Dependable AI (CSL7370)
Federated Learning

TEAM DESCRIPTION

Rishav Aich (B21AI029)
Pre-final Year (B.Tech. Artificial Intelligence & Data Science)

Tanish Pagaria (B21AI040)
Pre-final Year (B.Tech. Artificial Intelligence & Data Science)

IIT Jodhpur Undergraduates

OVERVIEW

This project aims to simulate a scenario to compare the performance of the federated learning algorithms on a heterogeneous dataset by implementing FedAvg and two other federated learning algorithms.

DATASET: MNIST

The MNIST dataset is a collection of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9, used for the task of classifying a given image of a handwritten digit into one of 10 classes representing integer values from 0 to 9.
Heterogeneous dataset was created using the MNIST dataset by **splitting** the dataset using a **non-iid sampling**, assigning different digits to different clients, instead of randomly distributing the data across clients.

Client	Client-1	Client-2	Client-3	Client-4	Client-5
Digits	1, 3	0, 6	2, 5	4, 7	8, 9

Client-1 Data

Client-2 Data

Client-3 Data

Client-4 Data

Client-5 Data

CENTRAL SERVER MODEL ARCHITECTURE

Convolutional Neural Network architecture was employed for the multi-class image classification (10 classes). It had two main parts: the feature extraction layers and the classifier. The feature extraction layers used convolutional and max pooling operations along with ReLU activation to capture important patterns from the input images. These patterns were then processed through two fully connected layers with ReLU activation in the classifier section. This helped convert the extracted features into a format suitable for multi-class classification, ultimately providing probabilities for each of the ten possible classes.

FEDERATED LEARNING ALGORITHMS

Federated learning is a decentralized approach to training AI models where data remains on the device, allowing for privacy-preserving machine learning. It involves multiple devices training a shared model without sharing raw data, improving model accuracy by aggregating updates from each device.
The following federated learning algorithms were implemented:

- Federated Stochastic Gradient Descent (FedSGD)
- Federated Averaging (FedAvg)
- Federated Proximal Optimization (FedProx)

FEDERATED STOCHASTIC GRADIENT DESCENT (FedSGD)

It is a foundational algorithm in federated learning that leverages the principles of Stochastic Gradient Descent (SGD) for decentralized model training.
The central model is distributed to clients, each of which computes gradients using local data. These gradients are then aggregated at the central server to update the global model. It is computationally efficient but requires a large number of training rounds to achieve good model performance. It serves as a baseline for federated learning.

Algorithm

$\boldsymbol{w_t}$	Model weights on communication round #t
$\boldsymbol{w}$	Model weights on communication round #t on client k
$\boldsymbol{C}$	Fraction of clients performing computations in each round
$\boldsymbol{E}$	Number of training passes each client makes over its local dataset on each round
$\boldsymbol{B}$	The local minibatch size used for the client updates
$\boldsymbol{\eta}$	The local minibatch size used for the client updates
$\boldsymbol{P_k}$	Set of data points on client k
$\boldsymbol{n_k}$	Number of data points on client k
$\boldsymbol{f_i(w)}$	Loss $l(x_i, y_i; w)$ i.e., loss on example (x_i, Y_i) with model parameters w

For FedSGD, the parameter C which controls the global batch size is set to 1. This corresponds to a full-batch (non-stochastic) gradient descent.
For the current global model w_t, the average gradient on its global model is calculated for each client k.

$$F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w) \quad \Big| \quad g_k = \nabla F_k(w_t)$$

The central server then aggregates these gradients and applies the update.

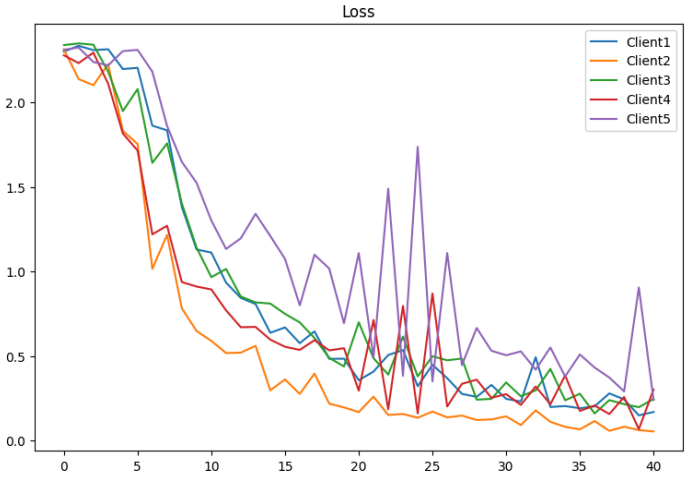
$$w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$$

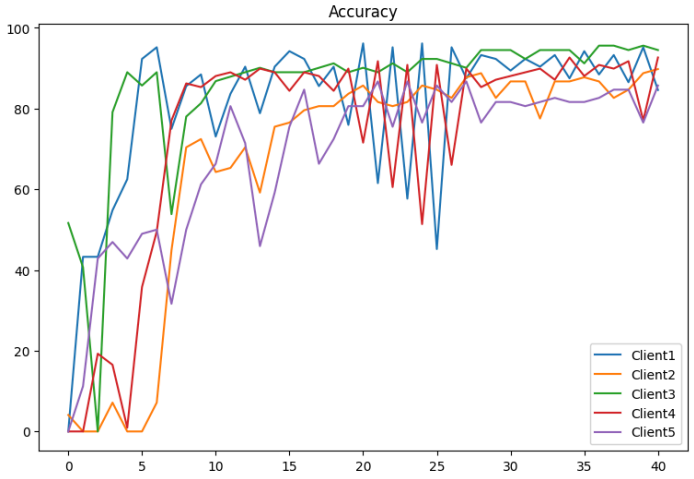
Python Implementation

- The global model was initialized and sent to the clients from the server.
- At each client:
    - Stochastic gradient descent (SGD) updates were applied to the received model using the local dataset.
    - The local training was carried out for a fixed number of epochs.
- The updated client models were sent back to the server.

- On the server:
    - The client models were aggregated, typically by taking a weighted average.
    - The aggregated model became the new global model.

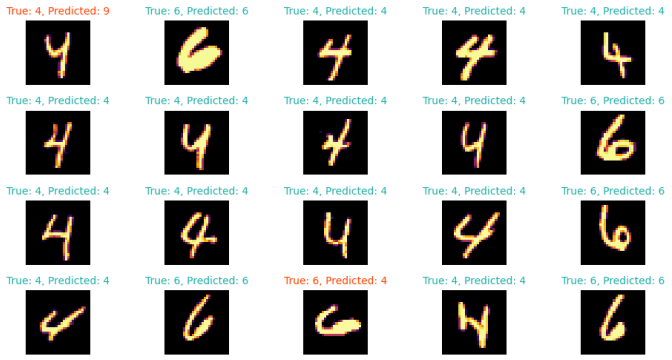
Training



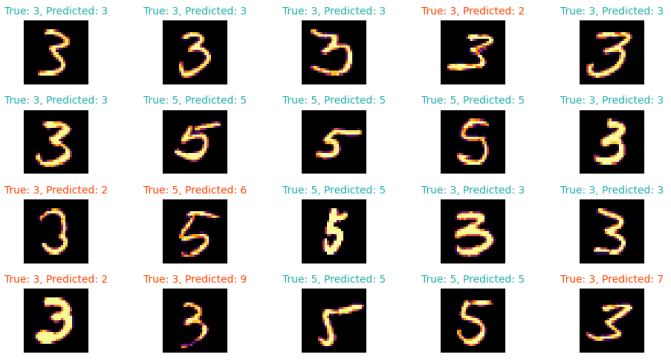


Predictions

Server Testing Accuracy (after 40 epochs)	86.19233491469917

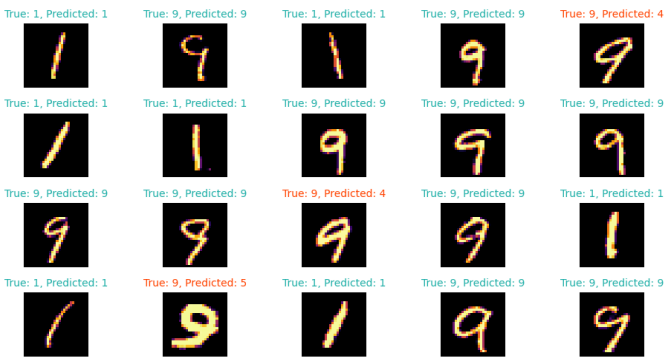


Client-1

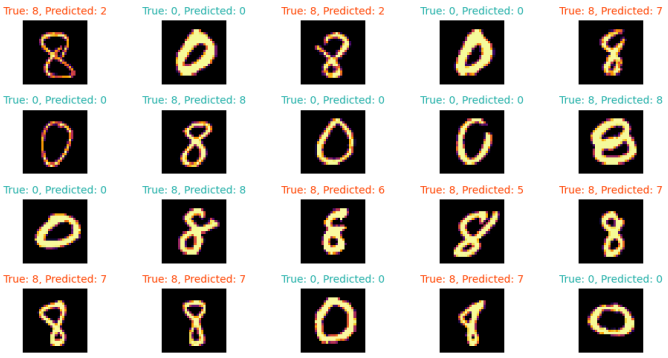


Client-2

Client-3



Client-4



Client-5

FEDERATED AVERAGING (FedAvg)

It involves aggregating the model updates from multiple devices to improve the global model. Each device trains a model on its local data, and the model updates are then averaged to update the global model. This process is repeated iteratively until the global model converges to a satisfactory performance level.

Algorithm

A small change is made to the update step in FedSGD.

$$\forall k, w_{t+1}^k \leftarrow w_t - \eta g_k \quad \Big| \quad w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$$

Each client locally takes one step of gradient descent on the current model using its local data, and the server then takes a weighted average of the resulting models. This way more computation can be added to each client by iterating the local update multiple times before doing the averaging step.

The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

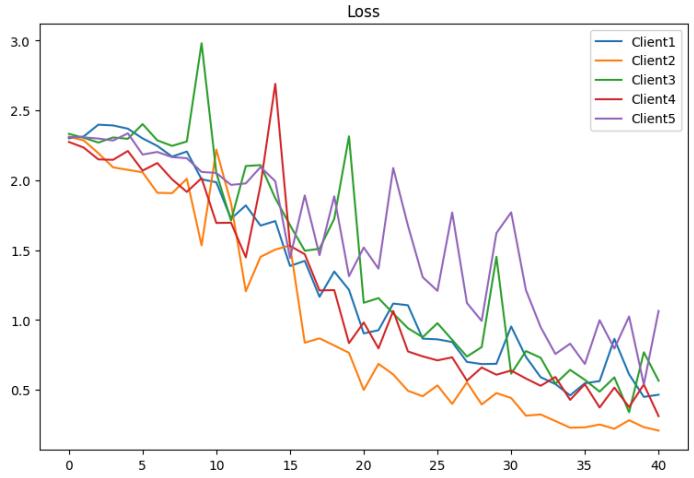
Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

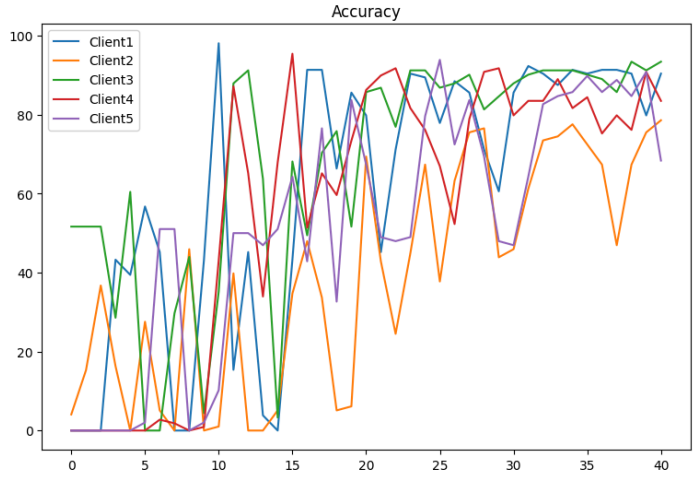
ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

Python Implementation

- Inherited from FedSGD.
- At each client:
  - Multiple epochs of local training were performed using an optimizer (e.g., SGD) on the local dataset.
  - The locally trained client models were sent to the server.
- On the server:
  - The client models were aggregated by taking a weighted average.
  - The aggregated model became the new global model.

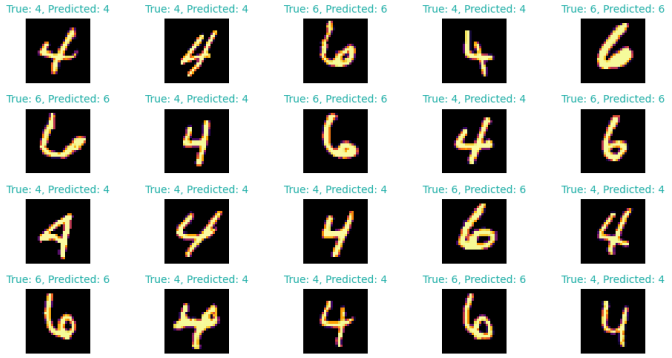
Training





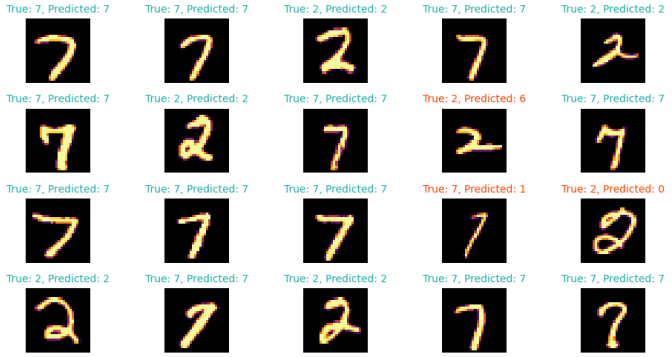
Predictions

Server Testing Accuracy (after 40 epochs)	85.51551888738013

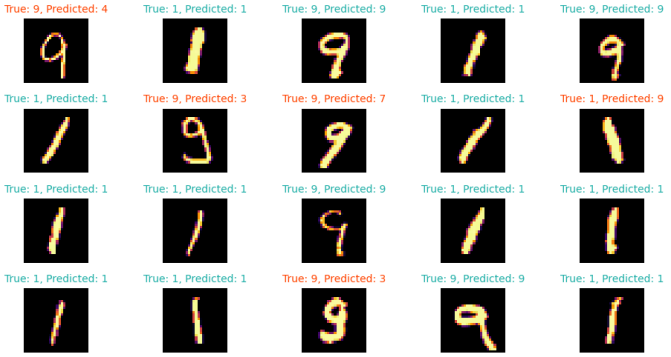


Client-1

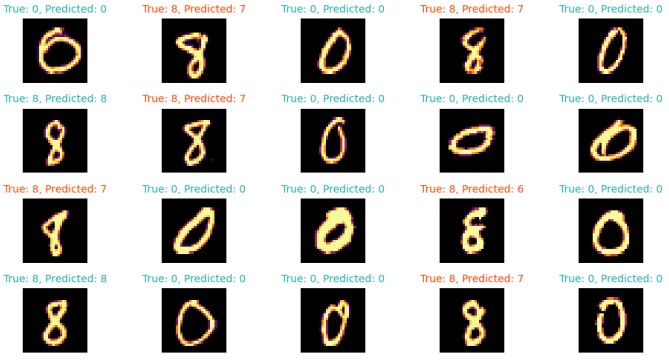
Client-2



Client-3



Client-4



Client-5

FEDERATED PROXIMAL OPTIMIZATION (FedProx)

It is an advanced federated learning algorithm designed to address the challenges of heterogeneity in federated networks, including system heterogeneity and non-identically distributed data.
It is a generalization and re-parametrization of FedAvg, with modifications that provide convergence guarantees for learning over data from non-identical distributions and adhere to device-level systems constraints. It demonstrates more robust convergence than FedAvg, especially in highly heterogeneous settings, improving absolute test accuracy significantly,

Algorithm

The loss for each client $F_k(w)$ in the FedAvg algorithm is replaced with the given objective:

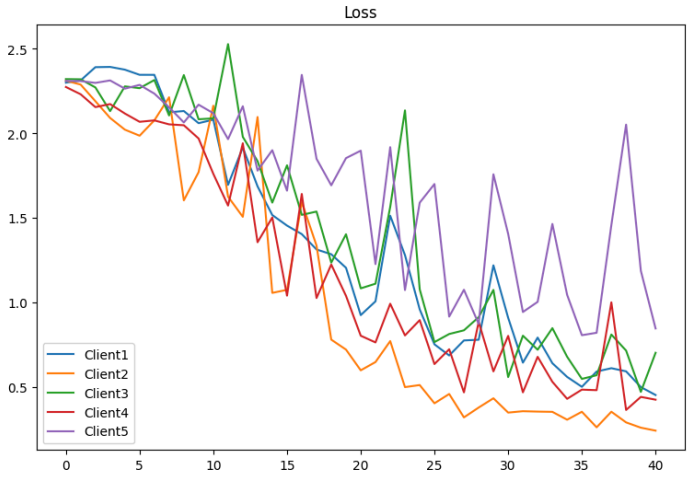
$$\min_w h_k(w;\ w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$$

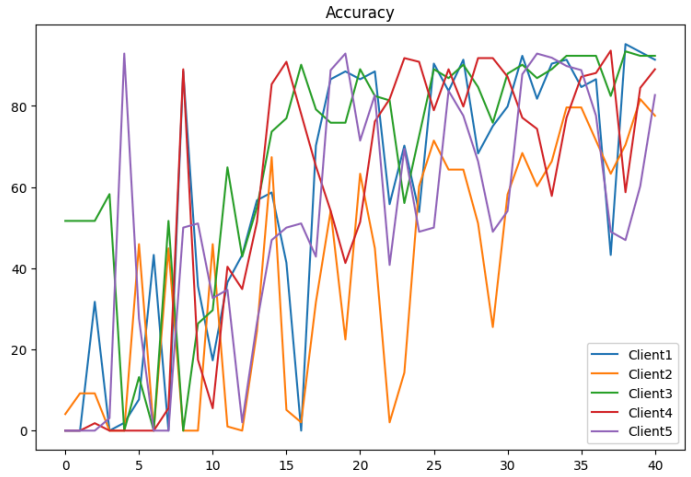
Input: $K, T, \mu, \gamma, w^0, N, p_k, k = 1, \cdots, N$
for $t = 0, \cdots, T-1$ **do**
Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
Server sends w^t to all chosen devices
Each chosen device $k \in S_t$ finds a w_k^{t+1} which is a γ_k^t-inexact minimizer of: $w_k^{t+1} \approx \arg\min_w\ h_k(w;\ w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$
Each device $k \in S_t$ sends w_k^{t+1} back to the server
Server aggregates the w's as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$
end for

Python Implementation

- Inherited from FedAvg.
- At each client:
  - During local training, the objective included the original loss function and a proximal term.
  - The proximal term penalized deviations from the global model, weighted by a hyperparameter ***mu***.
- The locally trained client models were sent to the server.
- On the server:
  - The client models were aggregated by taking a weighted average.
  - The aggregated model became the new global model.

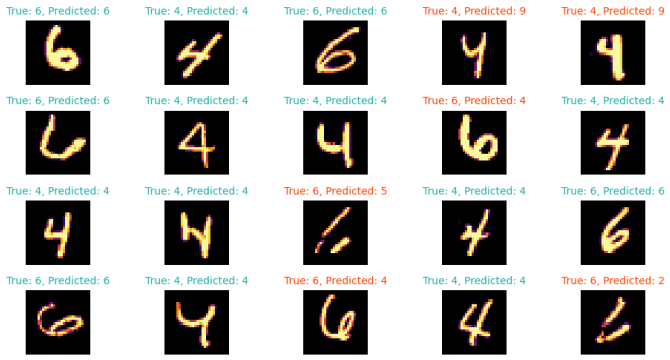
Training (mu = 0.3)



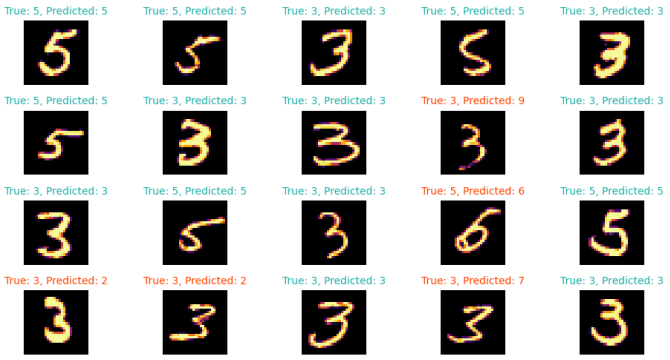


Predictions

Server Testing Accuracy (after 40 epochs)	80.60328881919023

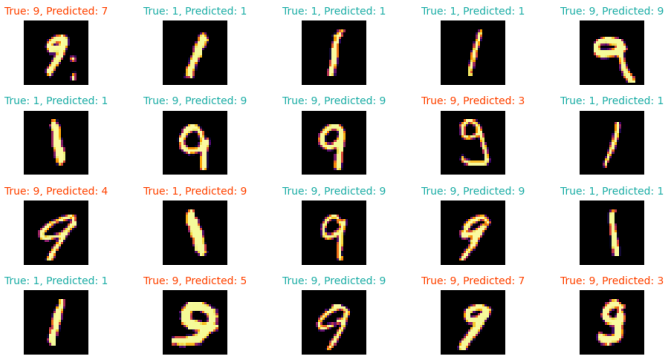


Client-1

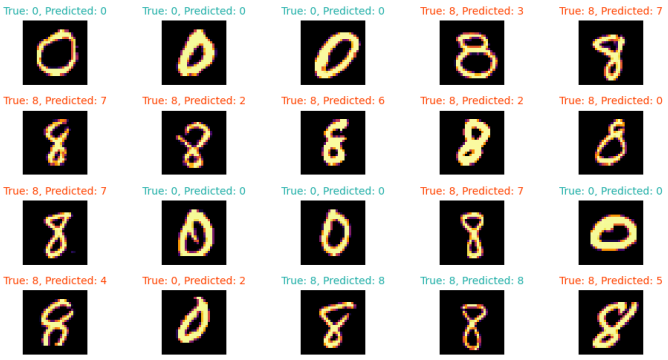


Client-2

Client-3



Client-4



Client-5

OBSERVATIONS

- FedSGD converged faster than FedAvg and FedProx in the initial iterations.
- FedSGD had a lower computational cost per iteration since it performed a single epoch of SGD updates.
- FedAvg and FedProx had a higher computational cost per iteration due to the multiple epochs of local training.
- FedSGD was less robust to non-IID data due to its reliance on direct summation of updates without any normalization or averaging.
- FedAvg was more robust to non-IID data due to the averaging mechanism which smooths out differences in data distributions among clients.
- FedProx was moderately robust to non-IID data due to the additional proximal term, which encourages convergence towards a consensus model.

REFERENCES

FedSGD & FedAvg: Communication-Efficient Learning of Deep Networks from Decentralized Data
https://arxiv.org/pdf/1602.05629.pdf
https://medium.com/@anhtuan_40207/paper-review-communication-efficient-learning-of-deep-networks-from-decentralized-data-f62eefceac42

FedProx: Federated Learning in Heterogeneous Systems
https://arxiv.org/pdf/1812.06127.pdf