Federated Learning

Devin Long, Raul Dozal, Luis Postigo

COMP 5130 - Data Mining

11/04/2025

AUBURN
ENGINEERING

Agenda

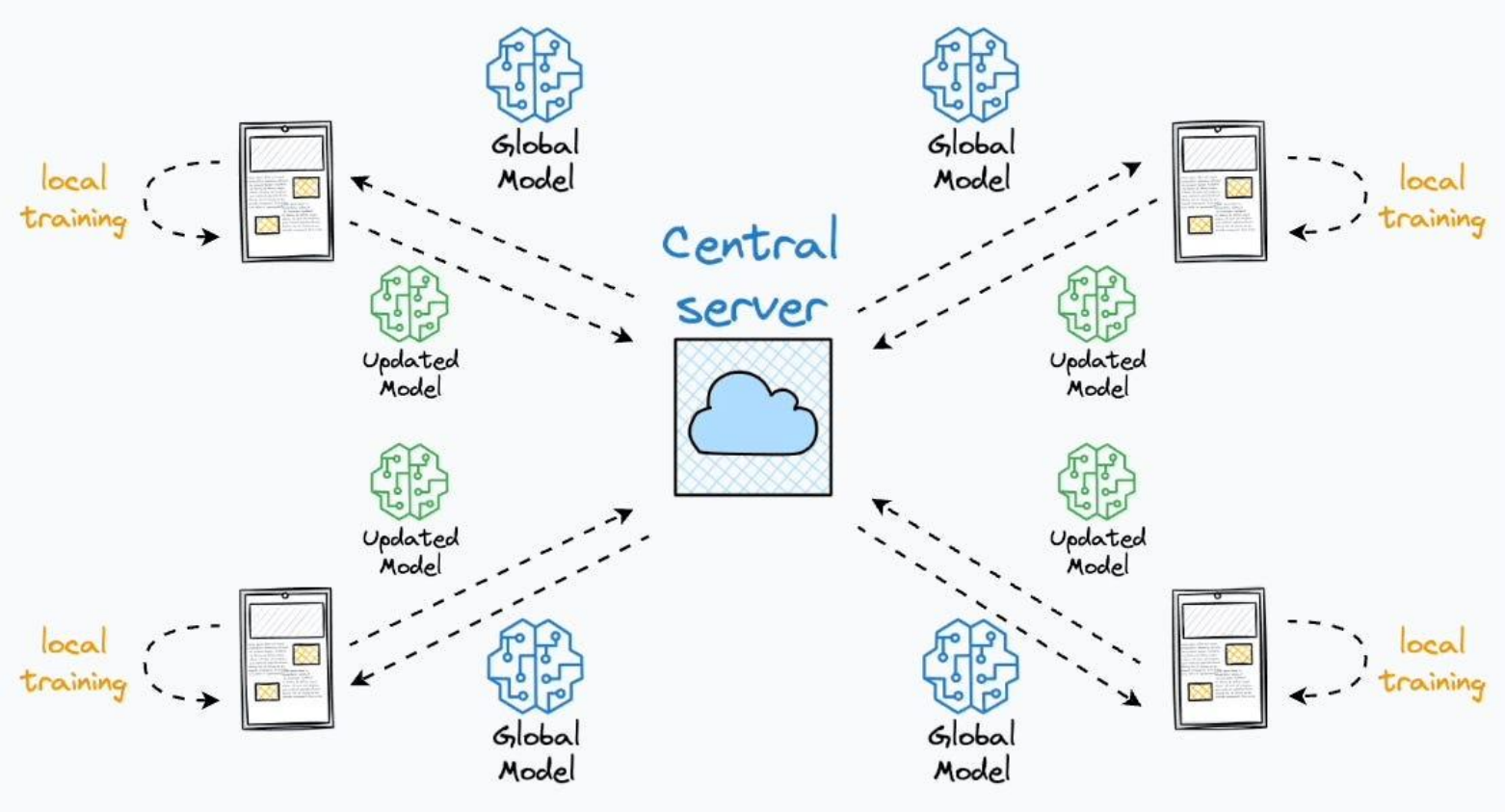
1. Why Federated Learning (FL)?
2. Data & preprocessing (IID vs non-IID)
3. Training scheme & hyperparameters
4. Algorithms & model design (FedSGD, FedAvg, SGD)
5. Evaluation protocol, results & analysis
6. Code, demo, and environment

1. Why Federated Learning?

Federated Learning [COMP 5130 - Intro to OS]

AUBURN UNIVERSITY
Samuel Ginn College of Engineering

Why Federated Learning?



- Train useful on-device models without centralizing raw data; clients keep data local and only send updates.
- Privacy & risk reduction via “focused collection / data minimization”; limits attack surface to device + server coordination.
- Contribution of FedAvg: robust to unbalanced, non-IID data; orders-of-magnitude fewer rounds vs naive approaches.

INTRODUCTION - Federated Learning

Federated Average (FEDAVG)

- Algorithm to train ML models
  - Returns model updated weights after training each mini-batch for multiple epochs (SGD updates) at the client-side
- Privacy-preserving design
- Improves user experience on mobile devices

Comparative Algorithms

- Stochastic Gradient Descent (SGD)
  - Returns raw data to the central server after each mini-batch to compute the gradients
- Federated SGD (FEDSGD)
  - Returns only gradient updates after every local batch at the client-side

INTRODUCTION - Federated Learning

Communication-Efficient Learning of Deep Networks from Decentralized Data

H. Brendan McMahan **Eider Moore** **Daniel Ramage** **Seth Hampson** **Blaise Agüera y Arcas**
Google, Inc., 651 N 34th St., Seattle, WA 98103 USA

Abstract

Modern mobile devices have access to a wealth of data suitable for learning models, which in turn can greatly improve the user experience on the device. For example, language models can improve speech recognition and text entry, and image models can automatically select good photos. However, this rich data is often privacy sensitive, large in quantity, or both, which may preclude logging to the data center and training there using conventional approaches. We advocate an alter-

promise of greatly improving usability by powering more intelligent applications, but the sensitive nature of the data means there are risks and responsibilities to storing it in a centralized location.

We investigate a learning technique that allows users to collectively reap the benefits of shared models trained from this rich data, without the need to centrally store it. We term our approach *Federated Learning*, since the learning task is solved by a loose federation of participating devices (which we refer to as *clients*) which are coordinated by a central *server*. Each client has a local training dataset which is never uploaded to the server. Instead, each client computes

H. Brendan McMahan
Principal Research Scientist at Google Research.

Best known as the pioneer of Federated Learning and lead author of this publication.

2. Data & preprocessing

Federated Learning [COMP 5130 - Intro to OS]

AUBURN UNIVERSITY
Samuel Ginn College of Engineering

DATASET - Modified National Institute of Standards and Technology (MNIST)

MNIST dataset

- 60,000 training images
- 10,000 testing images
- Each image is 28 x 28 pixels
- Each pixel is on a grayscale

Image source: Ultralytics, “MNIST – Sample Images & Annotations”, https://docs.ultralytics.com/datasets/classify/mnist/#sample-images-and-annotations

HOW TO OBTAIN AND PREPROCESS THE DATA

Step 1: List globals

- MNIST datasets:
  - 60,000 training samples
  - Obtained from python torchvision.datasets.MNIST
- num_clients: int = 5
- local_epochs: int = 2
- local_batch_size: int = 64
- rounds: int = 25

Step 2: Dividing dataset

- Images per client:
  - (Total training samples)/(num_clients)
  - 60,000/5
  - 12,000 images/client
- For IID:
  - Randomly dividing training samples evenly among clients (12,000 images each)
- For Non-IID:
  - Sort training samples by class and divide contiguously and evenly among clients (12,000 images each)

What happens per Communication Round (x25)?

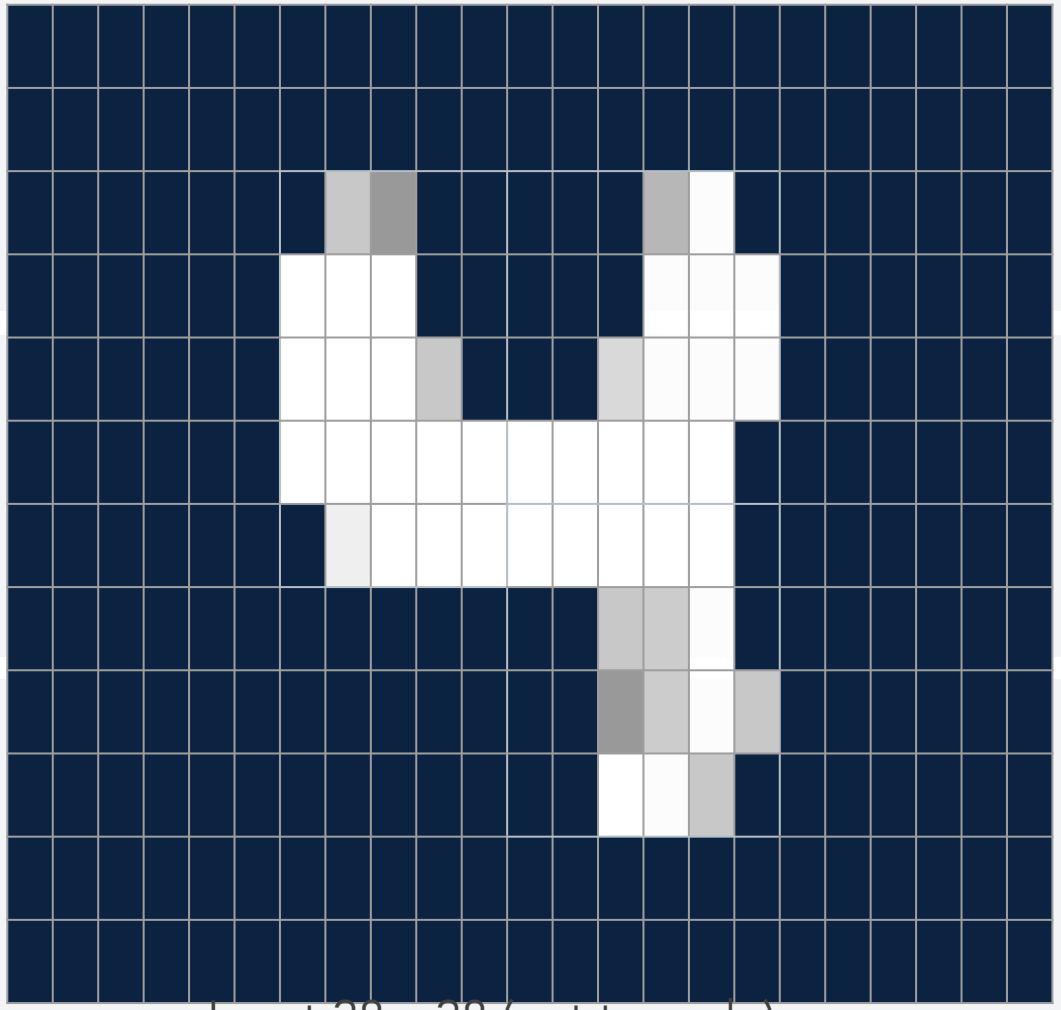
- Batches per epoch:
  - 12,000/64
  - 187.5 or 188 batches
- Total updates:
  - batches * local_epoch
  - 188 * 2
  - 376 local SGD updates

3. Training scheme & hyperparameters

Federated Learning [COMP 5130 - Intro to OS]

AUBURN UNIVERSITY
Samuel Ginn College of Engineering

TRAINING SCHEME - Image Processing Problem

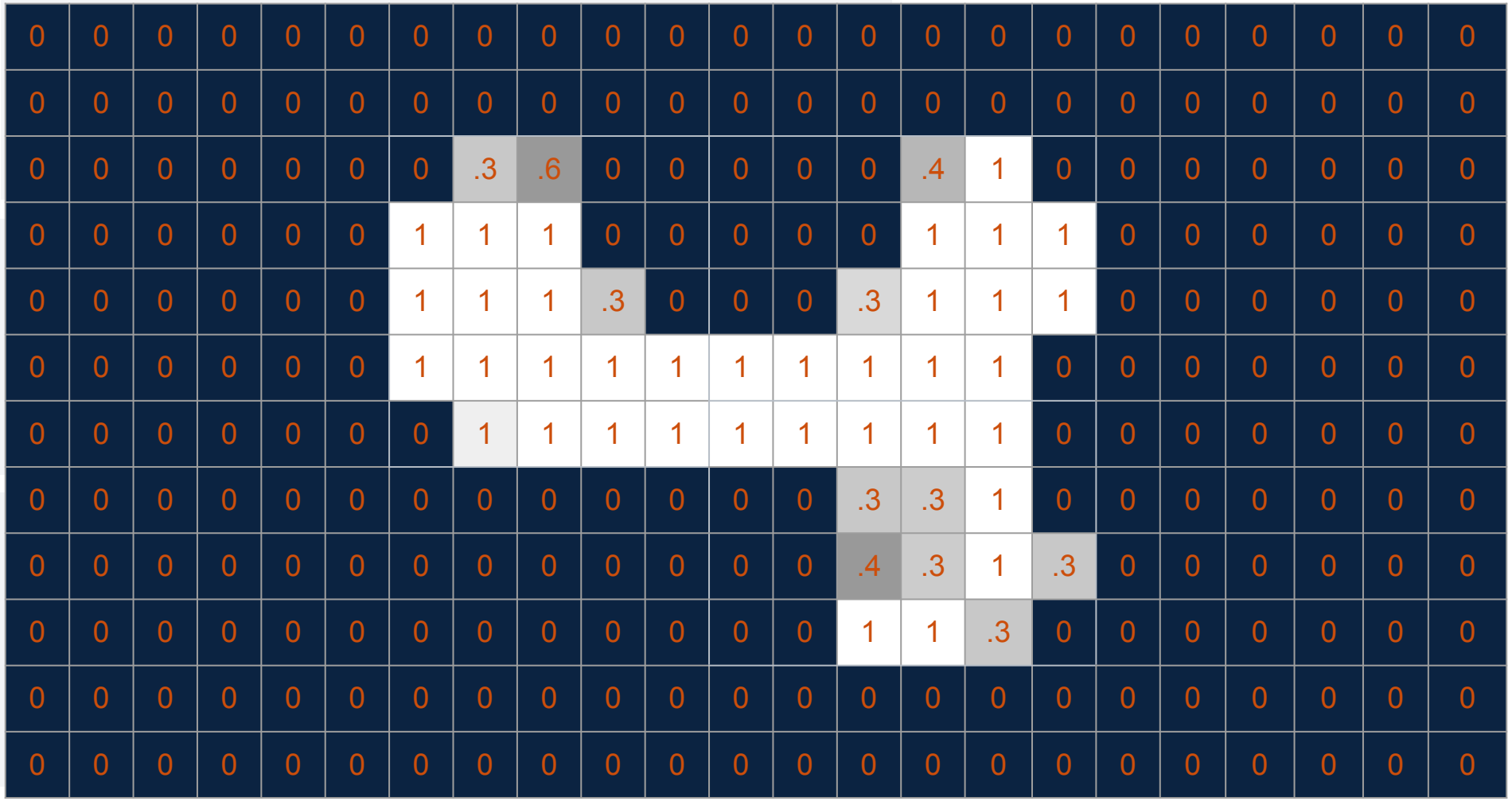


Input 28 x 28 (not to scale)

PREDICTION	
0	?
1	?
2	?
3	?
4	?
5	?
6	?
7	?
8	?
9	?

TRAINING SCHEME - Convolution 1

nn.Conv2d(1,32,3,padding=1), nn.ReLU(), nn.MaxPool2d(2)



Input 28 x 28

- not to scale
- does not include additional padding

Sliding 32 filters scans entire image (3x3 at a time)

x1	x0	x1
x1	x0	x1
x0	x1	x0

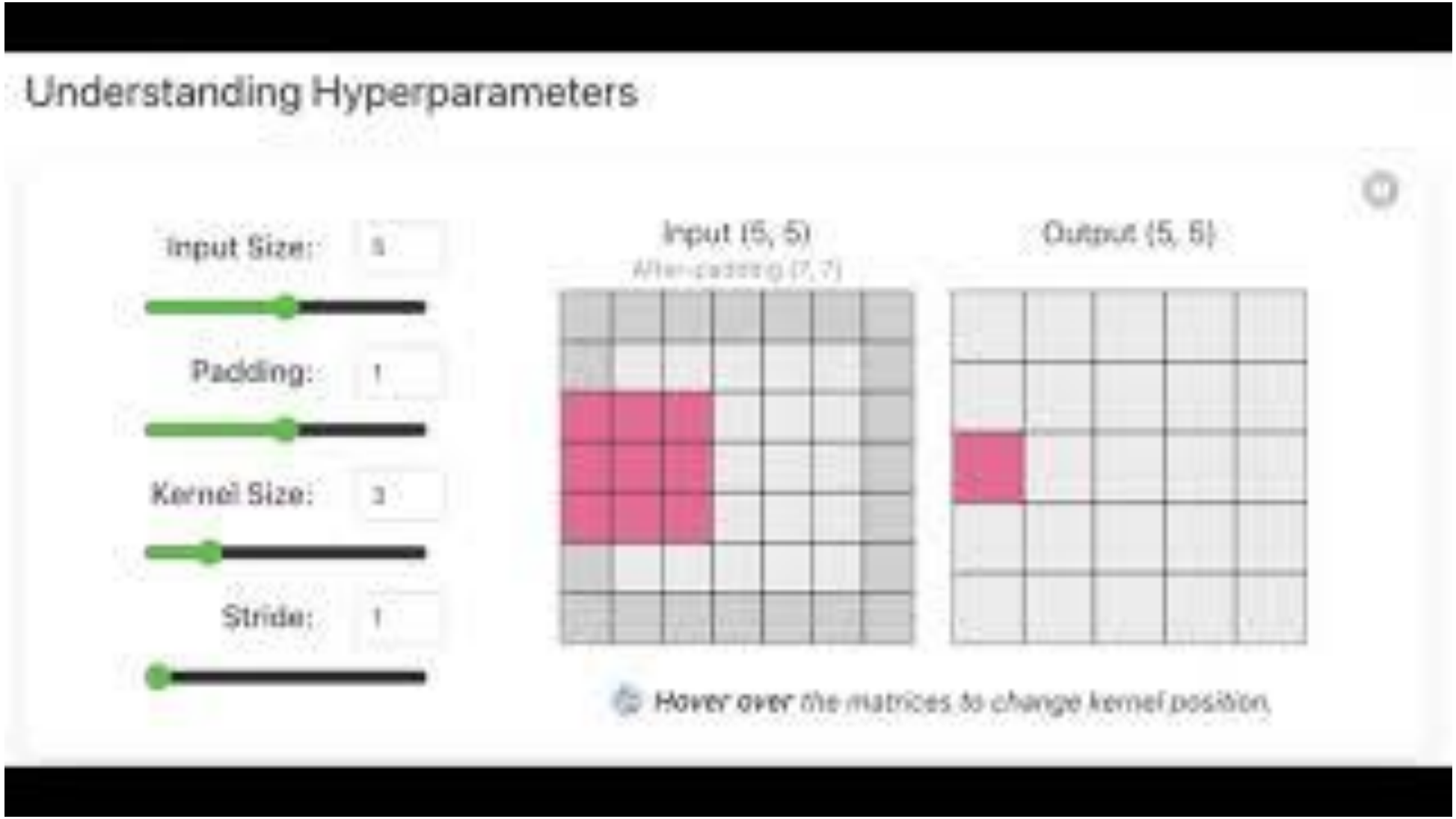
Output After Conv1

- 32 feature maps, 28 x 28

MaxPool(2)

- 32 feature maps, 14 x 14

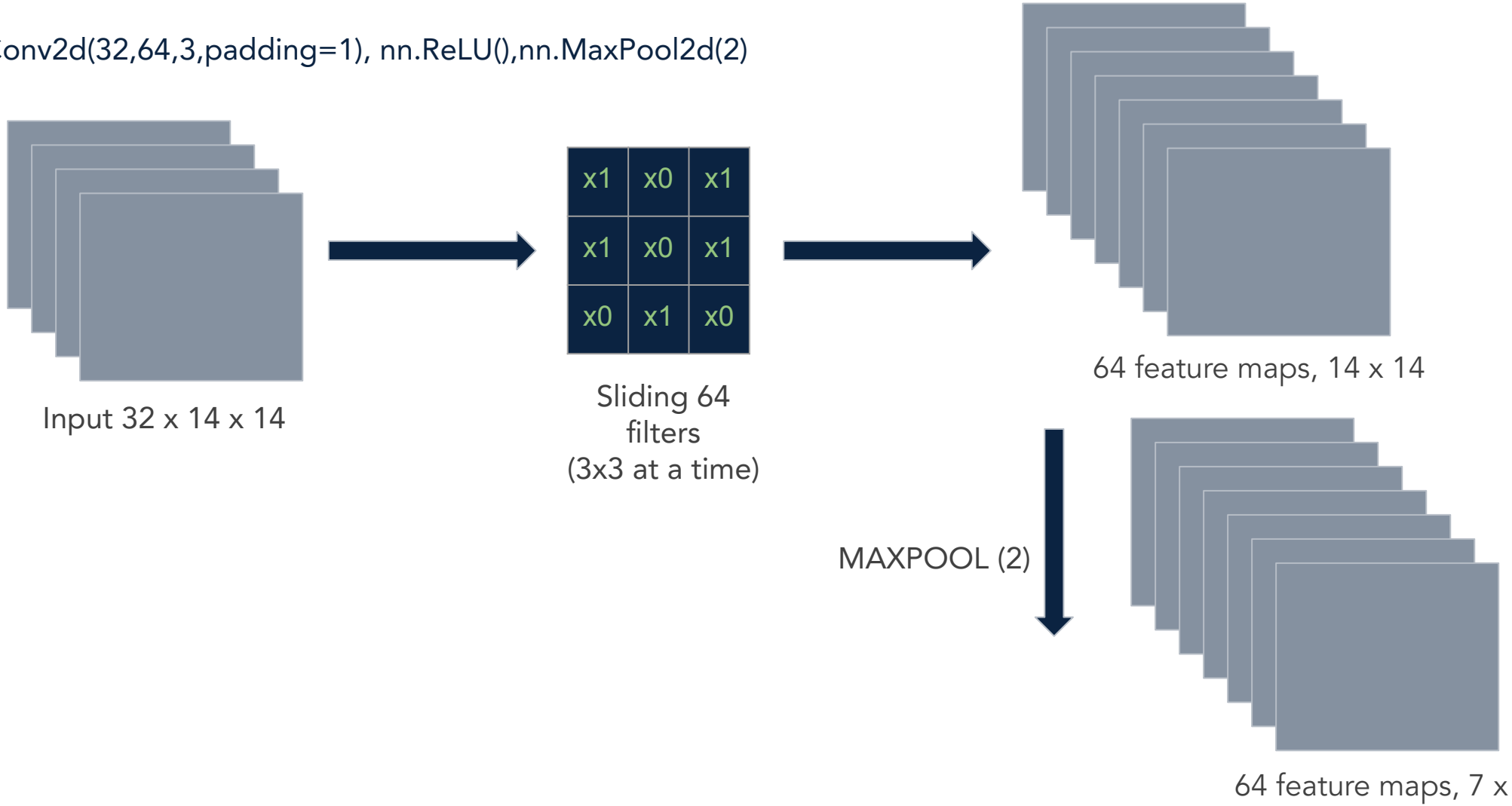
TRAINING SCHEME - Convolution Sliding Filter Example



Content source: CNN Explainer, https://poloclub.github.io/cnn-explainer/

TRAINING SCHEME - Convolution 2

nn.Conv2d(32,64,3,padding=1), nn.ReLU(),nn.MaxPool2d(2)



TRAINING SCHEME - Flatten and Linear

64 feature maps, 7 x 7

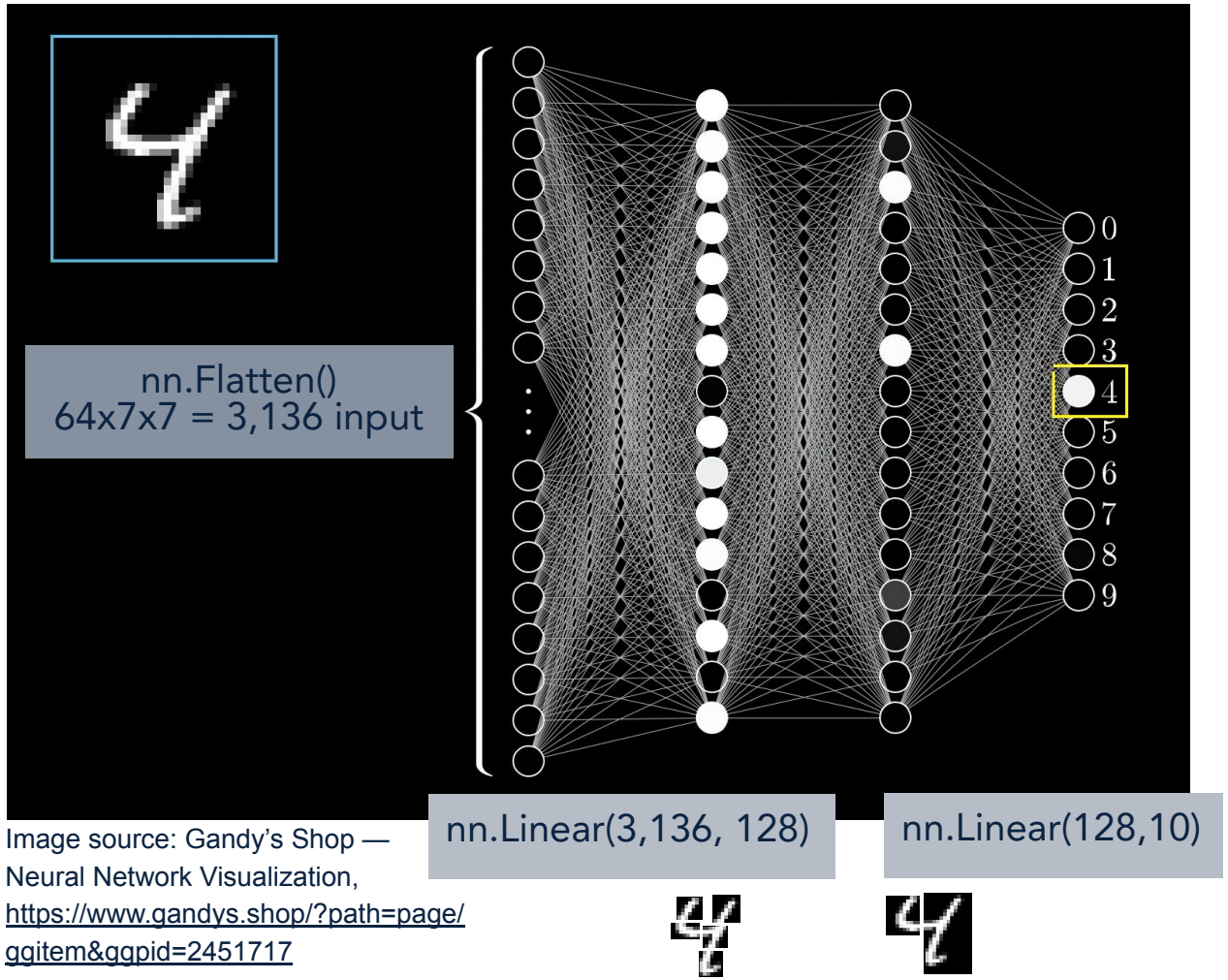


Image source: Gandy's Shop — Neural Network Visualization, https://www.gandys.shop/?path=page/ggitem&ggpid=2451717

PREDICTION	
0	.01
1	.08
2	.05
3	.09
4	.8
5	.01
6	.08
7	.06
8	.08
9	.06

TRAINING SCHEME - Recalibration

- Which weights are updated?

Layer	What's being updated (not including bias)	Count (not including bias)
Conv1(1 input -> 32 filters)	Each 32 filters of size 3x3 kernel	32 filters x (1x3x3) = 288 parameters
Conv2(32 input -> 64 filters)	Each 64 filters of size 3x3 kernel	64 filters x (32x3x3) = 18,432 parameters
Linear(64 * 7 * 7 input -> 128 neurons)	Each connection between 3136 inputs and 128 neurons	3136 x 128 = 401, 408 parameters
Linear(128 input -> 10 neurons)	Final classifier weights	128 x 10 = 1280 parameters

4. Algorithms & model design (FedSGD, FedAvg, SGD)

Federated Learning [COMP 5130 - Intro to OS]

AUBURN UNIVERSITY
Samuel Ginn College of Engineering

MODEL DESIGN — Federated CNN for MNIST

Server

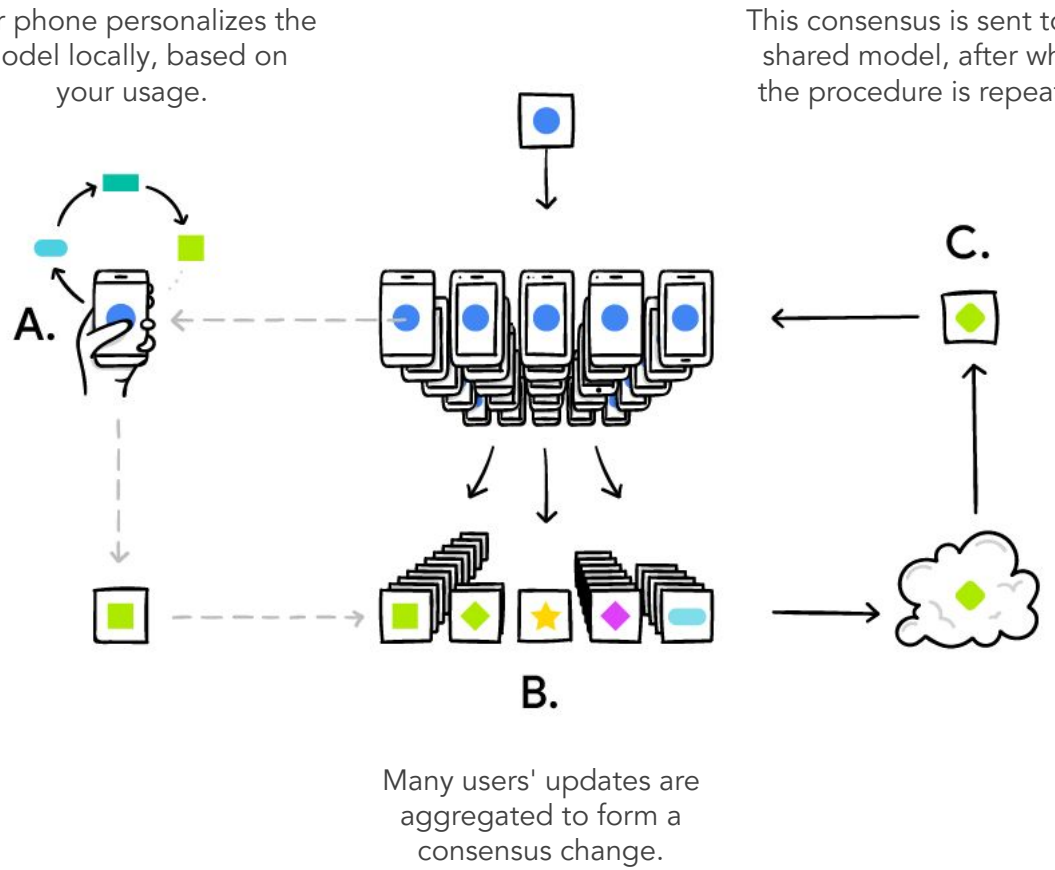
- Distributes the training dataset and learning model, w_t, to each client
- Aggregates/averages all client gradients or model updates, to update the global model.
- Determines if another communication round is needed

Client

- Predict classes for its local data
- Compute cross entropy loss (criterion) of the model predictions
- Compute gradient and update weights
- Return gradients/weights and sample size to the server

Evaluation

- Using the test dataset to evaluate the model predictions
- Calculating model accuracy and total loss



FED AVG Algorithm Implementation

Step 1: Server Selects subset of clients

- Each client is specified by its group by the fraction controlled parameter C
- Example of this would be for 10000 clients if C is 0.10 that would account for 1000 random clients that would be trained within that iteration.

Step 2: Client receives Global model weights

- All clients picked within the sample group receive the same copy of weights.

Step 3: Clients Train locally

- Global model takes advantage of the free computation that can happen at the device level allowing training on the end device
- Improves communication efficiency since all devices do not have to send their parameters back to the global model each round.

Step 4: Communicating back to the global model

- Only model parameters are sent back to the global model after the local training is completed
- Helps provide extra protection to local user data while also improving model performance

FED AVG Strengths And Weaknesses

Strengths

- Computationally inexpensive
- Increased privacy for end users
- Ideal for asynchronous communication for devices

Weakness

- Can be slower than global training
- Need to account for dropout and latency based on thresholds
- Cost of accounting clients to subsets can become expensive

5. Evaluation protocol, results & analysis

Federated Learning [COMP 5130 - Intro to OS]

AUBURN UNIVERSITY
Samuel Ginn College of Engineering

EVALUATING THE EFFECTIVENESS

Evaluation mechanism

- Post communication round

  Server evaluates the updated global model using the MNIST test dataset (10,000 unseen images).

Effectiveness measures

- Accuracy: percentage of correctly predicted labels
- Cross Entropy Loss: logarithmic loss of the model prediction of the true label
  - High confidence -> low loss
    - L = -log(0.95)
  - Uncertain or Wrong prediction -> high lose
    - L = -log(0.2)
- Communication Rounds: measures training efficiency
- Training Time: measures computation cost

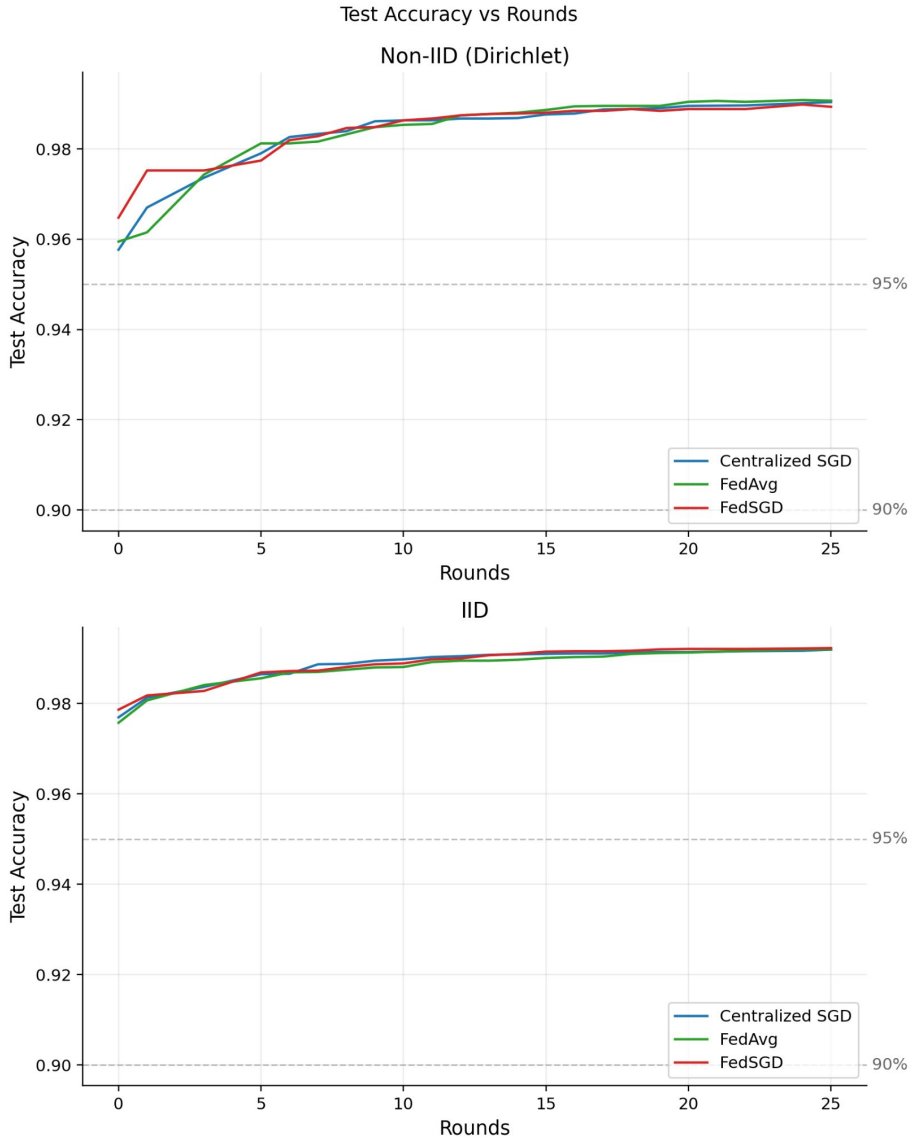
EVALUATION - Test Accuracy vs Rounds (IID vs Non-IID)

Key takeaways

- All methods cross 95% by ~3 rounds; by 20–25 rounds they all plateau near ~99%.
- Under Non-IID, FedSGD rises fastest in the first few rounds (less client-drift with E=1), but FedAvg and Centralized catch up by ~10–15 rounds.
- Under IID, curves practically overlap throughout—algorithm choice matters less.

Significance

- With moderate heterogeneity, early convergence favors more frequent synchronization (FedSGD), but final accuracy parity holds across methods.



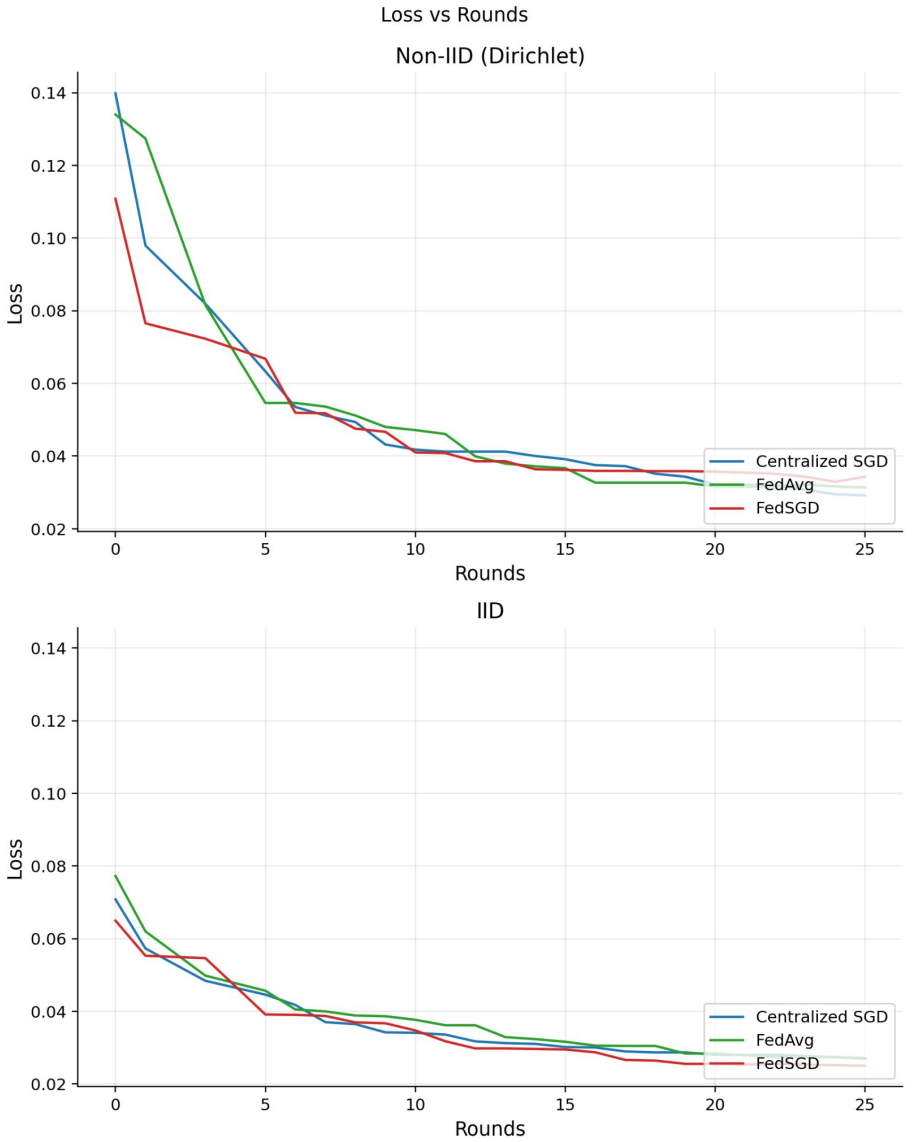
EVALUATION - Loss vs Rounds (IID vs Non-IID)

Key takeaways

- Non-IID starts with higher loss and a bigger early gap; FedSGD drops fastest initially.
- After ~10–15 rounds, all losses converge to a similar floor (~0.03–0.04).
- IID is smoother and slightly lower throughout; methods are nearly indistinguishable past the early phase.

Why it matters

- Heterogeneity mainly impacts early optimization stability; with our hyper-params, end-state quality is robust.



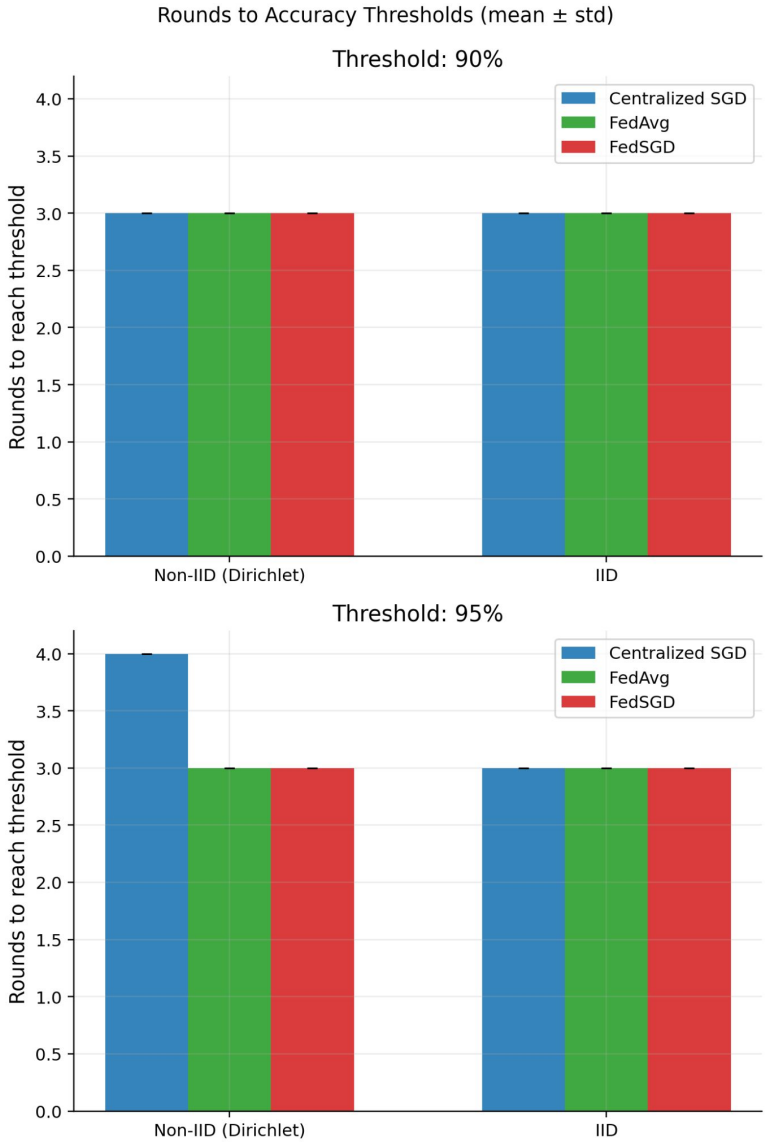
EVALUATION - Rounds to Accuracy Thresholds (90% / 95%)

Key takeaways

- 90%: ~3 rounds across the board (IID & Non-IID).
- 95%: Non-IID: Centralized-SGD ~4 rounds, FedAvg/FedSGD ~3. IID: all ~3.
- Early-round efficiency of FL is not worse than centralized here; frequent syncing (FedSGD) shines under heterogeneity.

Why it matters

- For “time-to-first-useful-model,” FL is competitive (and sometimes faster) even with non-IID data.



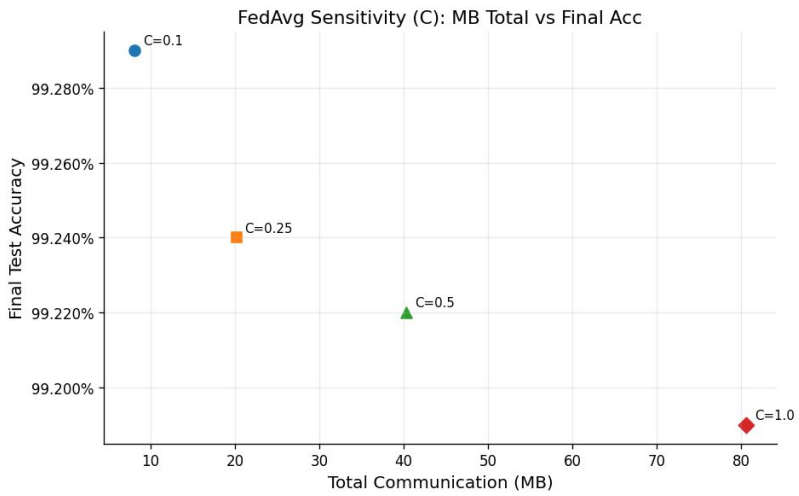
EVALUATION - Test Accuracy vs Time (≈ rounds)

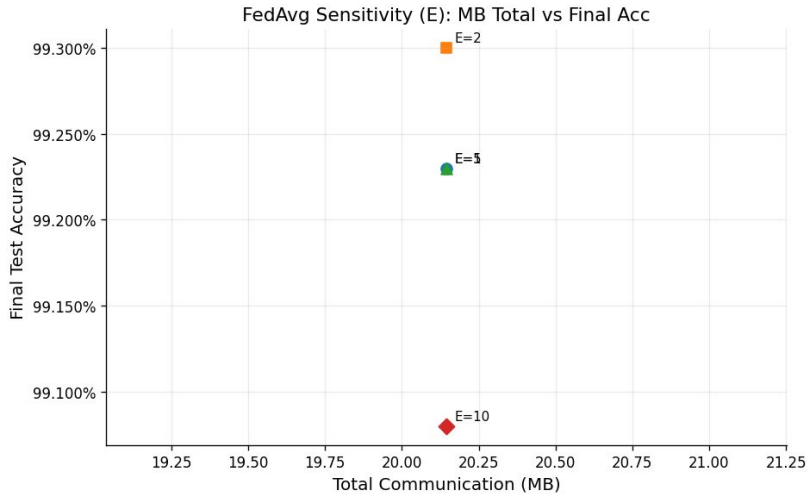
Key takeaways

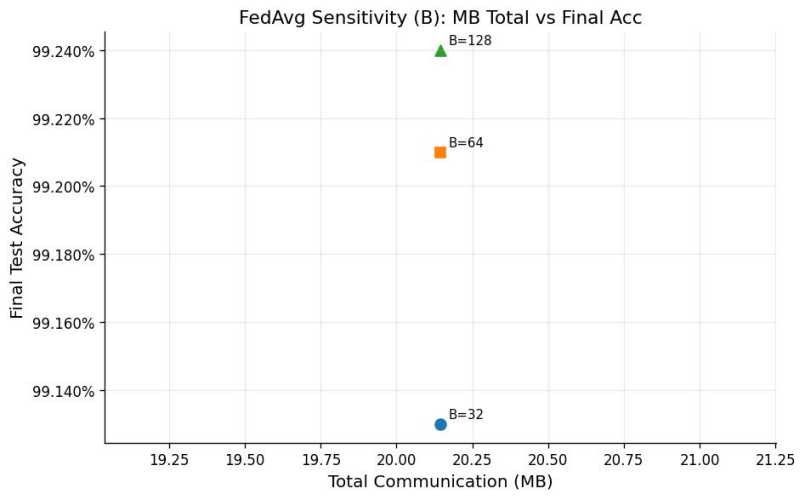
- Relative behaviors match the rounds plot: FedSGD leads early on Non-IID; parity by ~10–15 rounds; ~99% by ~25.
- This view is ready to become a true wall-clock comparison once per-round time is logged.

Why it matters

- For deployment decisions you'll want Acc vs wall-clock (and Acc vs MB). The plot is presentation-ready now and will drop in actual seconds once we log time_sec.







6. Code, demo & environment

Federated Learning [COMP 5130 - Intro to OS]

AUBURN UNIVERSITY
Samuel Ginn College of Engineering

CODE DEMONSTRATION

Questions?