COMPUTER SCIENCE AND DATA ANALYTICS

Course: **Guided Research**

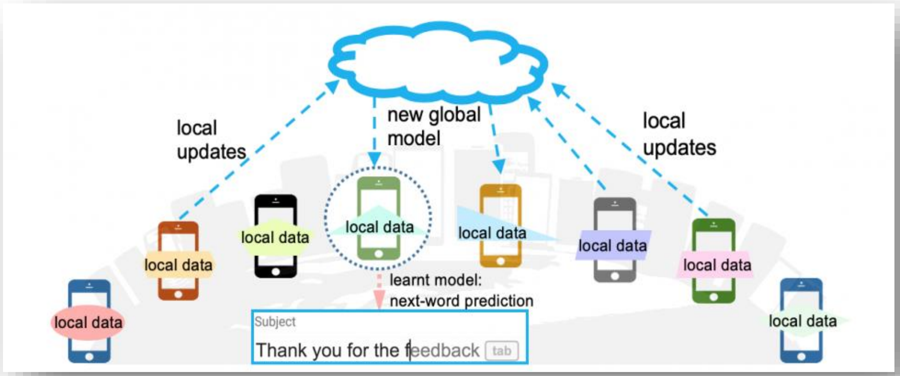
Project Title: **Federated Machine Learning Implementation on Image Classification**

Student: **Ali Asgarov**

Instructors & Supervisors: **Dr. Stephen Kaisler, Dr. Jamal Hasanov**

Date: **08.03.2023**

Project Objective



Why today ?

Standard machine learning approaches necessarily require storing training data on a single machine or in datacenter.

Why can't we just centralize the data all the time ?

What are the limits of current practice ?

Sending the data may be too costly

Self-driving cars generates several TBs of data a day

Wireless devices have limited bandwidth/power

Data may be considered too sensitive

Public awareness and regulations on data privacy

Control of data is advantage in business/research

What's new in our approach ?

Federated Learning (FL) – Keep data decentralized.

Collaborative ML model training on decentralized data.

Each client's raw data is stored locally .

Parameter aggregation.

Control over data.

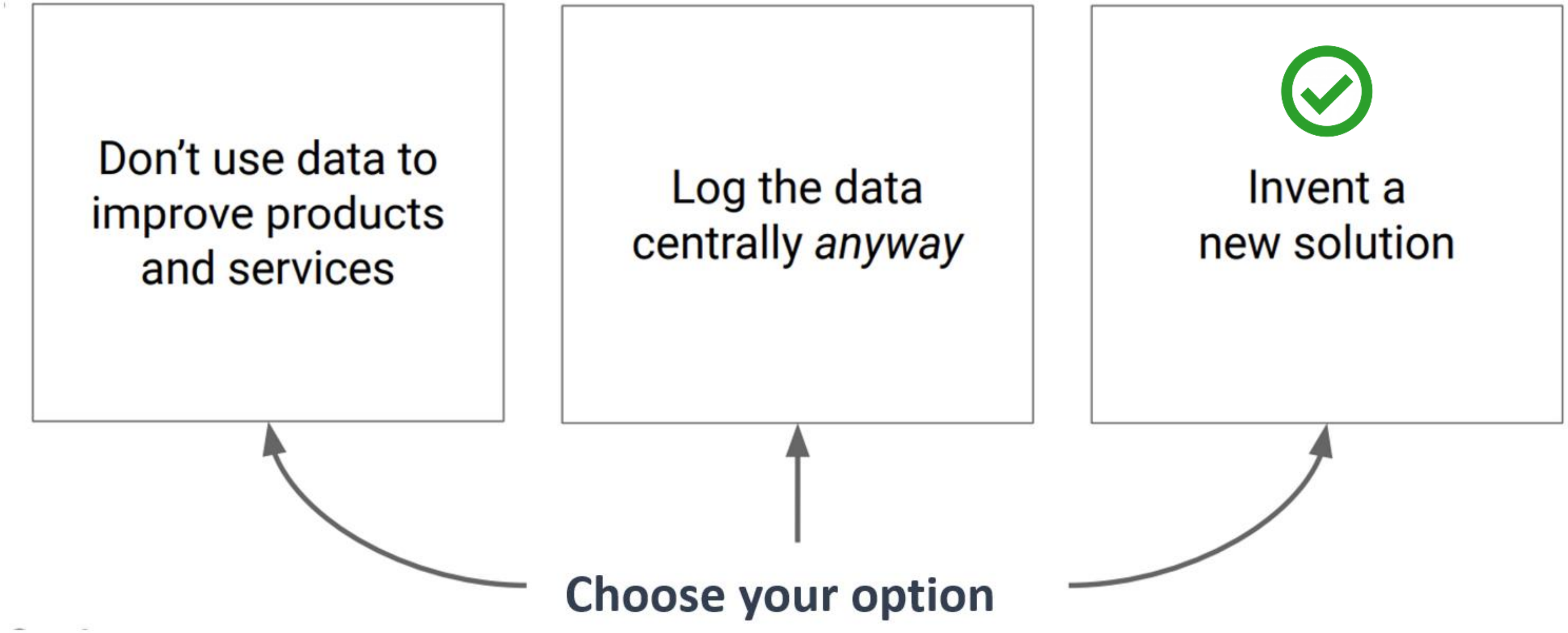
No need to high data transmission bandwidth/power.

Leveraging local data diversity – improving accuracy.

Heilmeier questions noted in green through the whole presentation.

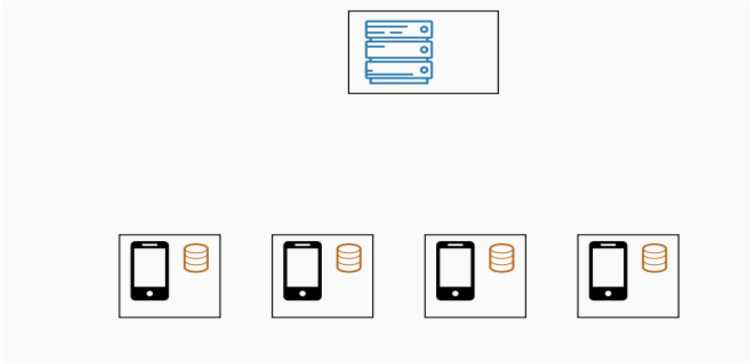
Literature Review

Between 2014 – 2016 Google had three options about the data

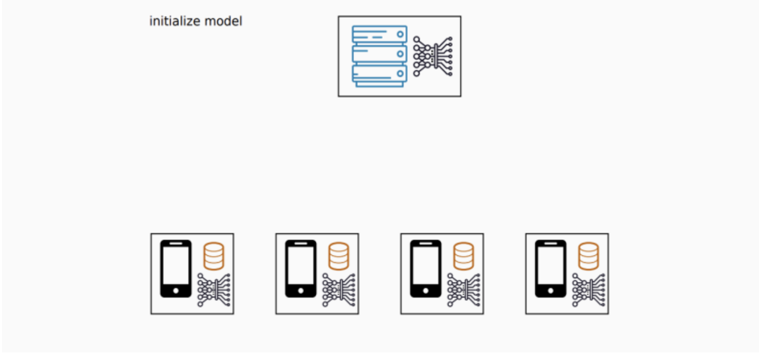


Solution Architecture

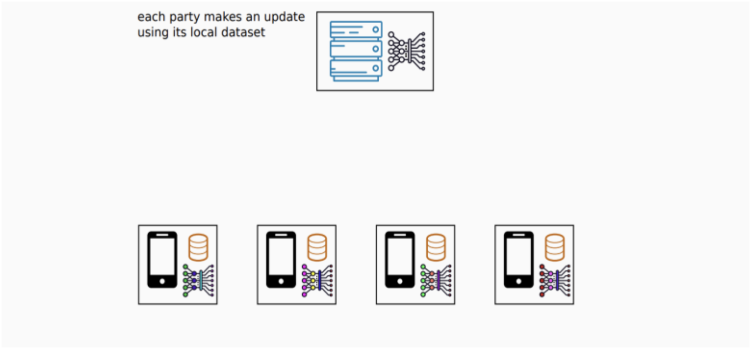
Data preparation (IID)

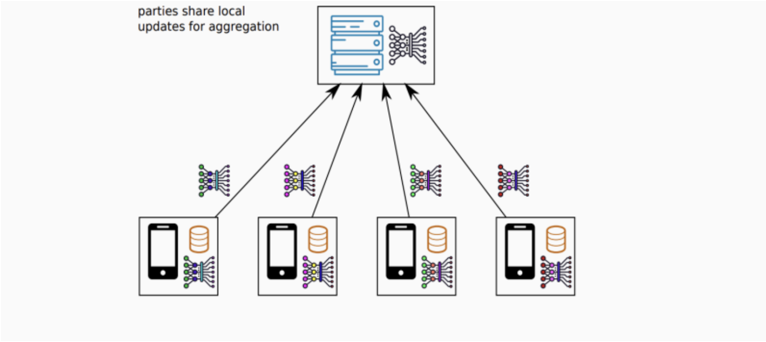


Initialize models

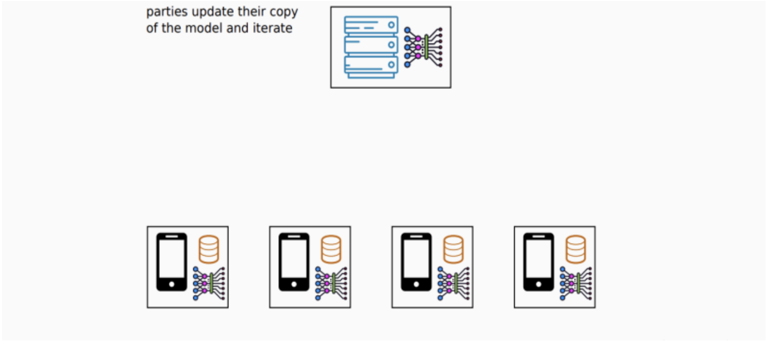


Train local models on the data samples

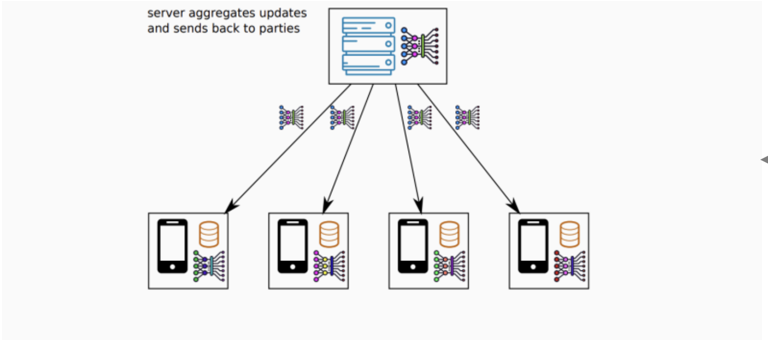




Sharing the model weights to the central model



Local models weights are being averaged



Averaged weights are sent to the local clients

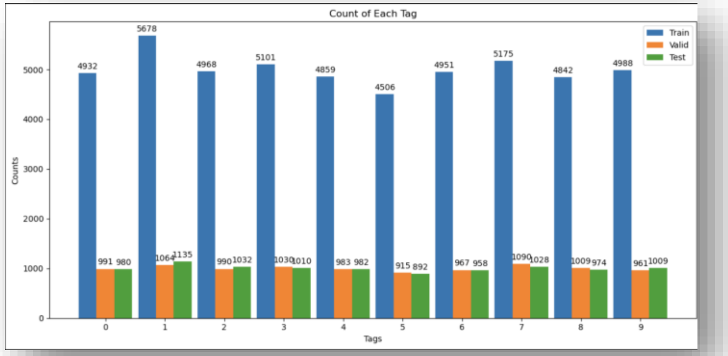
Dataset | Ingestion | Preprocessing

MNIST Dataset - 28 * 28 pixel grayscale images of numbers from 0 to 9.

The MNIST data set does not contain each label equally.

The IID sampling of the training data needed.

To fulfill the IID requirement :



Shuffling data and building dictionary for indexes of labels

Building of client dataset dictionaries with shuffled data

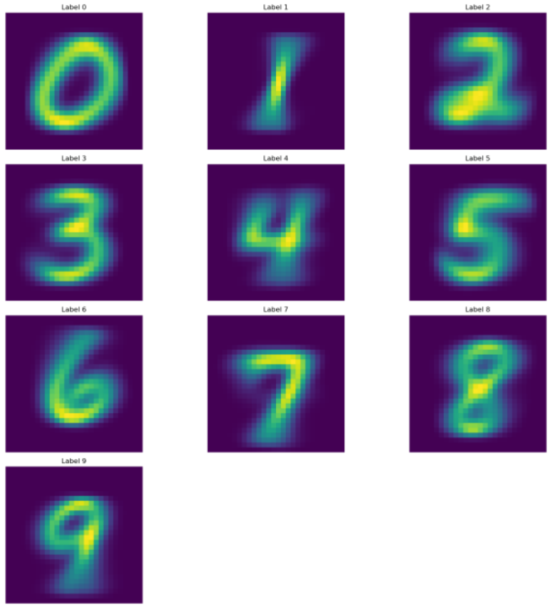
Building of client datasets with previous steps' dictionaries

Dataset Cleaning

Heatmap for each label

Calculated the mean image (2000 random samples) for building heatmap.

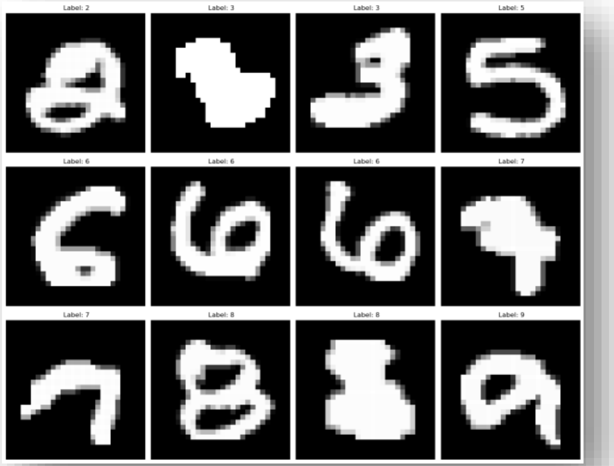
Mean image - array mean values reshaped into a 28x28 matrix.



Outlier detection

Calculating the Euclidean distance between samples and the mean image.

Defining outlier threshold – (100).



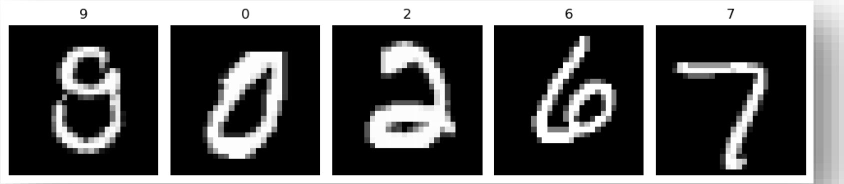
Indices of outlier images:
[41453, 24798, 25315, 36193, 29489, 25317, 8488, 59423, 8586, 18598]

Normalization

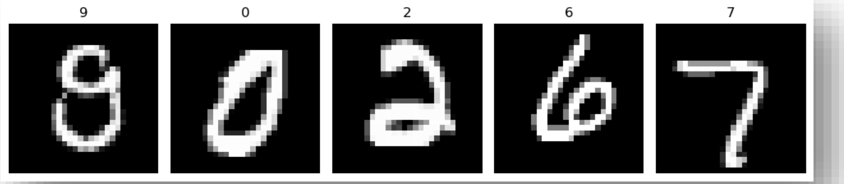
Feature scaling to ensure all features (pixels) are on a similar scale.

Normalized the pixel values of the images to a range between 0 and 1 by dividing to 255.

Before normalization

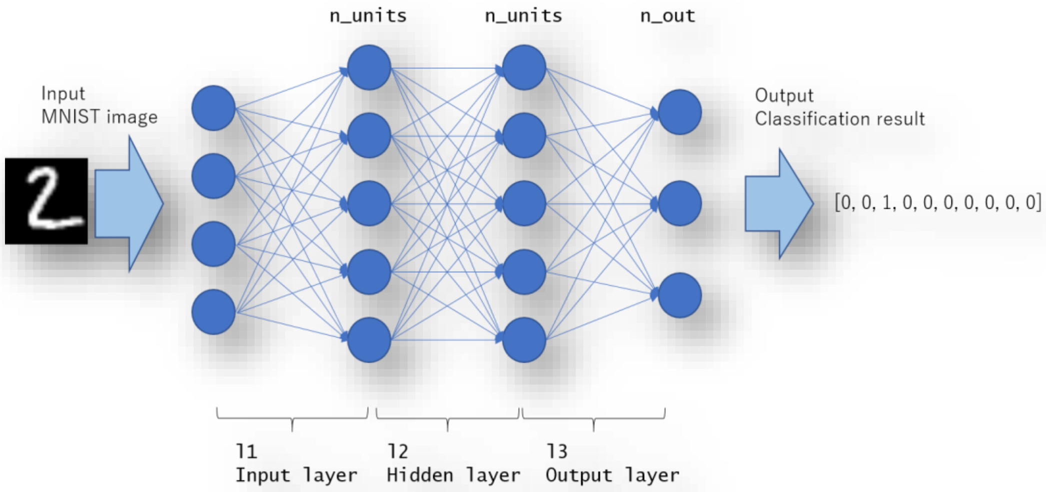


After normalization



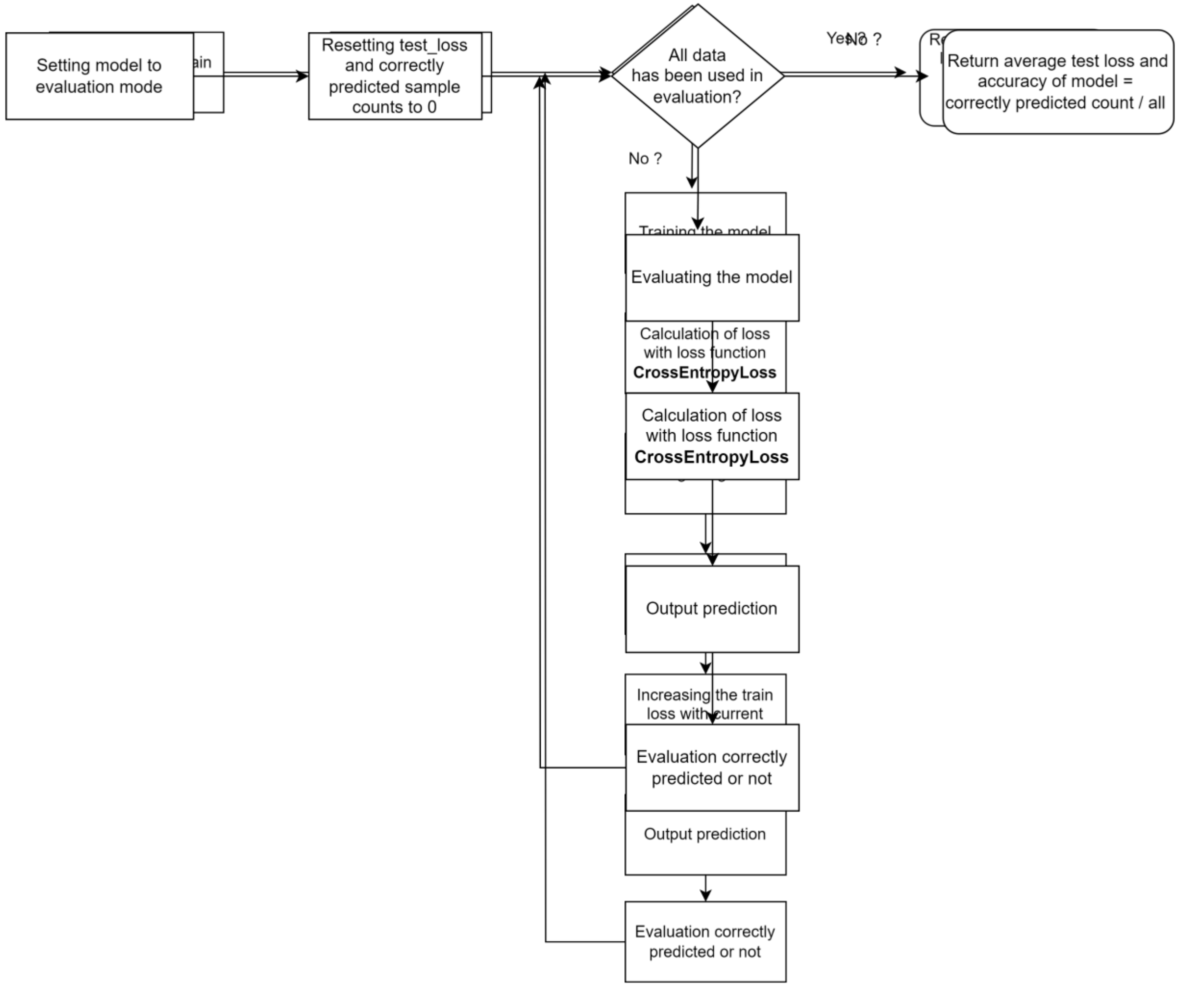
Modelling

A 3-layer model was created for the classification process.




```
net2nn(
  (fc1): Linear(in_features=784, out_features=200, bias=True)
  (fc2): Linear(in_features=200, out_features=200, bias=True)
  (fc3): Linear(in_features=200, out_features=10, bias=True)
)
```

Validation



Federated averaging methods

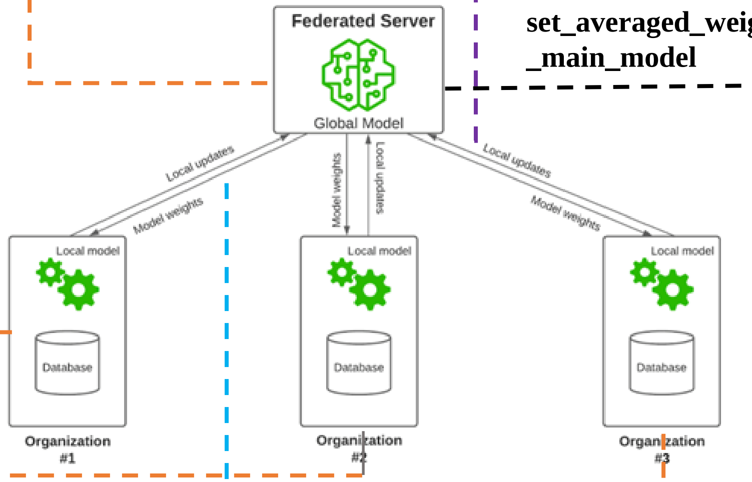
Federated learning functions implemented:

get_averaged_weights

Getting averaged weights from all clients

create_model_optimizer_criterion_dict

Building dictionaries for models & optimizers & loss



set_averaged_weights_as_main_model_weights_and_update_main_model

Setting averaged weights to centralized model

send_main_model_to_nodes_and_update_model_dict

Sending main model parameters (averaged) to local models

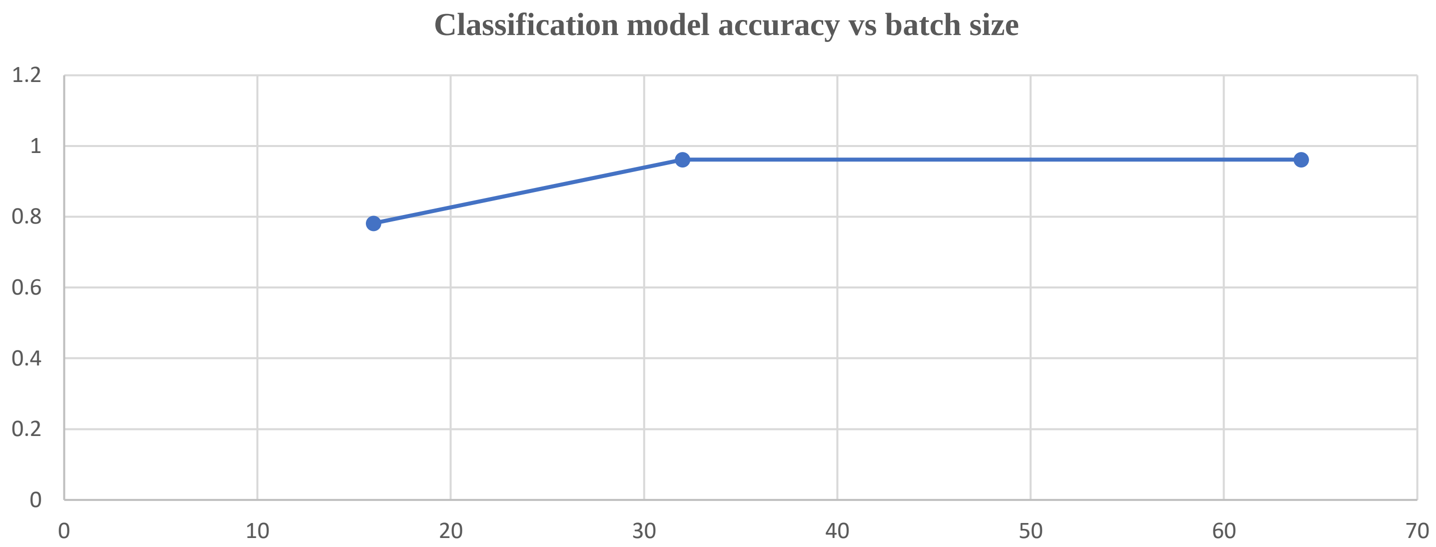
Measurement and Analysis

Dependent variable : Accuracy of the main classification model

Independent variables :

Number of clients
Learning rate
Number of epochs for training
Batch size

Designing experiments: Created a series of experiments where I systematically vary the independent variables while keeping other factors constant



Results

Centralized model with centralized and non-IID data ?

```
----------- Centralized ( Non - Distributed ) Model ------
--------------------- Training Started -------------------
Epoch:   1 | Train accuracy:  0.8893 | Test accuracy:  0.8629
Epoch:   2 | Train accuracy:  0.9617 | Test accuracy:  0.9652
Epoch:   3 | Train accuracy:  0.9741 | Test accuracy:  0.9713
Epoch:   4 | Train accuracy:  0.9797 | Test accuracy:  0.9743
Epoch:   5 | Train accuracy:  0.9855 | Test accuracy:  0.9736
Epoch:   6 | Train accuracy:  0.9885 | Test accuracy:  0.9738
Epoch:   7 | Train accuracy:  0.9909 | Test accuracy:  0.9782
Epoch:   8 | Train accuracy:  0.9935 | Test accuracy:  0.9761
Epoch:   9 | Train accuracy:  0.9945 | Test accuracy:  0.9624
Epoch:  10 | Train accuracy:  0.9965 | Test accuracy:  0.9791
--------------------- Training finished ------------------
```

Train (50000) and test (10000) amounts are full train and test data

learning_rate is 0.2

momentum is 0.2

numEpoch is 30

Chosen parameters from the measurement and analysis

number_of_clients is 100
learning_rate is 0.2
numEpoch is 30
batch_size is 64
momentum is 0.2
train_amount is 4000 for each label (build IID data)
test_amount is 1000 for each label (build IID data)

```
Iteration 2 : main_model accuracy on all test data:  0.8915
Iteration 3 : main_model accuracy on all test data:  0.9134
Iteration 4 : main_model accuracy on all test data:  0.9243
Iteration 5 : main_model accuracy on all test data:  0.9319
Iteration 6 : main_model accuracy on all test data:  0.9394
Iteration 7 : main_model accuracy on all test data:  0.9438
Iteration 8 : main_model accuracy on all test data:  0.9427
Iteration 9 : main_model accuracy on all test data:  0.9490
Iteration 10 : main_model accuracy on all test data:  0.9504
Iteration 11 : main_model accuracy on all test data:  0.9526
Iteration 12 : main_model accuracy on all test data:  0.9542
Iteration 13 : main_model accuracy on all test data:  0.9565
Iteration 14 : main_model accuracy on all test data:  0.9577
Iteration 15 : main_model accuracy on all test data:  0.9601
```

15 iterations have been done for the convergence.

Conclusion and future scope

What have been achieved ?

Centralized model (not FL) achieved an accuracy of 97.9%.

FedAvg averaging process reaching an impressive 96.01% accuracy without seeing the data.

Data Privacy

Decentralized Training

Lower Bandwidth and Power Usage

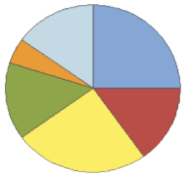
Control Over Data

Scalability

Cost-Efficient

What were the risks and what are the next steps ?

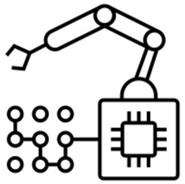
Handling Heterogeneity



FedAvg is dependent on the IID data

- Working on the non-IID data.

Adaptive Learning Rates



Fixed learning rates for all devices

- Client specific learning rate.

Fault Tolerance and Dynamic Client Selection

Faulty or malicious devices - inaccurate updates.

Detection of those clients needed in this scenario.

- Selecting devices to participate in training.

References

- McMahan, H. B. (2016, February 17). *Communication-Efficient Learning of Deep Networks from Decentralized Data*. arXiv.org. https://arxiv.org/abs/1602.05629
- S. Xing, Z. Ning, J. Zhou, X. Liao, J. Xu and W. Zou, "N-FedAvg: Novel Federated Average Algorithm Based on FedAvg," 2022 14th International Conference on Communication Software and Networks (ICCSN), Chongqing, China, 2022, pp. 187-196, doi: 10.1109/ICCSN55126.2022.9817607.
- *H. B. Mcmahan and D. Ramage, "Communication-Efficient Learning of Deep Networks from Decentralized Data," vol. 54, 2017.*
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278–2324, November 1998.
- Liang, Paul Pu et al. "Think Locally, Act Globally: Federated Learning with Local and Global Representations." *ArXiv* abs/2001.01523 (2020): n. pag.
- T. Sun, D. Li and B. Wang, "Decentralized Federated Averaging," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 4, pp. 4289-4301, 1 April 2023, doi: 10.1109/TPAMI.2022.3196503.
- Speedup, Linear, Zhaonan Qu, Kaixiang Lin and Zhaojian Li. "FEDERATED LEARNING'S BLESSING: FEDAVG." (2020).

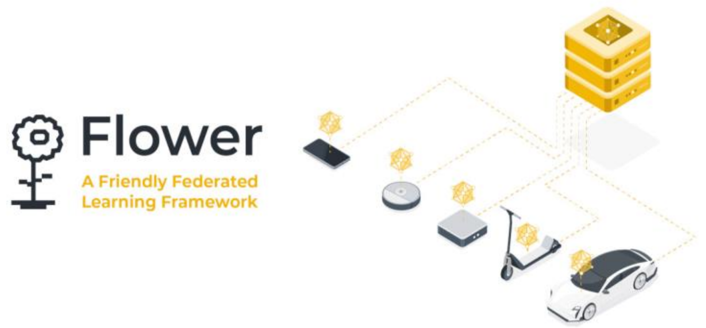
Thank You

Backup

FL Frameworks Under Development

Several open-source libraries are under development: PySyft, TensorFlow Federated, FATE, Flower, Substra..

Private Machine learning as a Service using PySyft



Applications - I

ARTIFICIAL INTELLIGENCE, DIAGNOSTICS

UPenn, Intel partner to use federated learning AI for early brain tumor detection

The project will bring in 29 institutions from North America, Europe and India and will use privacy-preserved data to train AI models. Federated learning has been described as being born at the intersection of AI, blockchain, edge computing and the Internet of Things.

By ALARIC DEARMENT

Post a comment / May 11, 2020 at 10:03 AM

"The University of Pennsylvania and chipmaker Intel are forming a partnership to enable 29 heatlhcare and medical research institutions around the world to train artificial intelligence models to detect brain tumors early."

"The program will rely on a technique known as federated learning, which enables institutions to collaborate on deep learning projects without sharing patient data. The partnership will bring in institutions in the U.S., Canada, U.K., Germany, Switzerland and India. The centers – which include Washington University of St. Louis; Queen's University in Kingston, Ontario; University of Munich; Tata Memorial Hospital in Mumbai and others – will use Intel's federated learning hardware and software."

ALL ABOUT CIRCUITS

NEWS

Is Machine Learning for Tumor Research at Odds With Patient Privacy? Not With Federated Learning, Intel Says

May 13, 2020 by Tyler Charboneau

VB The Machine

Intel partners with Penn Medicine to develop brain tumor classifier with federated learning

Bio-IT World

Next-Gen Technology • Big Data • Personalized Medicine

RELATED STORIES

No Cytokine Storm, 'Bursty' Disease Spread: Biomarkers For Elevated Risk: COVID-19 Updates | Sep 10, 2020

Beyond the Rule of 5: Vastly Expanding Targetable Drugs | Sep 08, 2020

Computational Tool Could Improve Clinical Success Rate Of Drugs | Sep 08, 2020

Mouse Models, Machine Learning, Triggering Proteins: COVID-19 Updates | Sep 03, 2020

Truth Challenge v2: Latest Challenge Results From Genome In A Bottle | Sep

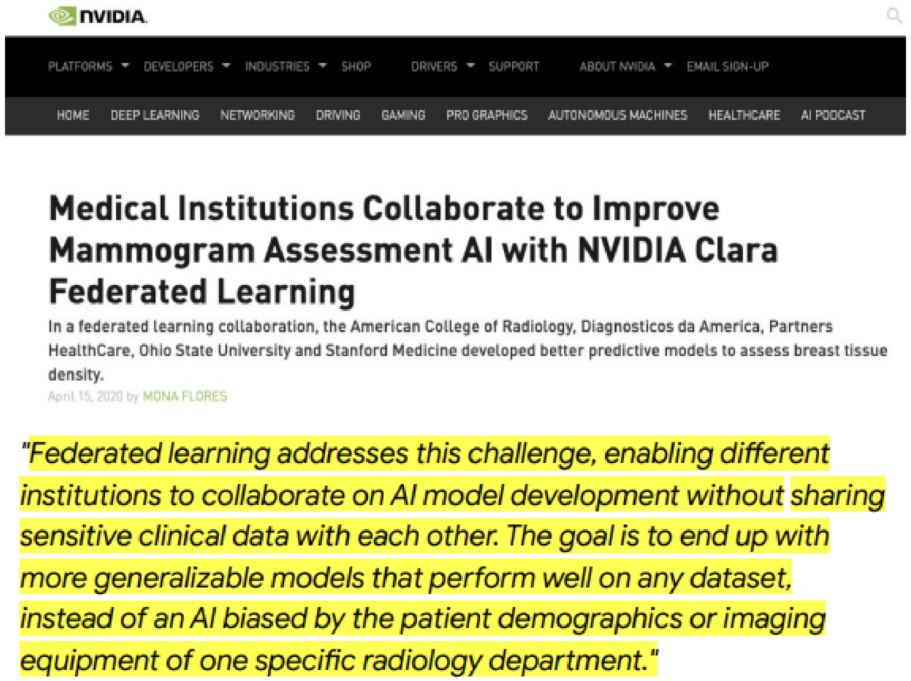
Intel, Penn Medicine Launch Federated Le

By Allison Proffitt

May 28, 2020 | The University of Pennsylvania and Intel have built a federation of 30 institutions to use federated learning to train artificial intelligence (AI) models to identify boundaries of brain tumors.

Led by Spyridon Bakas at the Center for Biomedical Image Computing and Analytics (CBICA) at the Perelman School of Medicine at the University of Pennsylvania, the federation is the next step forward in a years-long effort to gather data that would empower AI in brain image analysis.

Applications - II



PLATFORMS DEVELOPERS INDUSTRIES SHOP DRIVERS SUPPORT ABOUT NVIDIA EMAIL SIGN-UP

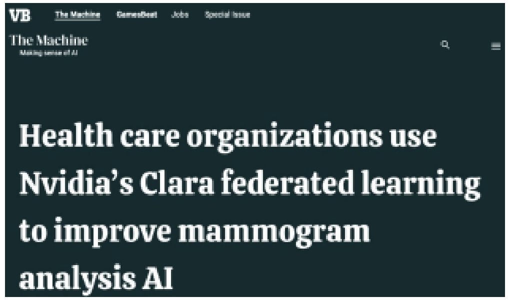
HOME DEEP LEARNING NETWORKING DRIVING GAMING PRO GRAPHICS AUTONOMOUS MACHINES HEALTHCARE AI PODCAST

Medical Institutions Collaborate to Improve Mammogram Assessment AI with NVIDIA Clara Federated Learning

In a federated learning collaboration, the American College of Radiology, Diagnosticos da America, Partners HealthCare, Ohio State University and Stanford Medicine developed better predictive models to assess breast tissue density.

April 15, 2020 by MONA FLORES

"Federated learning addresses this challenge, enabling different institutions to collaborate on AI model development without sharing sensitive clinical data with each other. The goal is to end up with more generalizable models that perform well on any dataset, instead of an AI biased by the patient demographics or imaging equipment of one specific radiology department."



The Machine

Health care organizations use Nvidia's Clara federated learning to improve mammogram analysis AI

MENU SEARCH MedCityNews

HOSPITALS, ARTIFICIAL INTELLIGENCE, HEALTH TECH

Nvidia says it has a solution for healthcare's data problems

The chipmaker touted a new framework that would allow hospitals and pharmaceutical companies to collaborate on AI projects without sharing sensitive data. Nvidia said the framework is already gaining ... ls and drug developers.

The Machine

Nvidia and Mercedes-Benz detail self-driving system with automated routing and parking

Weights Updating

```
Before updating :
Main model weight :  tensor([[-0.0301, -0.0302, -0.0298,  0.0291,  0.0280]],
       grad_fn=<SliceBackward0>)
Client model weight :  tensor([[-0.0015,  0.0124, -0.0091,  0.0177,  0.0314]],
       grad_fn=<SliceBackward0>)
```

```
After updating :
Main model weight :  tensor([[ 0.0585,  0.0462, -0.0580, -0.0622,  0.0498]],
       grad_fn=<SliceBackward0>)
Client model weight :  tensor([[ 0.0585,  0.0462, -0.0580, -0.0622,  0.0498]],
       grad_fn=<SliceBackward0>)
```

SGD

Stochastic Gradient Descent (SGD) is an optimization algorithm used to train machine learning models.

It updates model parameters based on small random subsets of data, called batches, to minimize the difference between predicted and actual outputs (loss)

FL relies on SGD due to its ability to train models on decentralized data while preserving privacy.

SGD only requires sending small gradients, making it suitable for FL scenarios with limited bandwidth

SGD allows each device to update its local model using only its own data without requiring information from other devices or a central server.

weight = weight - lr * gradient - momentum * previous_update

Why not other optimization algos ?

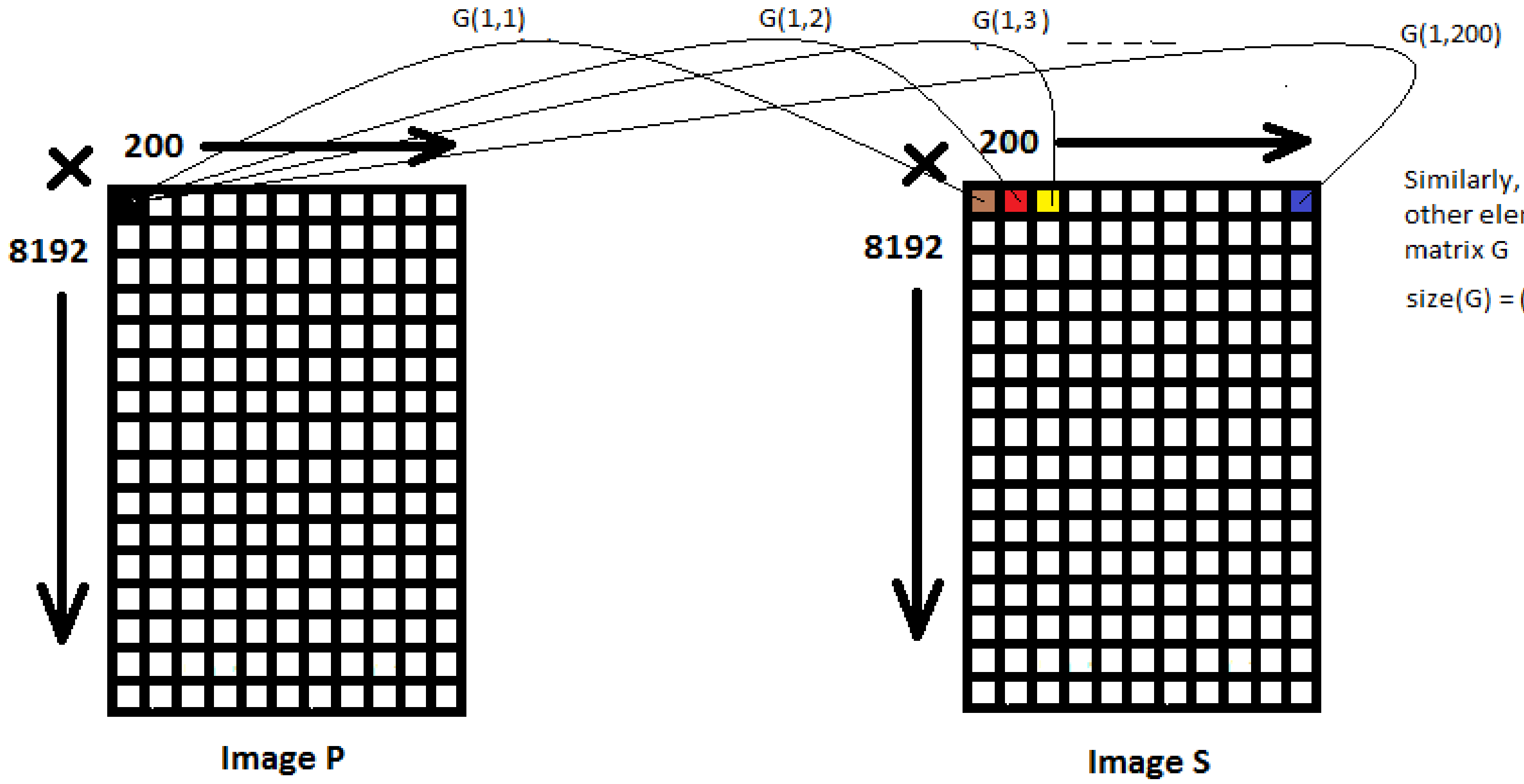
- **Batch Gradient Descent**: Requires the entire dataset to compute gradients, it would involve transmitting large amounts of data, compromising privacy and communication efficiency.
- **Mini-batch Gradient Descent**: Requires sharing data between devices, leading to privacy concerns and communication challenges.
- **Genetic Algorithms:** Involve a population-based approach with evolving solutions. This process might require sharing sensitive information and lacks the communication efficiency needed in FL.
- **Newton's Method:** Requires calculating the Hessian matrix, which is computationally expensive and not practical for the distributed and resource-constrained nature of FL.

FedAvg

$$F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w)$$

$$g_k = \nabla F_k(w_t)$$

Euclidean distance in images



Similarly, Calculate other elements of matrix G

size(G) = (8192 X200) X (8192 X200)