Reproduction and Evaluation for Optimal Device Selection in Federated Learning using Deep Q-learning

Tian Liu, Yuting Cai, Cheng Niu

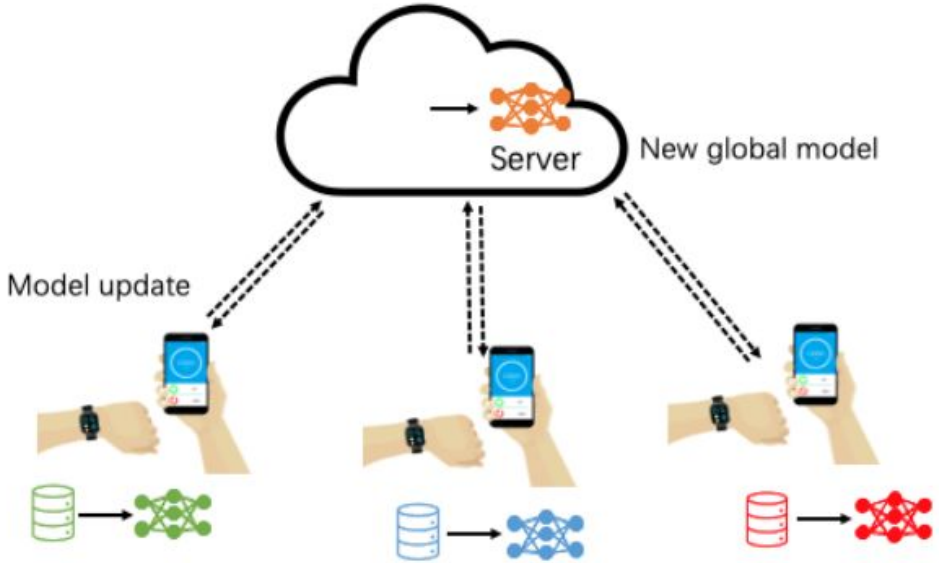
Federated Learning

TEXAS A&M UNIVERSITY

Traditional machine learning raises privacy concern.

In each round:

1. **Clients use local data to train and update the model locally.**
2. **Server aggregates the updates and send back new updated model.**
3. **Clients conduct more local training using received model.**



Problems & Motivation

TEXAS A&M UNIVERSITY

- **Having all devices to participate in the training is impractical**
  Just select portion of the devices

- **Selecting devices randomly will degrades learning performance and leat to slow convergence.**
  Non-IID (not independent and identically distributed) data introducing unbalanced dataset degrades learning performance

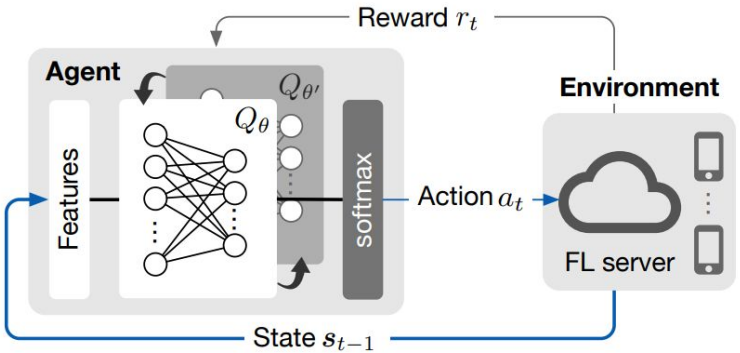
- **Project Objective**: reproduce the work by Wang et al. to use DQN for device selection

Optimizing Federated Learning on Non-IID Data with Reinforcement Learning

Hao Wang[1], Zakhary Kaplan[1], Di Niu[2] and Baochun Li[1]
[1]*University of Toronto*, {haowang, bli}@ece.utoronto.ca, zakhary.kaplan@mail.utoronto.ca [2]*University of Alberta*, dniu@ualberta.ca

MDP Definition

- **Action**: select 1 out of N devices during training, select k/N during testing
- **Action space**: total clients number
- **State**: updated weights of all clients and global model
- **State space**: (number of clients + 1(global model))* 100 (reduced by PCA)
- **Reward**: based on testing accuracy of global model on withheld testing data



$$r_t = \Xi^{(\omega_t - \Omega)} - 1, t = 1, \ldots, T$$

New reward function based on performance improvements:

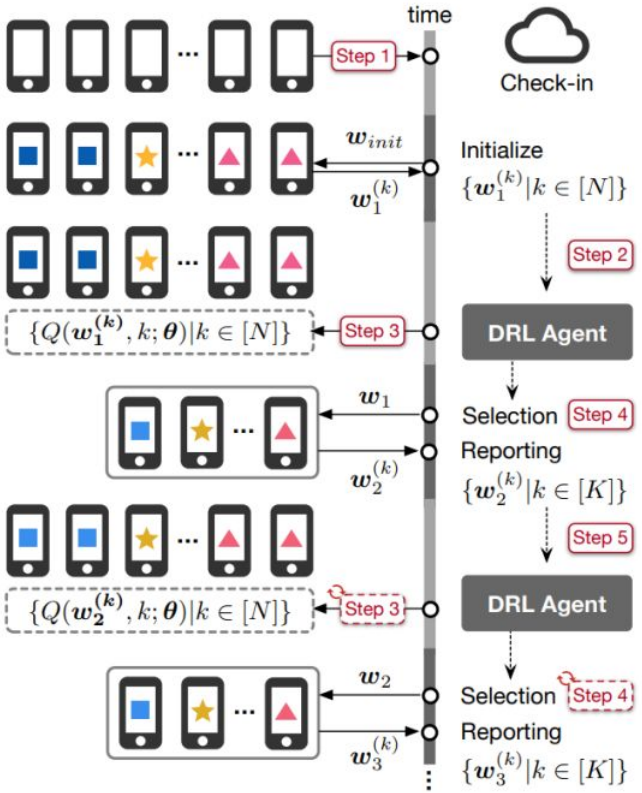
$$r_t = \begin{cases} \Xi^{(\omega_t - \omega_{t-1})} & \omega_t > \omega_{t-1} \\ -\Xi^{(\omega_{t-1} - \omega_t)} & \omega_t < \omega_{t-1} \end{cases}$$

Device Selection Workflow via DQN

TEXAS A&M UNIVERSITY

In each episode:

1. Initialize randomly weights for available N clients' model.
2. Train N clients' local model. Return to FL server and aggregate new global model.
3. Generate Q value approximator for all devices.
4. Select K devices based on top-K Q values. Download latest global model, train new local model and return.
5. Aggregate new global model through return models.
6. Update Q value approximator based on the new model performance.
7. Repeat to step 4, until rounds/accuracy meets target.



Experiments Setup

DQN Train Server:
Train the DQN model

DQN Evaluation server:
- Select 10 out of 100
- Select 4 out of 20

Compare with Baselines:
- FedAvg (random selection)
- K-means clustering
- K-center clustering

Linux server

64 Intel(R) Xeon(R) Silver 4313 CPUs @2.4GHz

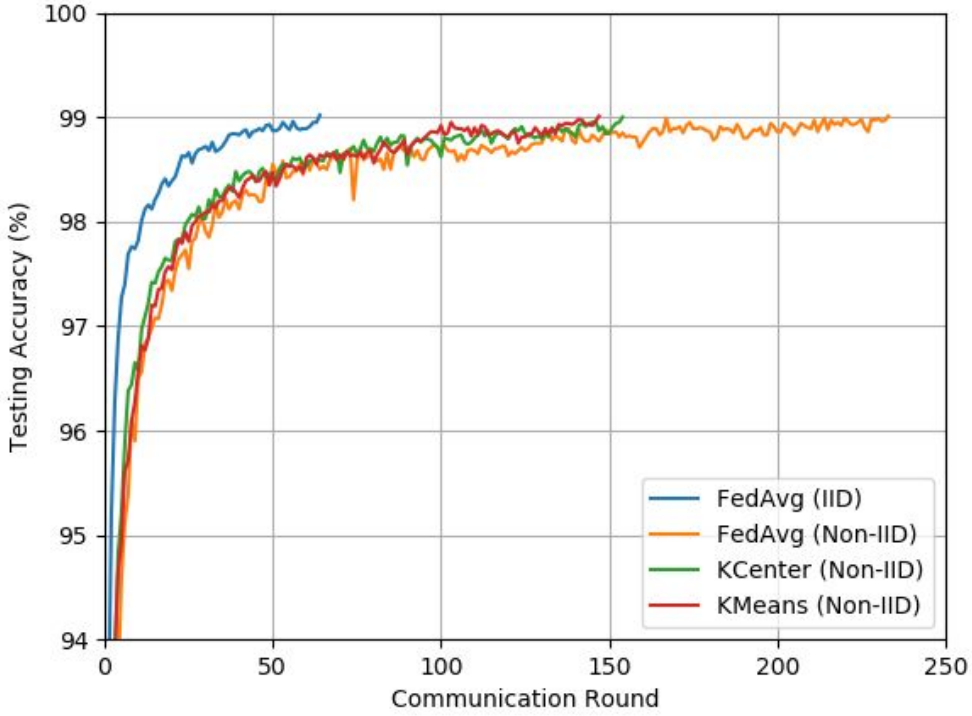
3 Nvidia A30 GPUs (24GB each)

Training time

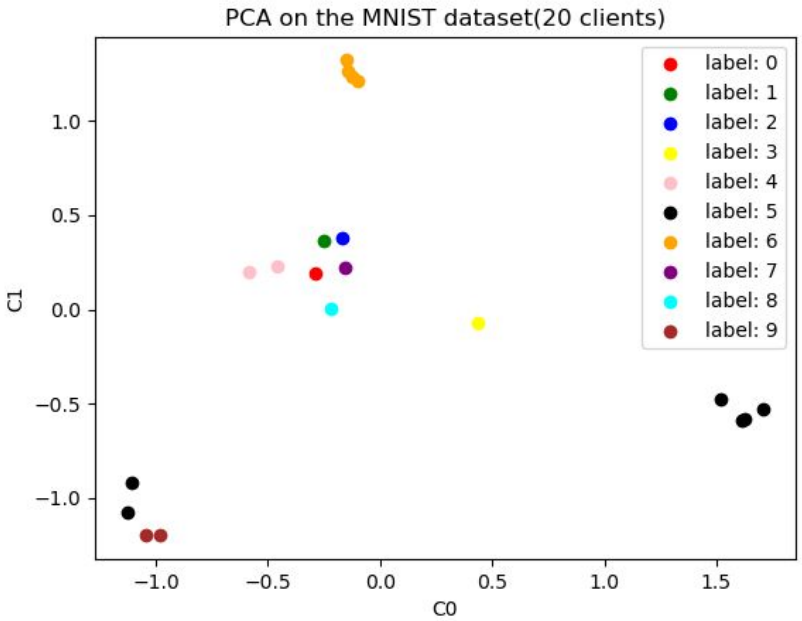
10 minutes for 1 episode

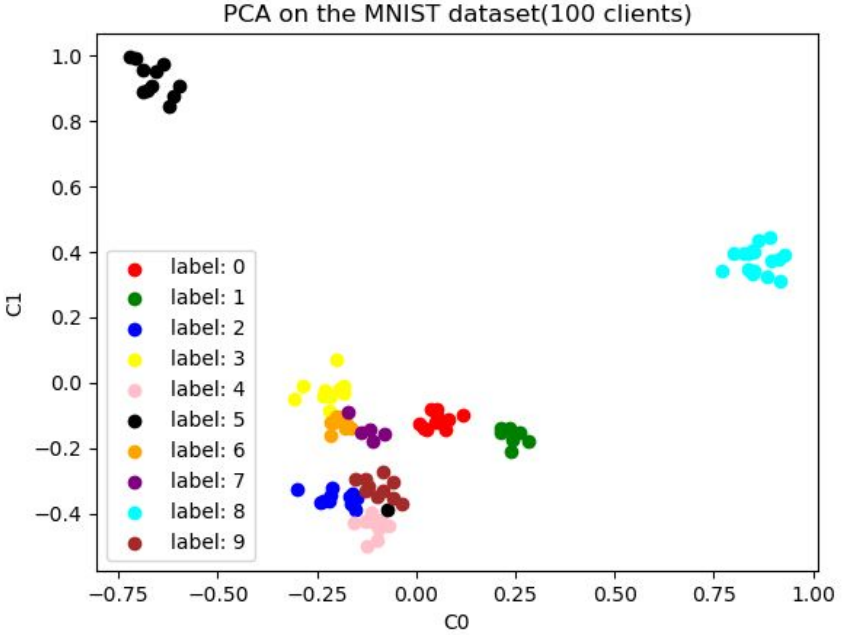
30+ hours total for 1 train

Federated Learning on Non-IID Data

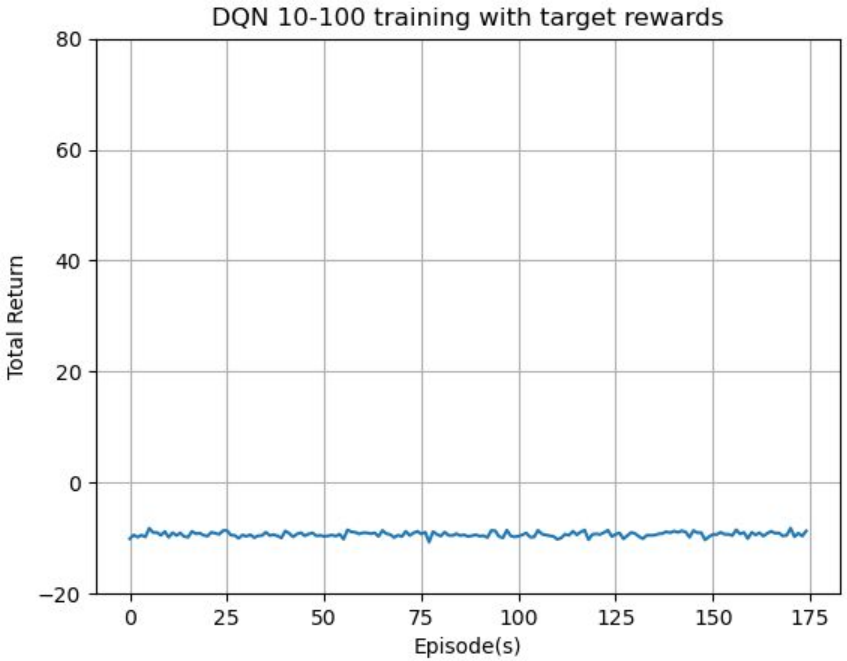


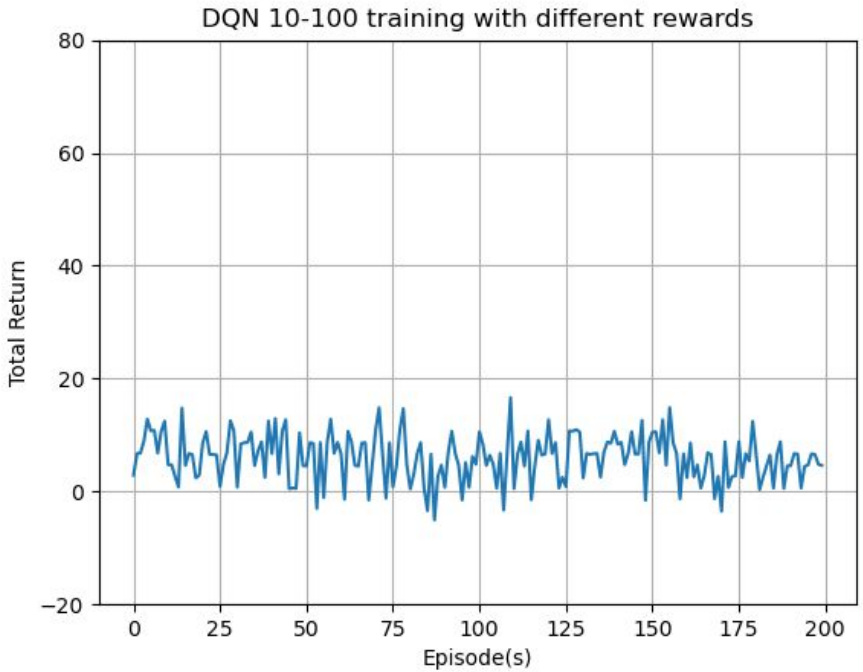
PCA for State Compression



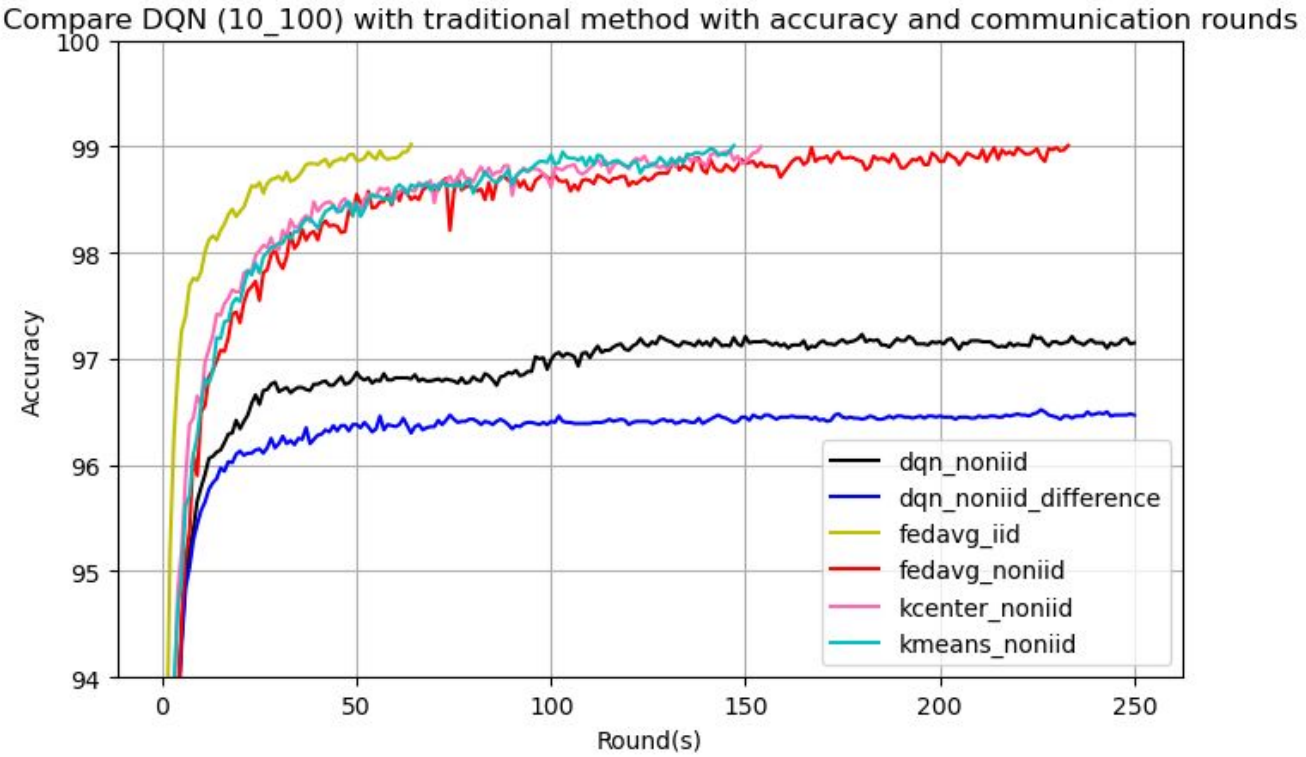


Compare DQN with new rewards





DQN Compare with Baseline



Result Analysis

Why DQN did not work well?

1. Reward function doesn't reflect actual return of DQN training accurately. Testing accuracy will increase with training rounds regardless of selecting which device

2. Selecting only 1 client to update Q-value each round during training leads to highly biased action selection

3. DQN is not suitable for problems which has **strong dependency** between actions. (such as the Montezuma and Double Dunk games as mentioned in DQN paper)

4. DQN agent doesn't explore enough due to limited memory buffer size and large action and state space (selecting 10 out 100 gives 1.73E13 combinations)

(Many other researchers shared the same experiences and thoughts, see https://github.com/iQua/flsim/issues/7)

Conclusions

- **Our contributions:**
  - Implemented DQN for device selection in FL
  - Implemented PCA for state space compression
  - Proposed new reward function based on accuracy improvements
  - Analyzed why DQN does not work for device selection in FL

- **Our findings:**
  - Our DQN implementation cannot reproduce the work shown by Wang et al.
  - Reward function in original paper is problematic
  - DQN is not suitable for problems which has strong dependency between actions

- **Future work:**
  - Select multiple clients rather than one during DQN training, which will possibly improve performance but increase cost exponentially
  - Try policy gradient models such as actor-critic