 2019.

[18] Dongyuan Su, Yipeng Zhou, and Laizhong Cui. Boost decentralized federated learning in vehicular networks by diversifying data sources. In *2022 IEEE 30th International Conference on Network Protocols (ICNP)*, pages 1–11. IEEE, 2022.

[19] Tao Sun, Dongsheng Li, and Bao Wang. Decentralized federated averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[20] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL `https://arxiv.org/abs/1708.07747`.

[21] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Asynchronous federated optimization. *CoRR*, abs/1903.03934, 2019. URL `http://arxiv.org/abs/1903.03934`.

[22] Guojun Xiong, Gang Yan, Shiqiang Wang, and Jian Li. Deprl: Achieving linear convergence speedup in personalized decentralized learning with shared representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 16103–16111, 2024.

[23] Haibo Yang, Minghong Fang, and Jia Liu. Achieving linear speedup with partial worker participation in non-iid federated learning. *Proceedings of ICLR*, 2021.

[24] Zhengxin Yu, Jia Hu, Geyong Min, Han Xu, and Jed Mills. 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)*, pages 601–608. IEEE, 2020.

[25] Liangqi Yuan, Ziran Wang, Lichao Sun, S Yu Philip, and Christopher G Brinton. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, 2024.

[26] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. *Advances in neural information processing systems*, 32, 2019.

A 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 4 (Smoothness)**.** *A differentiable function f is L-smooth if for $\forall x, y,\ f(y) - f(x) \leq \langle \nabla f(x), y - x\rangle + \frac{L}{2}||y - x||^2$, where $L > 0$.*

Definition 5 (Bounded Variance)**.** *There exists a constant $\varsigma > 0$ such that the global variability of the local gradient of the loss function is bounded $||\nabla F_j(x) - \nabla F(x)||^2 \leq \varsigma^2, \forall j \in [N], x \in \mathbb{R}^d$.*

Definition 6 (L-Lipschitz Continuous Gradient)**.** *There exists a constant $L > 0$, such that $||\nabla F(x) - \nabla F(x')|| \leq L'||x - x'||, \forall x, x' \in \mathbb{R}^d$.*

Theorem 5. *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 $\forall z \sim \mathcal{D}^i, ||\nabla f(x; z)||^2 \leq V, ||\nabla g_{x'}(x; z)||^2 \leq V, \forall x' \in \mathbb{R}^d$, and ∇F satisfies L-Lipschitz Continuous Gradient. For any small constant $\epsilon > 0$, if we take $\rho > 0$, and satisfying $-(1 + 2\rho + \epsilon)V + (\rho^2 - \frac{\rho}{2})||x(t, k-1) - x(t)||^2 \geq 0, \forall x(t, k-1), x(t)$, after T global epochs, Algorithm 1 converges to a critical point:*

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

A.1 Proof

Similar to Theorem 1 in [21], we can get the boundary before and after K times local updates on j-th device, $\forall j \in [N]$:

$$\begin{aligned}\mathbb{E}[F(\tilde{x}_j(t)) - F(x_j(t))] &= \mathbb{E}[F(x_j(t, K)) - F(x_j(t, 0))] \\ &\leq -\eta\epsilon \sum_{k=0}^{K-1} \mathbb{E}||\nabla F(x_j(t, k))||^2 + \eta^2 \mathcal{O}(\rho K^3 V). \end{aligned} \tag{5}$$

For any epoch t, we find the index $M(t)$ of the agent whose model is the "worst", i.e., $M(t) = \arg\max_{j\in[N]}\{F(x_j(t))\}$, then we can get,

$$\begin{aligned}\mathbb{E}[F(x_i(t+1)) - F(x_*)] \leq & \mathbb{E}[F(x_{M(t+1)}(t+1)) - F(x^*)] \\ \leq & \mathbb{E}\Big[F\Big(\frac{1}{|C_{M(t+1)}(t)|}\sum_{j,\tau\in C_{M(t+1)}(t)} \tilde{x}_j(\tau)\Big)\Big] - F(x^*) \\ \leq & \mathbb{E}\Big[\frac{1}{|C_{M(t+1)}(t)|}\sum_{j,\tau\in C_{M(t+1)}(t)} F(\tilde{x}_j(\tau))\Big] - F(x^*) \\ \leq & \mathbb{E}\Big[\frac{1}{|C_{M(t+1)}(t)|}\sum_{j,\tau\in C_{M(t+1)}(t)} F(x_j(\tau))\Big] - F(x^*) + \eta^2\mathcal{O}(\rho K^3 V) \\ & - \frac{\epsilon\eta}{|C_{M(t+1)}(t)|}\sum_{j,\tau\in C_{M(t+1)}(t)}\sum_{k=0}^{K-1}\mathbb{E}||\nabla F(x_j(\tau, k))||^2 \\ \leq & \mathbb{E}\Big[\frac{1}{|C_{M(t+1)}(t)|}\sum_{\tau=t-\tau_{max}+1}^{t}\Big(\sum_{\substack{x_j(\tau)\in C_{M(t)}(t)\\ x_j(\tau)\neq x_{M(\tau)}(\tau)}} F(x_j(\tau)) + \sum_{x_{M(\tau)}(\tau)\in C_{M(t)}(t)} F(x_{M(\tau)}(\tau))\Big)\Big]\end{aligned}$$

$$- \nabla F - F(x^*) + \eta^2 \mathcal{O}(\rho K^3 V). \tag{6}$$

here $\nabla F = \frac{\epsilon\eta}{|C_{M(t+1)}(t)|} \sum_{j,\tau \in C_{M(t+1)}(t)} \sum_{k=0}^{K-1} \mathbb{E}||\nabla F(x_j(\tau,k))||^2$.
We can easy know, for given $\tau, \forall j \neq M(\tau)$, we have $F(x_{M(\tau)}(\tau)) > F(x_j(\tau))$, then,

$$\mathbb{E}\Big[\sum_{\substack{x_j(\tau)\in C_{M(t)}(t)\\ x_j(\tau)\neq x_{M(\tau)}(\tau)}} F(x_j(\tau))\Big] = \Big(\sum_{\substack{x_j(\tau)\in C_{M(t)}(t)\\ x_j(\tau)\neq x_{M(\tau)}(\tau)}} \cdot 1\Big) h(\tau) F(x_{M(\tau)}(\tau)). \tag{7}$$

Here $0 < h(\tau) < 1$, so we rearrange the equation above, we can get:

$$\begin{aligned}
&\mathbb{E}[F(x_{M(t+1)}(t+1)) - F(x^*)] \\
\leq &\mathbb{E}\Big[\sum_{\tau=t}^{t-\tau_{max}+1} \frac{\sum_{\substack{x_j(\tau)\in C_{M(t)}(t)\\ x_j(\tau)\neq x_{M(\tau)}(\tau)}} \cdot h(\tau) + \sum_{x_{M(\tau)}(\tau)\in C_{M(t)}(t)} \cdot 1}{|C_{M(t+1)}(t)|} F(x_{M(\tau)}(\tau))\Big] \\
&- \nabla F - F(x^*) + \eta^2 \mathcal{O}(\rho K^3 V) \\
= &\mathbb{E}\Big[\sum_{\tau=t}^{t-\tau_{max}+1} \frac{\sum_{\substack{x_j(\tau)\in C_{M(t)}(t)\\ x_j(\tau)\neq x_{M(\tau)}(\tau)}} \cdot h(\tau) + \sum_{x_{M(\tau)}(\tau)\in C_{M(t)}(t)} \cdot 1}{|C_{M(t+1)}(t)|} F(x_{M(\tau)}(\tau))\Big] \\
&- \nabla F - F(x^*) + \eta^2 \mathcal{O}(\rho K^3 V) \\
= &\mathbb{E}\Big[\sum_{\tau=t}^{t-\tau_{max}+1} \frac{\sum_{\substack{x_j(\tau)\in C_{M(t)}(t)\\ x_j(\tau)\neq x_{M(\tau)}(\tau)}} \cdot h(\tau) + \sum_{x_{M(\tau)}(\tau)\in C_{M(t)}(t)} \cdot 1}{\sum_{\tau=t}^{t-\tau_{max}+1} (\sum_{\substack{x_j(\tau)\in C_{M(t)}(t)\\ x_j(\tau)\neq x_{M(\tau)}(\tau)}} \cdot 1 + \sum_{x_{M(\tau)}(\tau)\in C_{M(t)}(t)} \cdot 1)} F(x_{M(\tau)}(\tau))\Big] \\
&- \nabla F - F(x^*) + \eta^2 \mathcal{O}(\rho K^3 V).
\end{aligned} \tag{8}$$

We define the "worst" model on all agents over the time period of $[t - \tau_{max} + 1, t]$ as

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

Then we can get,

$$\begin{aligned}
&\mathbb{E}[F(x_{M(t+1)}(t+1)) - F(x^*)] \\
\leq &\mathbb{E}\Big[\sum_{\tau=t}^{t-\tau_{max}+1} \frac{\sum_{\substack{x_j(\tau)\in C_{M(t)}(t)\\ x_j(\tau)\neq x_{M(\tau)}(\tau)}} \cdot h(\tau) + \sum_{x_{M(\tau)}(\tau)\in C_{M(t)}(t)} \cdot 1}{\sum_{\tau=t}^{t-\tau_{max}+1} (\sum_{\substack{x_j(\tau)\in C_{M(t)}(t)\\ x_j(\tau)\neq x_{M(\tau)}(\tau)}} \cdot 1 + \sum_{x_{M(\tau)}(\tau)\in C_{M(t)}(t)} \cdot 1)} \\
&\cdot F(x_{(\mathcal{T}(t,\tau_{max}))}(\mathcal{T}(t, \tau_{max})))\Big] - \nabla F - F(x^*) + \eta^2 \mathcal{O}(\rho K^3 V) \\
= &\mathbb{E}\Big[\Big(1 - \sum_{\tau=t}^{t-\tau_{max}+1} \frac{\sum_{\substack{x_j(\tau)\in C_{M(t)}(t)\\ x_j(\tau)\neq x_{M(\tau)}(\tau)}} \cdot (1 - h(\tau))}{|C_{M(t+1)}(t)|}\Big) F(x_{(\mathcal{T}(t,\tau_{max}))}(\mathcal{T}(t, \tau_{max})))\Big] \\
&- \nabla F - F(x^*) + \eta^2 \mathcal{O}(\rho K^3 V) \\
= &\mathbb{E}\Big[\Big(1 - \xi(t, \tau_{max})\Big) F(x_{(\mathcal{T}(t,\tau_{max}))}(\mathcal{T}(t, \tau_{max})))\Big] - \nabla F - F(x^*) + \eta^2 \mathcal{O}(\rho K^3 V) \\
\leq &\mathbb{E}\Big[F(x_{(\mathcal{T}(t,\tau_{max}))}(\mathcal{T}(t, \tau_{max})))\Big] - \nabla F - F(x^*) + \eta^2 \mathcal{O}(\rho K^3 V).
\end{aligned} \tag{9}$$

Here, $\xi(t, \tau_{max}) = \sum_{\tau=t}^{t-\tau_{max}+1} \frac{\sum_{\substack{x_j(\tau)\in C_{M(t)}(t)\\ x_j(\tau)\neq x_{M(\tau)}(\tau)}} \cdot (1-h(\tau))}{|C_{M(t+1)}(t)|} \in [0, 1)$, so we have,

$$\mathbb{E}[F(x_{M(t+1)}(t+1)) - F(x_{(\mathcal{T}(t,\tau_{max}))}(\mathcal{T}(t, \tau_{max})))] \leq -\nabla F + \eta^2 \mathcal{O}(\rho K^3 V). \tag{10}$$

Then,

$$\begin{aligned}
&\mathbb{E}[F(x_{M(t+1)}(t+1)) - F(x_{(\mathcal{T}(t,\tau_{max}))}(\mathcal{T}(t,\tau_{max})))] \\
&\quad \leq -\nabla F + \eta^2 \mathcal{O}(\rho K^3 V) \\
&\quad = -\frac{\epsilon\eta}{|C_{M(t+1)}(t)|} \sum_{j,\tau \in C_{M(t+1)}(t)} \sum_{k=0}^{K-1} ||\nabla F(x_j(\tau,k))||^2 + \eta^2 \mathcal{O}(\rho K^3 V). \qquad (11)
\end{aligned}$$

By taking use of the property of L-Lipschitz Continuous Gradient, we can get,

$$\begin{aligned}
||\nabla F(x_j(\tau,k)) - \nabla F(x_j(\tau,k-1))|| &\leq L'||x_j(\tau,k) - x_j(\tau,k-1)|| \\
&\leq \eta L'||\nabla g_{x_j(\tau)}(x_j(\tau,k-1))|| \\
&\leq \eta L'\sqrt{V}. \qquad (12)
\end{aligned}$$

Then, we can get, $\forall k \in [1,K]$, $||\nabla F(x_j(\tau,k))||^2 \geq ||\nabla F(x_j(\tau,k-1))||^2 + \eta^2 L'^2 V$, then

$$||\nabla F(x_j(\tau,k))||^2 \geq ||\nabla F(x_j(\tau))||^2 + \eta^2 L'^2 kV. \qquad (13)$$

So we have,

$$\begin{aligned}
&\sum_{j,\tau \in C_{M(t+1)}(t)} \sum_{k=0}^{K-1} ||\nabla F(x_j(\tau,k))||^2 \\
&\qquad \geq \sum_{j,\tau \in C_{M(t+1)}(t)} K||\nabla F(x_j(\tau))||^2 + \eta^2 L'^2 K(K-1)V \\
&\qquad \geq \sum_{j,\tau \in C_{M(t+1)}(t)} K||\nabla F(x_j(\tau))||^2 + \eta^2 \mathcal{O}(K^2 V). \qquad (14)
\end{aligned}$$

We assume the ratio of gradient expectation on the global distribution to gradient expectation on local data distribution of any device is bounded, i.e.,

$$\mathbb{E}[||\nabla F(x_j(t))||^2] \geq C_1 \mathbb{E}[||\nabla F(x(t))||^2], \qquad (15)$$

So we get,

$$\begin{aligned}
&\sum_{j,\tau \in C_{M(t+1)}(t)} \sum_{k=0}^{K-1} \mathbb{E}[||\nabla F(x_j(\tau,k))||^2] \\
&\qquad \geq C_1 K |C_{M(t+1)}(t)| \min_{\tau=t-\tau_{max}+1}^{t} \mathbb{E}[||\nabla F(x(\tau))||^2] \\
&\qquad\quad + \eta^2 |C_{M(t+1)}(t)| \mathcal{O}(K^2 V), \qquad (16)
\end{aligned}$$

Inequality (11) becomes:

$$\begin{aligned}
&\mathbb{E}[F(x_{M(t+1)}(t+1)) - F(x_{(\mathcal{T}(t,\tau_{max}))}(\mathcal{T}(t,\tau_{max})))] \\
&\quad \leq -\epsilon\eta C_1 K \min_{\tau=t-\tau_{max}+1}^{t} ||\nabla F(x(\tau))||^2 + \eta^2 \mathcal{O}(\rho K^3 V). \qquad (17)
\end{aligned}$$

By rearranging the terms, we have,

$$\begin{aligned}
\min_{\tau=t-\tau_{max}+1}^{t} ||\nabla F(x(\tau))||^2 \leq &(\frac{1}{\epsilon\eta C_1 K}) \mathbb{E}[F(x_{(\mathcal{T}(t,\tau_{max}))}(\mathcal{T}(t,\tau_{max}))) \\
&- F(x_{M(t+1)}(t+1))] + \mathcal{O}(\frac{\eta\rho K^2 V}{C_1 \epsilon}). \qquad (18)
\end{aligned}$$

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

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

From the equation above, we can also get the inequality about N_T,

$$\frac{T}{\tau_{max}} \leq N_T \leq T. \tag{19}$$

Applying inequality (18) at all time instances $\{T'_0, T'_1, T'_2, ..., T'_{N_T}\}$, after T global epochs, we have,

$$\begin{aligned} \min_{t=0}^{T-1} \mathbb{E}[||\nabla F(x(t))||^2] \leq & \frac{1}{N_T} \sum_{t=T'_0}^{T'_N} \min_{\tau=t-\tau_{max}+1}^{t} ||\nabla F(x(\tau))||^2 \\ \leq & \frac{1}{\epsilon\eta C_1 K N_T} \sum_{t=T'_0}^{T'_N} \mathbb{E}[F(x_{M_{T(t,\tau_{max})}}(T(t,\tau_{max}))) - F(x_{M(t+1)}(t+1))] + \mathcal{O}(\frac{\eta\rho K^2 V}{\epsilon C_1}) \\ \leq & \frac{1}{\epsilon\eta C_1 K N_T} \mathbb{E}[F(x_{M(T'_0)}(T'_0)) - F(x_{M(T'_{N_T}+1)}(T'_{N_T}+1))] + \mathcal{O}(\frac{\eta\rho K^2 V}{\epsilon C_1}) \\ \leq & \frac{1}{\epsilon\eta C_1 K N_T} \mathbb{E}[F(x_{M(0)}(0)) - F(x_{M(T)}(T))] + \mathcal{O}(\frac{\eta\rho K^2 V}{\epsilon C_1}) \\ \leq & \frac{1}{\epsilon\eta C_1 K N_T} \mathbb{E}[F(x(0)) - F(x_{M(T)}(T))] + \mathcal{O}(\frac{\eta\rho K^2 V}{\epsilon C_1}) \\ \leq & \frac{\tau_{max}}{\epsilon\eta C_1 K T} \mathbb{E}[F(x(0)) - F(x_{M(T)}(T))] + \mathcal{O}(\frac{\eta\rho K^2 V}{\epsilon C_1}) \\ \leq & \frac{\tau_{max}}{\epsilon\eta C_1 K T} \mathbb{E}[F(x(0)) - F(x(T)] + \mathcal{O}(\frac{\eta\rho K^2 V}{\epsilon C_1}) \\ \leq & \mathcal{O}(\frac{\tau_{max}}{\epsilon\eta C_1 K T}) + \mathcal{O}(\frac{\eta\rho K^2}{\epsilon C_1}). \end{aligned} \tag{20}$$

With the results in (20) and by leveraging Theorem 1 in Yang et al. [23], Algorithm 1 converges to a critical point after T global epochs.

B Experimental details

B.1 Data distribution in Group-based LRU Cache Experiment

B.1.1 n-overlap

For MNIST, FashionMNIST with 10 classes labels, we allocate labels into 3 groups as:
Non-overlap:
Area 1: (0,1,2,3), Area 2: (4,5,6), Area 3: (7,8,9)
1-overlap:
Area 1: (9,0,1,2,3), Area 2: (3,4,5,6), Area 3: (6,7,8,9)
2-overlap:
Area 1: (8,9,0,1,2,3), Area 2: (2,3,4,5,6), Area 3: (5,6,7,8,9)
3-overlap:
Area 1: (7,8,9,0,1,2,3), Area 2: (1,2,3,4,5,6), Area 3: (4,5,6,7,8,9)

B.2 Computing infrastructure

For MNIST and FashionMNIST experiments, we use 10 computing nodes, each with 10 CPUs, model: Lenovo SR670, to simulate DFL on 100 vehicles. CIFAR-10 results are obtained from 1 compute node with 5 CPUs and 1 A100 NVIDIA GPU, model: SD650-N V2. They are all using Ubuntu-20.04.1 system with pre-installed openmpi/intel/4.0.5 module, and 100 Gb/s network speed.

B.3 Results with variation

As our metric is the average test accuracy over 100 devices, we also provide the variation of the average test accuracy in our experimental results as following,

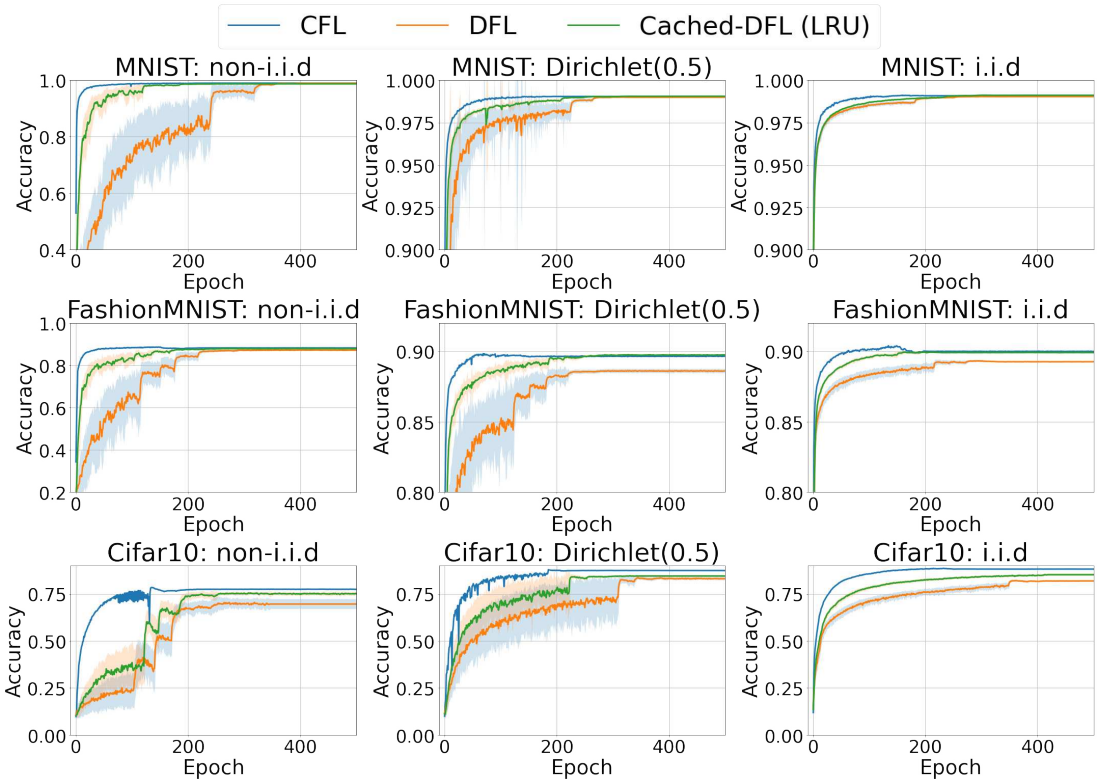

Figure 7: DFL with Caching vs. DFL without Caching with Variation.

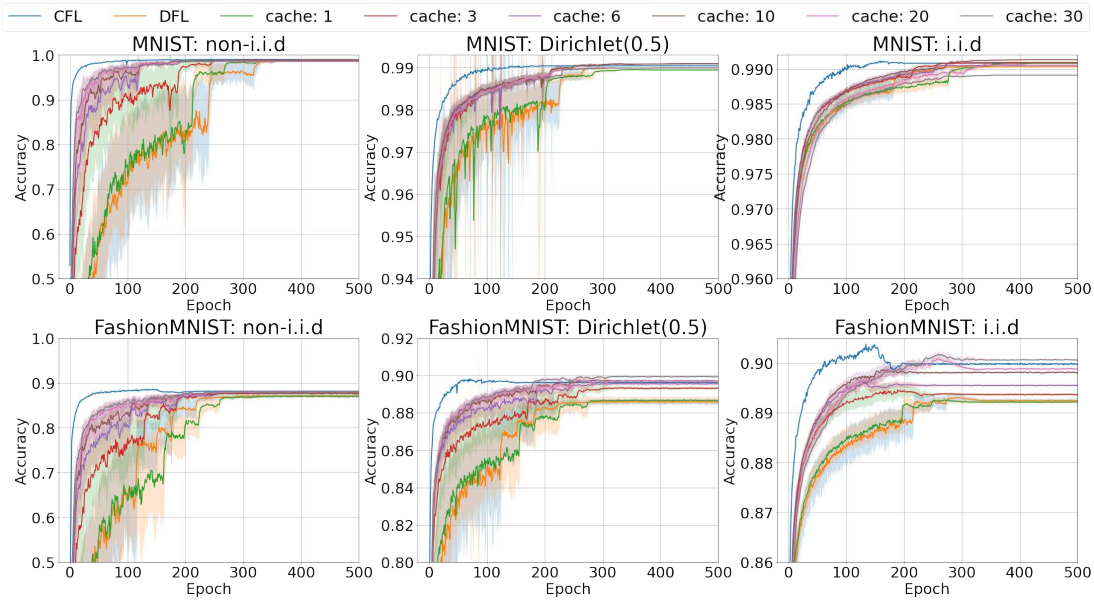

Figure 8: DFL with LRU at Different Cache Sizes with Variation.

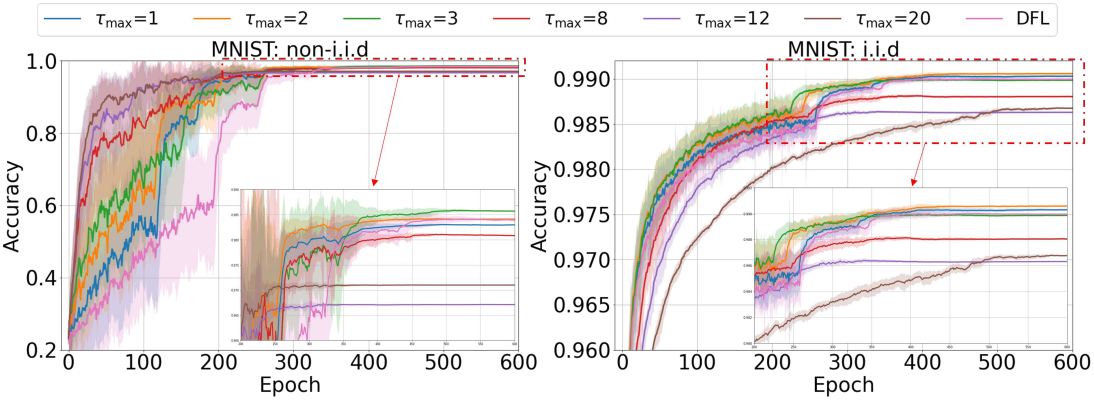

Figure 9: Impact of τ_{max} on Model Convergence with Variation.

Table 3: Average number and average age of cached models with different τ_{max} and different epoch time: 30s, 60s, 120s. Columns for different τ_{max}, and rows for different epoch time. There are four sub-rows for each epoch time, with the first two sub-rows be the average number of cached models and variation, the second two sub-rows be their average age and variation.

	τ_{max}	1	2	3	4	5	10	20
30s	num	0.8549	1.6841	2.6805	3.8584	5.2054	14.8520	44.8272
	var_n	0.0321	0.0903	0.1941	0.3902	0.6986	6.2270	42.8144
	age	0.0000	0.4904	1.0489	1.6385	2.2461	5.4381	11.5735
	var_a	0.0000	0.0052	0.0116	0.0203	0.0320	0.1577	1.0520
60s	num	1.6562	3.8227	6.4792	10.3889	14.0749	43.0518	90.2576
	var_n	0.0865	0.3594	0.9925	2.824	5.1801	32.8912	55.5829
	age	0.000	0.5593	1.1604	1.8075	2.4396	5.5832	9.7954
	var_a	0.0000	0.0035	0.0089	0.0188	0.0251	0.1448	0.8375
120s	num	3.6936	9.9149	18.5016	31.3576	35.1958	90.2469	98.5412
	var_n	0.3166	2.3825	5.9696	21.7408	56.8980	28.8958	29.9676
	age	0.0000	0.6189	1.2715	1.9194	1.5081	4.7131	5.2357
	var_a	0.0000	0.0028	0.0058	0.0212	0.9742	0.1249	0.1808

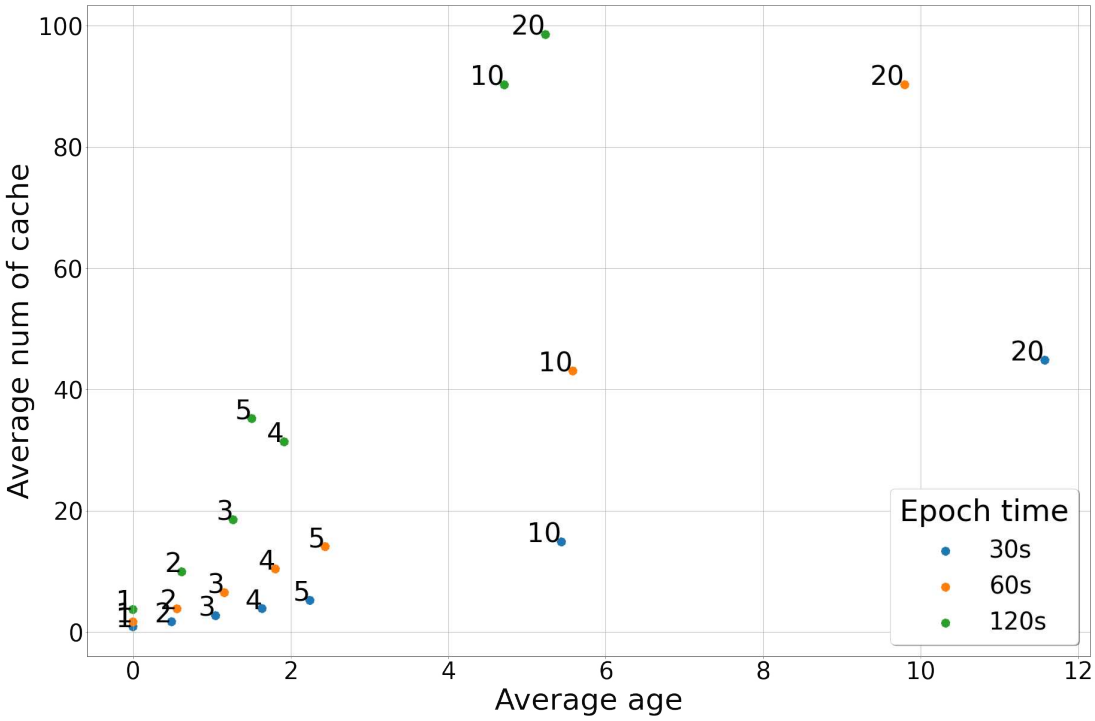

Figure 10: Relation Between Average Number and Average Age of Cached Models with Different τ_{max} (number close to the dot), with Unlimited Cache Size.

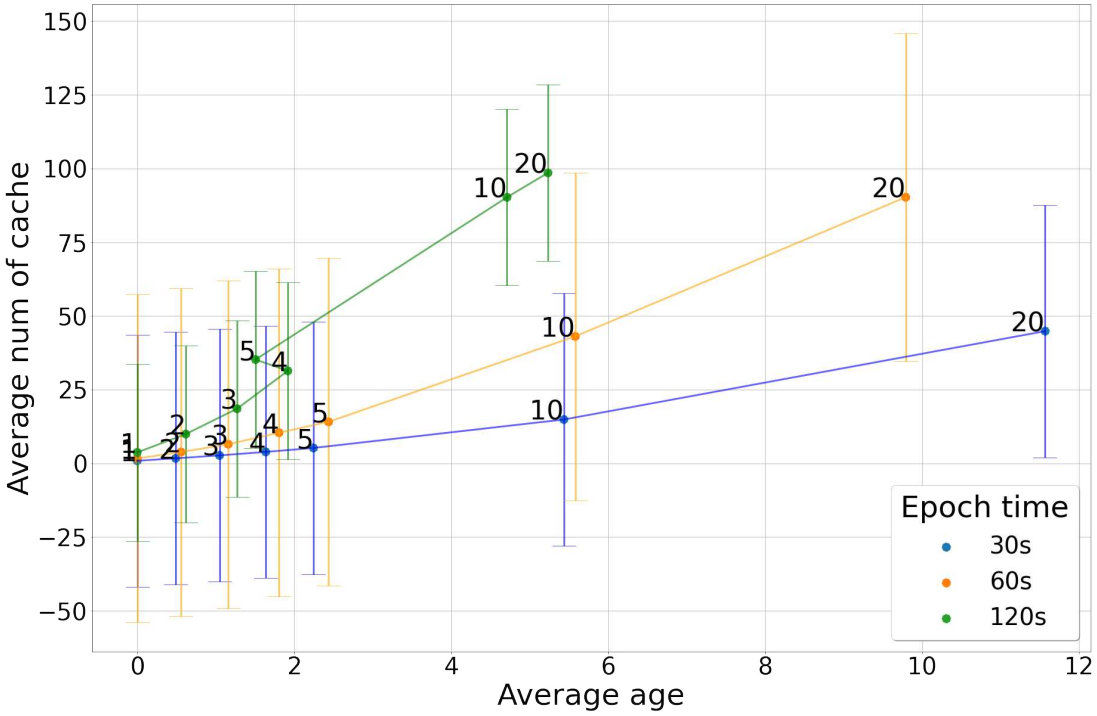

Figure 11: Relation Between Average Number and Average Age of Cached Models with Variation of Average Age.

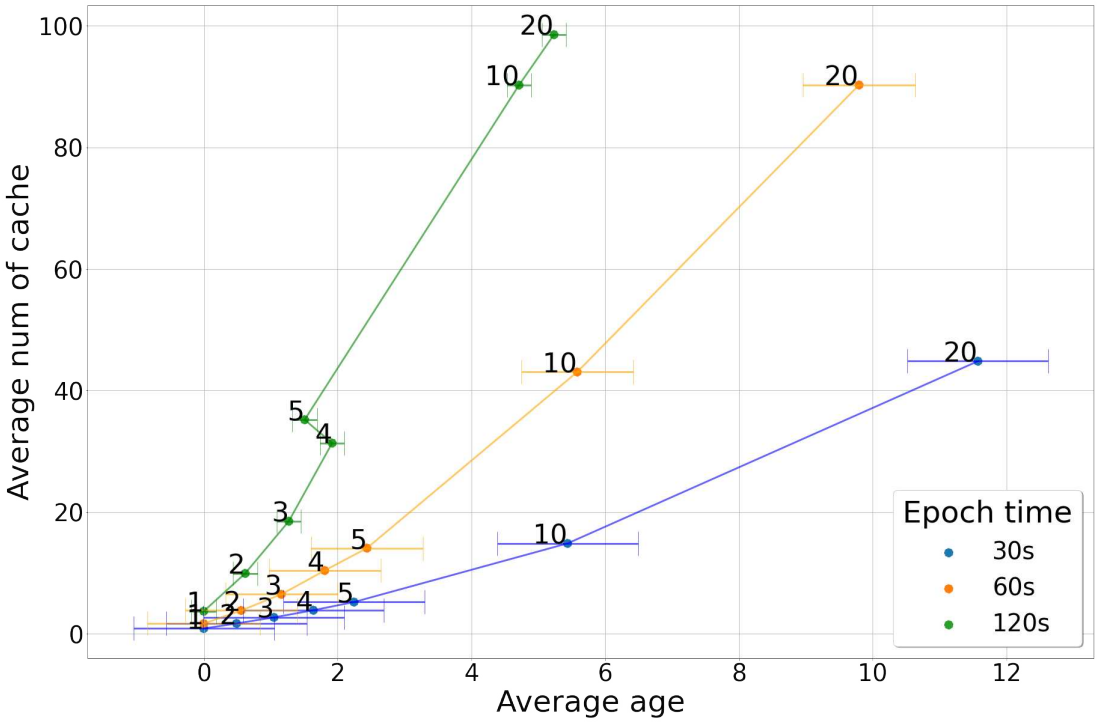

Figure 12: Relation Between Average Number and Average Age of Cached Models with Variation of Average Number.

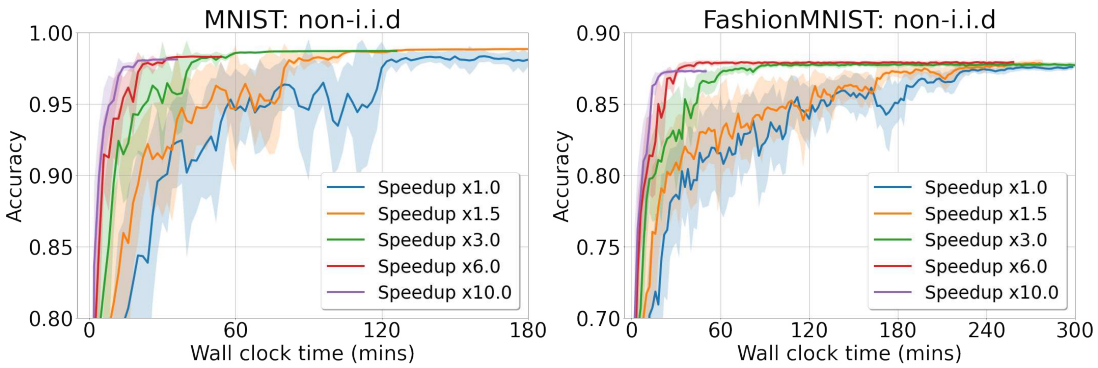

Figure 13: Convergence at Different Mobility Speed with Variation.

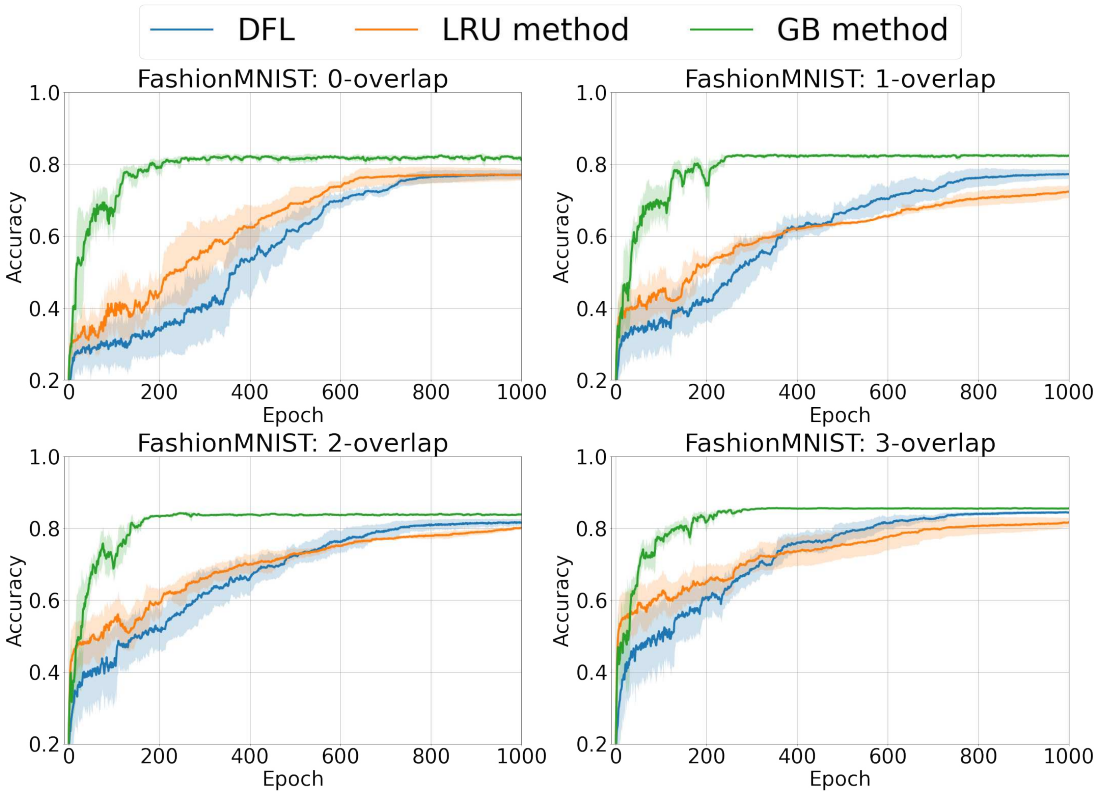

Figure 14: Group-based LRU Cache Update Performance under Different Data Distribution Overlaps with Variation.

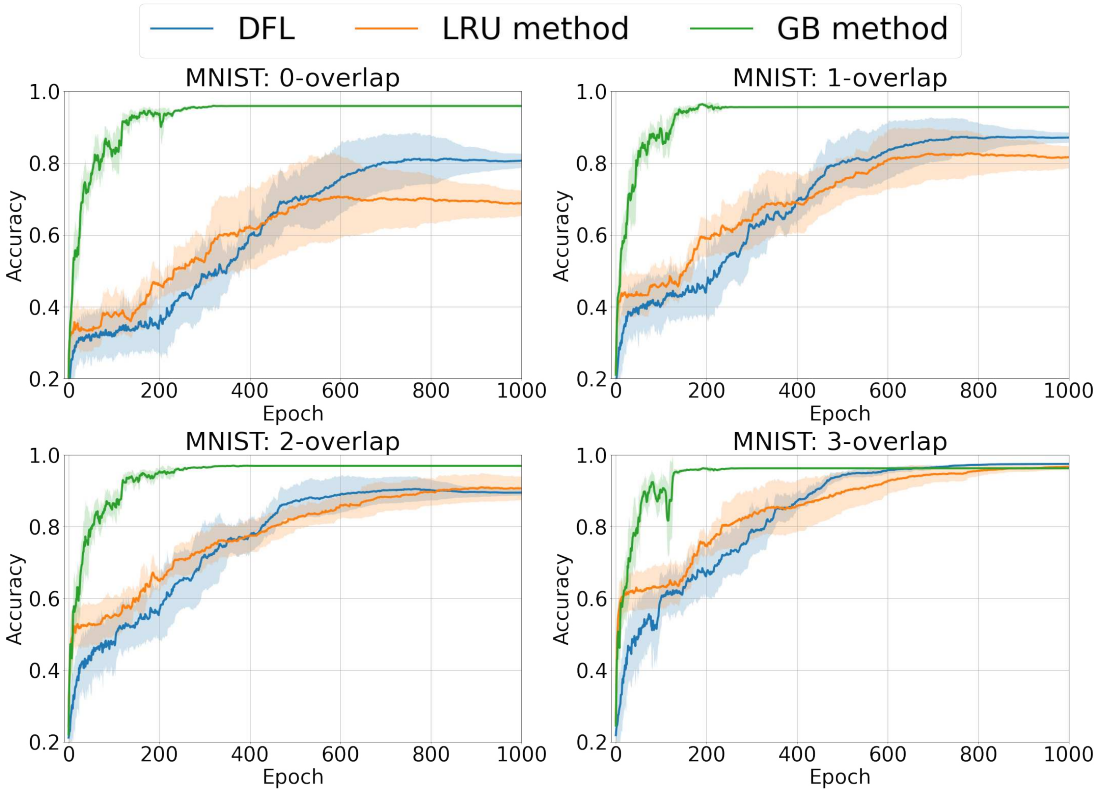

Figure 15: Group-based LRU Cache Update Performance under Different Data Distribution Overlaps with Variation.

B.4 Final (hyper-)parameters in experiments

learning rate $\eta = 0.1$, batch size = 64. 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. Without specific explanation, we take the following parameters as the final default parameters in our experiments. Local epoch $K = 10$, vehicle speed $v = 13.59m/s$, communication distance $100m$, communication interval between each epoch $120s$, $\tau_{max} = 10$, number of devices $N = 100$, cache size 10.

B.5 NN models

We choose the models from the classical repository [3]: we run convolutional neural network (CNN) as shown in Table 4, for MNIST, convolutional neural network (CNN) as shown in Table 5, for FashionMNIST. Inspired by [26], we choose ResNet-18 for Cifar-10, as shown in Table 6.

[3]Federated-Learning-PyTorch , Implementation of Communication-Efficient Learning of Deep Networks from Decentralized Data © 2024 GitHub, Inc.`https://github.com/AshwinRJ/Federated-Learning-PyTorch`.

Table 4: CNN Architecture for MNIST

Layer Type	**Size**
Convolution + ReLU	$5 \times 5 \times 10$
Max Pooling	2×2
Convolution + ReLU + Dropout	$5 \times 5 \times 20$
Dropout(2D)	$p = 0.5$
Max Pooling	2×2
Fully Connected + ReLU	320×50
Dropout	$p = 0.5$
Fully Connected	50×10

Table 5: CNN Architecture for FashionMNIST

Layer Type	**Size**
Convolution + BatchNorm + ReLU	$5 \times 5 \times 16$
Max Pooling	2×2
Convolution + BatchNorm + ReLU	$5 \times 5 \times 32$
Max Pooling	2×2
Fully Connected	$7 \times 7 \times 32 \times 10$

Table 6: ResNet-18 Architecture

Layer Type	**Output Size**
Convolution + BatchNorm + ReLU	$32 \times 32 \times 64$
Residual Block 1 (2x BasicBlock)	$32 \times 32 \times 64$
Residual Block 2 (2x BasicBlock)	$16 \times 16 \times 128$
Residual Block 3 (2x BasicBlock)	$8 \times 8 \times 256$
Residual Block 4 (2x BasicBlock)	$4 \times 4 \times 512$
Average Pooling	$1 \times 1 \times 512$
Fully Connected	$1 \times 1 \times 10$