# Federated Learning - FedAVG by Google

**Reference Paper:**
Brendan McMahan, H., Moore, E., Ramage, D., Hampson, S., & Agüera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 54.*

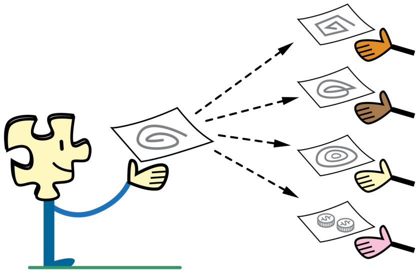
**Bahy** Helmi Hartoyo Putra
bahyhelmi97@gmail.com

# Federated Learning

# **Federated Learning** - Background

- **Phones and tablets** are the **primary computing devices** for many people now.
- These modern edge devices contain **rich data** as human use it **frequently.**
- However, this rich data is often **privacy sensitive**, **large in quantity**, or both.
- A new way to **utilize** those data **without breaking** the **privacy** and **efficiency** issues must be developed.

# **Federated Learning -** Definition



- A **learning technique** that allows users to collectively **reap the benefits of shared models** trained from rich data, **without** the need to **centrally store it.**
- A **decentralized learning** approach, that leaves the training data distributed on the mobile/edge devices, and learns a shared model by **aggregating locally-computed updates.**

# **Federated Learning -** Advantages

- **Decoupling** of **model training** from the need for direct access to the raw training data.
- Significantly **reduce privacy and security risks** by limiting the attack surface to only the device, rather than the device and the cloud.
- Reap the benefits of **shared models** which **will greatly improve the user experience** on the user device.

# Federated Learning - Use Cases

- **Image classification**;  predicting the likelihood of which kind of photos are being shared, deleted, or viewed.
    - photos people take on their phone are likely quite different (unique, rich) than typical internet photos.
- **Language modelling**; improve voice recognition and text entry on touch-screen.
    - the use of language in chat and text messages is generally much different than standard language corpora.

# **Federated Learning** - Related Work (1)

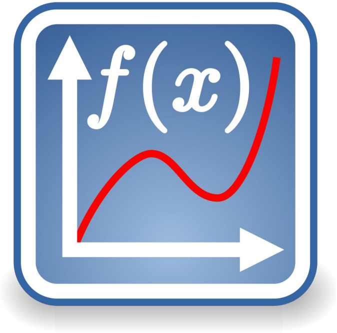
- **Distributed training** by iteratively averaging locally trained models for:
  - perceptron (McDonald et al., 2010)
  - speech recognition DNNs (Povey et al., 2015)
- **Asynchronous distributed training** with "soft" averaging, **not considering** unbalance and non-IID data (Zhang et al., 2015)
- Advantages of **keeping sensitive user data** on the device (Neverova et al., 2016)

# **Federated Learning -** Related Work (2)

- **Privacy preserving deep learning**, not considering unbalanced and non-IID data with a limited empirical evaluation (Shokri & Shmatikov, 2015)
- **Large scale distributed** (asynchronous) SGD, with a huge number of updates (Dean et al., 2012)

# Federated Optimization

# Federated Optimization - Definition




- The optimization problem implicit in **federated learning.**
- Has several **key properties** that **differentiate** it from a typical **distributed optimization** problem.
- **Communication efficiency** is the greatest factor of importance.

# Federated Optimization - Key Properties

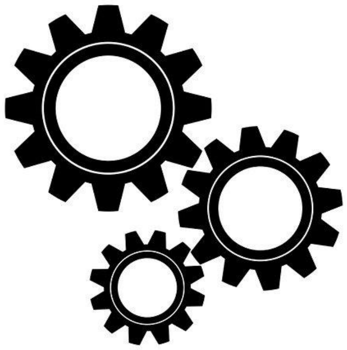
- **Non-IID;** training data on a given client is typically different, any particular user's local dataset will not be representative of the population distribution.
- **Unbalanced similarly;** some users make heavier use of the app than others, lead to varying kinds of local training data.
- **Massively distributed;** the number of clients participating in an optimization to be much larger than the average number of examples per client.
- **Limited communication;** mobile devices are frequently offline or slow or cause expensive connections.

# **Federated Optimization -** Limitations

- Federated optimization **experiment** in this paper will **used a controlled environment**, but **still addresses the key issues** of client availability, unbalanced, and non-iid data.
- Only **using a fraction of clients for efficiency**, as the experiments show diminishing returns for adding more clients beyond a certain point.

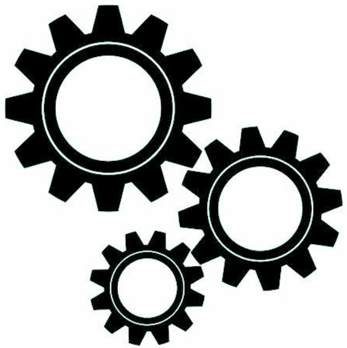
# Federated Algorithms FedSGD & FedAVG

# **Federated SGD -** Definition



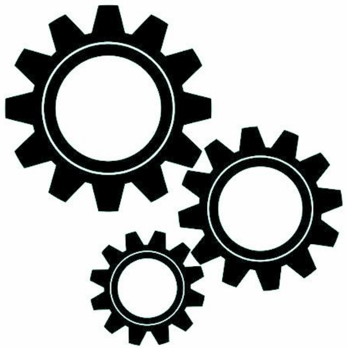
- A **large-batch synchronous SGD** applied **at a federated setting**.
- To apply this approach in the federated setting, we **select a C fraction of clients on each round**, and **compute the gradient** of the loss over all the data held by these clients.
- **C controls the global batch size**, with C = 1 corresponding to full-batch (non-stochastic) gradient descent.

# Federated AVG - Definition



- A **generalization of FedSGD**, which allows local nodes to **perform more than one batch update** on local data
- Computation is controlled by **three key** parameters:
  - **C**, the fraction of clients perform computation in each round.
  - **E**, local epoch client makes on its local dataset in each round.
  - **B**, the local minibatch size used for the client updates.
- Using $B = \infty$ and $E = 1$ (full local dataset treated as a single minibatch with one local epoch), will corresponds exactly to **FedSGD**.

# Federated AVG - Algorithm



**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ 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**
    $w_{t+1}^k \leftarrow$ 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

Fraction of available clients on the given round

Weighted-average of model parameters from all clients

Allow client to do more than one batch local update

# Experiment Results (Original)

# **Initial Experiment** - MNIST Digit Recognition

- **Model used:**

**Simple multilayer perceptron (2NN):**

- Input Layer (28x28, Flatten)
- Hidden Layer (200 units, ReLU)
- Hidden Layer (200 units, ReLU)
- Output Layer (10 units, Softmax)

Total trainable param: 199,210

**Convolutional Neural Network (CNN):**

- Conv. Layer (32 Channels, Kernel Size 5x5)
- Max Pooling (Pool Size 2x2)
- Conv. Layer (64 Channels, Kernel Size 5x5)
- Max Pooling (Pool Size 2x2)
- Flatten Layer
- Hidden Layer (512 units, ReLU)
- Output Layer (10 units, Softmax)

Total trainable param: 1,663,370

# **Initial Experiment** - MNIST Digit Recognition

- **Data used:**
  - **MNIST Digit Recognition** (tf.keras.datasets.mnist.load_data)
  - **Train data:** 60,000 images
  - **Test data:** 10,000 images
- **Distributed as follows:**
  - **IID:**
    - Train data was shuffled, distributed equally to 100 clients. Each client received 600 random images.
  - **NON-IID:**
    - Train data was sorted by label, distributed equally to 100 clients based on the sorted version. Each client received 600 random images, containing maximum 2 labels.

# **Initial Experiment** - MNIST Digit Recognition

- **Experiment done by:**
  - **Increasing parallelism.** Varying the number of client fraction (C). Trying C = {0.0, 0.1, 0.2, 0.5, 1.0}. Each C corresponds to the proportion of clients participate at the given round.
  - **Increasing computation.** Client fraction (C) was fixed to 0.1 (10 clients). The computation (number of updates) per client was varied by either decreasing B (local batch size), increasing E (local epoch), or both.

# **Initial Experiment** - MNIST Digit Recognition

- **Results:** Increasing parallelism

| **2NN** | IID | | Non-IID | |
|---|---|---|---|---|
| $C$ | $B = \infty$ | $B = 10$ | $B = \infty$ | $B = 10$ |
| 0.0 | 1455 | 316 | 4278 | 3275 |
| 0.1 | 1474 (1.0×) | 87 (3.6×) | 1796 (2.4×) | 664 (4.9×) |
| 0.2 | 1658 (0.9×) | 77 (4.1×) | 1528 (2.8×) | 619 (5.3×) |
| 0.5 | — (—) | 75 (4.2×) | — (—) | 443 (7.4×) |
| 1.0 | — (—) | 70 (4.5×) | — (—) | 380 (8.6×) |
| **CNN**, $E = 5$ | | | | |
| 0.0 | 387 | 50 | 1181 | 956 |
| 0.1 | 339 (1.1×) | 18 (2.8×) | 1100 (1.1×) | 206 (4.6×) |
| 0.2 | 337 (1.1×) | 18 (2.8×) | 978 (1.2×) | 200 (4.8×) |
| 0.5 | 164 (2.4×) | 18 (2.8×) | 1067 (1.1×) | 261 (3.7×) |
| 1.0 | 246 (1.6×) | 16 (3.1×) | — (—) | 97 (9.9×) |

# **Initial Experiment** - MNIST Digit Recognition

- **Results:** Increasing parallelism
  - **With B = 600** (treating all examples in client as one batch), there is **only a small advantage** in increasing the client fraction.
  - **Using the smaller batch size** B = 10 **shows a significant improvement** in using C ≥ 0.1, especially in the non-IID case.

# **Initial Experiment** - MNIST Digit Recognition

- **Results:** Increasing computation

MNIST CNN, 99% ACCURACY

| CNN | $E$ | $B$ | $u$ | IID | NON-IID |
|---|---|---|---|---|---|
| FEDSGD | 1 | $\infty$ | 1 | 626 | 483 |
| FEDAVG | 5 | $\infty$ | 5 | 179 (3.5×) | 1000 (0.5×) |
| FEDAVG | 1 | 50 | 12 | 65 (9.6×) | 600 (0.8×) |
| FEDAVG | 20 | $\infty$ | 20 | 234 (2.7×) | 672 (0.7×) |
| FEDAVG | 1 | 10 | 60 | 34 (18.4×) | 350 (1.4×) |
| FEDAVG | 5 | 50 | 60 | 29 (21.6×) | 334 (1.4×) |
| FEDAVG | 20 | 50 | 240 | 32 (19.6×) | 426 (1.1×) |
| FEDAVG | 5 | 10 | 300 | 20 (31.3×) | 229 (2.1×) |
| FEDAVG | 20 | 10 | 1200 | 18 (34.8×) | 173 (2.8×) |

# **Initial Experiment** - MNIST Digit Recognition

- **Results:** Increasing computation
  - Increasing computation (local updates) in each client is **effective**.
  - In **IID setting**, using more computation per client **decreases the number of rounds** to reach the target accuracy by **35× for the CNN** and **46× for the 2NN.**
  - In **non-IID setting**, the number is smaller but still substantial, with **2.8 – 3.7×** decrease.

# Experiment Results (Reproduce)

# Experiment Result - Task 1

1) Simulate the following cases in Table 1 in [1] with validation sets.

(i) 2NN, IID, B=10, C=0.1, E=1

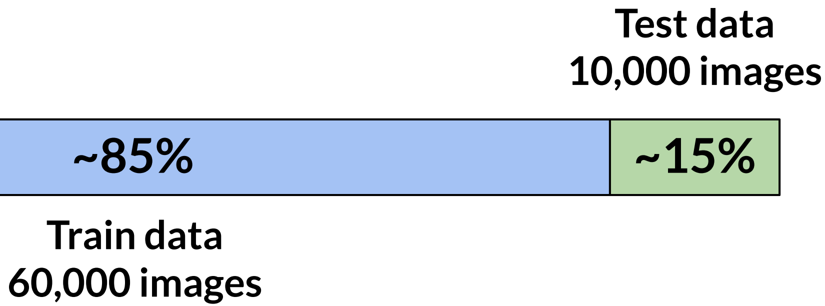
(ii) 2NN, Non-IID, B=10, C=0.1, E=1

- Use the validation set and explain how the validation set was chosen.

- For each case in the table above, provide two figures for accuracy and loss. In each figure, include three curves for training, validation, and test data. Define the accuracy and loss used.

# Experiment Result - Task 1

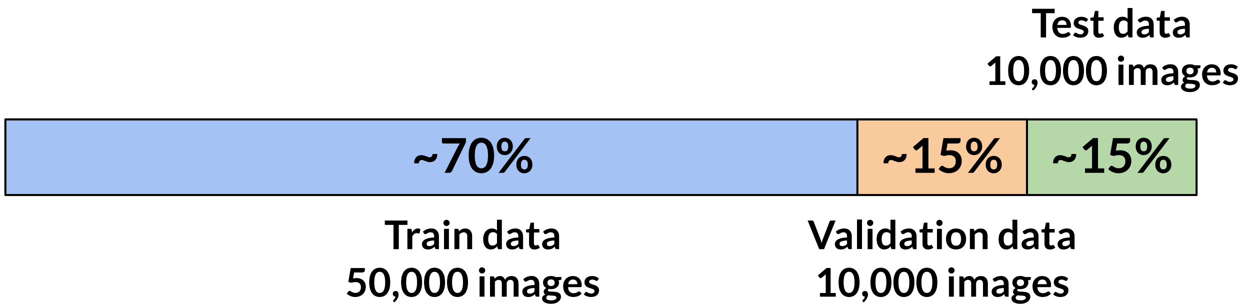
- **How the validation set was chosen?**
  - **MNIST data loaded** from the Tensorflow datasets API.
  - Divide the data into **train data** and **test data** based on the default TF partition settings.
  - **Shuffle** the **train data** before doing any splitting.




**Original TF MNIST Digit dataset compartment**

# Experiment Result - Task 1

- **How the validation set was chosen?**
  - The **original train data** splitted into two parts, **new train data** and **validation data.**
  - Splitting done by selecting **10,000 images** randomly, unstratified, from the **original train data** as **validation data.**




**Modified TF MNIST Digit dataset compartment**

# Experiment Result - Task 1

- **Reproduce Table [1] : 2NN with Validation Set**
  - Dataset used for this experiment is MNIST Digit Recognition from Tensorflow. **Train data** contains **50,000 images, validation data** contains **10,000 images** and **test data** contains **10,000 images**.
  - Threshold used for stopping the training process of the 2NN is **0.97.**
  - A learning rate of **0.1** was used.

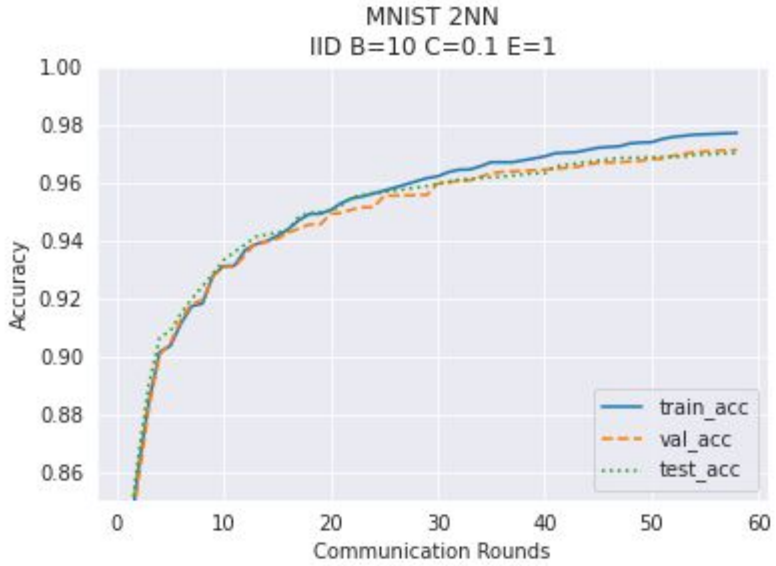
# Experiment Result - Task 1

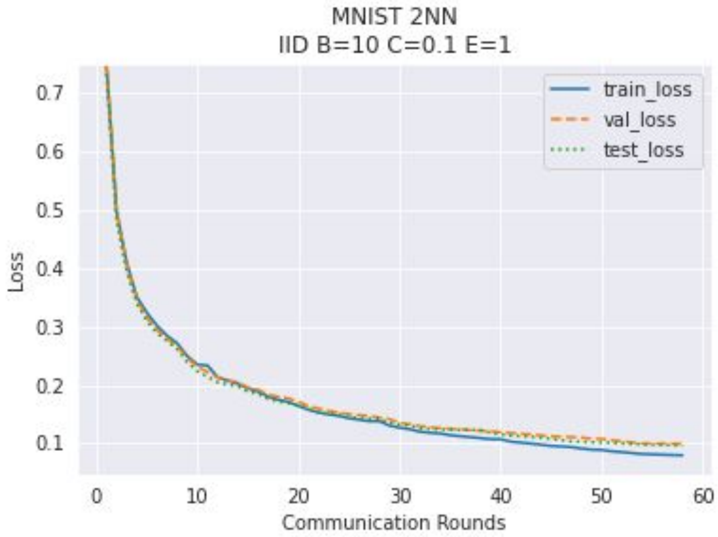
- **Reproduce Table [1] : 2NN with Validation Set**
  - The training graph presented is **plotted monotonically improving**, done by taking the best test accuracy among the obtained values from previous communication rounds.
  - Metrics used:
    - Accuracy: `AccuracyScore`*
    - Loss: `SparseCategoricalCrossentropy`

*calculated by comparing the ground truth label and index of the maximum probability output by each prediction

# Experiment Result - Task 1

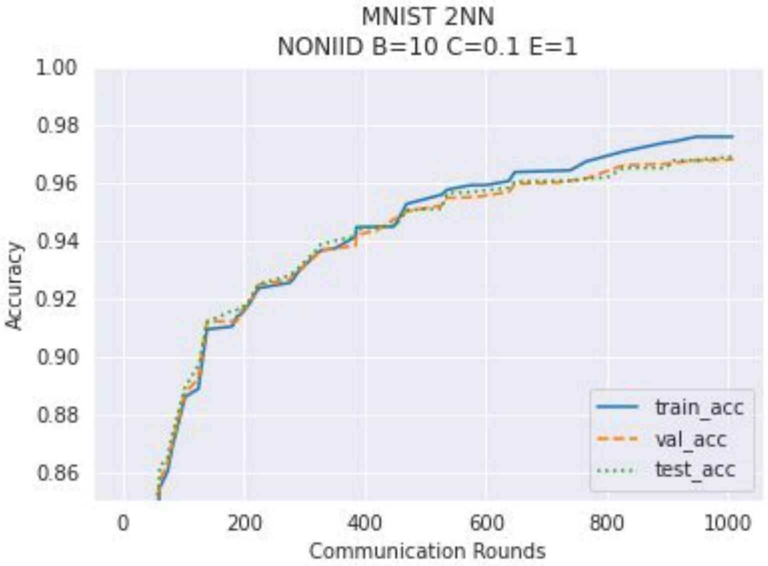
- **2NN, IID, B=10, C=0.1, E=1 (lr=0.1)**

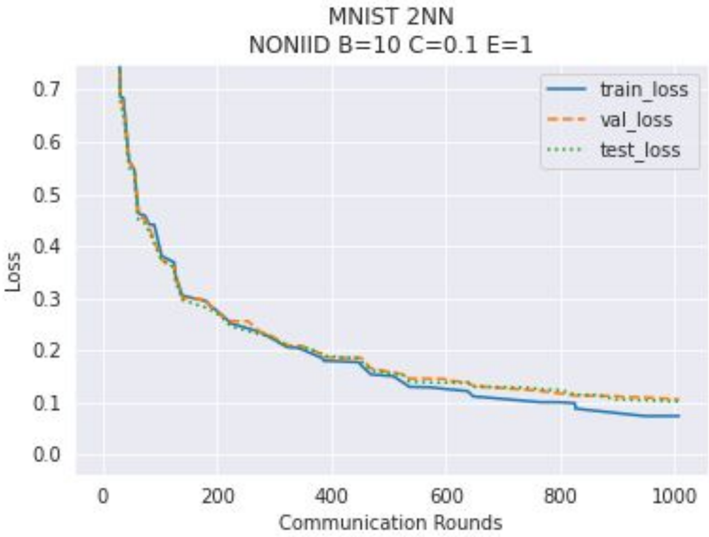


# Experiment Result - Task 1

- **2NN, NONIID, B=10, C=0.1, E=1 (lr=0.1)**



# Experiment Result - Task 1

- **Reproduce Table [1]: Summary Table**

| Case | Communication Round | Train Accuracy | Validation Accuracy | Test Accuracy |
|---|---|---|---|---|
| 2NN, IID, B=10, C=0.1, E=1 | 58 | 0.9769 | 0.9711 | **0.9700** |
| 2NN, NONIID, B=10, C=0.1, E=1 | 1008 | 0.9756 | 0.9683 | **0.9702** |

# Experiment Result - Task 1

- **Key Takeaways:**
  - Using the same parameters (B, E, C), **faster convergence happens in IID setting**.
  - **Validation set is a good estimate of the testing set**. The testing set margin with the validation set is only 0.0011 - 0.0019, compared to its margin with the train set 0.0031 - 0.0046.

# Experiment Result - Task 2

2) Reproduce the following table in [1] for $C = 0.0, 0.1, 1.0$.

| 2NN | IID | | Non-IID | |
|---|---|---|---|---|
| $C$ | $B = \infty$ | $B = 10$ | $B = \infty$ | $B = 10$ |
| 0.0 | 1455 | 316 | 4278 | 3275 |
| 0.1 | 1474 (1.0×) | 87 (3.6×) | 1796 (2.4×) | 664 (4.9×) |
| 0.2 | 1658 (0.9×) | 77 (4.1×) | 1528 (2.8×) | 619 (5.3×) |
| 0.5 | — (—) | 75 (4.2×) | — (—) | 443 (7.4×) |
| 1.0 | — (—) | 70 (4.5×) | — (—) | 380 (8.6×) |
| CNN, $E = 5$ | | | | |
| 0.0 | 387 | 50 | 1181 | 956 |
| 0.1 | 339 (1.1×) | 18 (2.8×) | 1100 (1.1×) | 206 (4.6×) |
| 0.2 | 337 (1.1×) | 18 (2.8×) | 978 (1.2×) | 200 (4.8×) |
| 0.5 | 164 (2.4×) | 18 (2.8×) | 1067 (1.1×) | 261 (3.7×) |
| 1.0 | 246 (1.6×) | 16 (3.1×) | — (—) | 97 (9.9×) |

- Do not use the validation set.

- For each case in the table above, provide two figures for accuracy and loss. In each figure, include two curves for training and test data. Define the accuracy and loss used.

- You don't need to simulate the case the reference value is not available (i.e., '—') in the table.

# **Experiment Result -** Task 2

- **Reproduce Table [1] : 2NN & CNN**
  - This experiment done to show the effect of **increasing parallelism** to the number of communication rounds.
  - Dataset used for this experiment is [MNIST Digit Recognition]() from Tensorflow. **Train data** contains **60,000 images** and **test data** contains **10,000 images**.
  - Threshold used for stopping the training process is **0.97** for 2NN and **0.99** for the CNN .
  - A learning rate of **0.1** used for 2NN and **0.215** for CNN. Both IID and non-IID setting has no difference in learning rate choice.

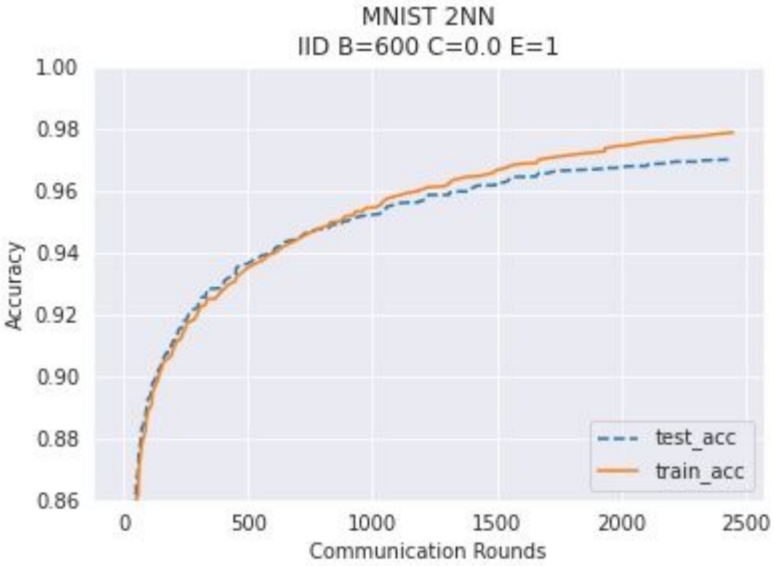
# Experiment Result - Task 2

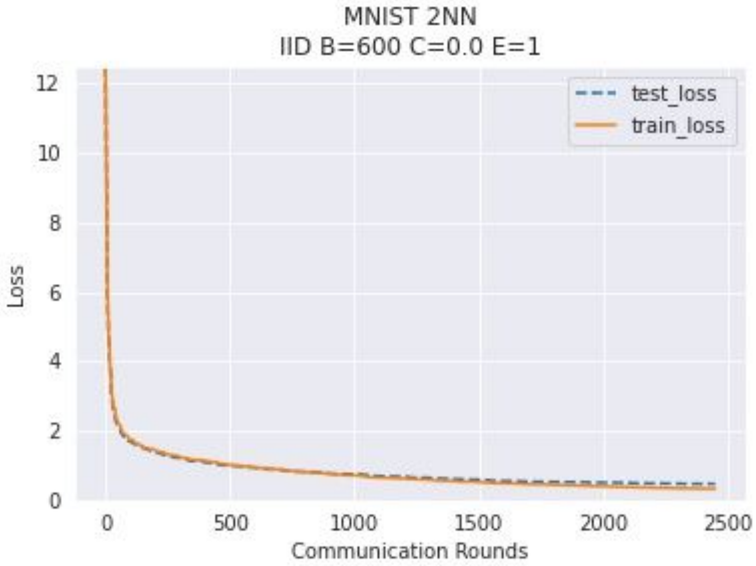
- **Reproduce Table [1] : 2NN & CNN**
  - The training graph presented is **plotted monotonically improving**, done by taking the best test accuracy among the obtained values from previous communication rounds.
  - Metrics used:
    - Accuracy: `AccuracyScore`*
    - Loss: `SparseCategoricalCrossentropy`

*calculated by comparing the ground truth label and index of the maximum probability output by each prediction

# Experiment Result - Task 2

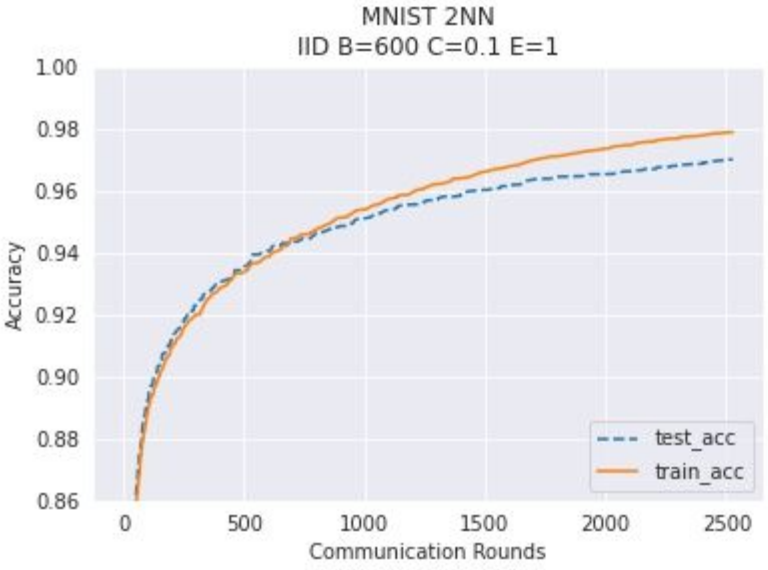
- **2NN, IID, B=600, C=0.0, E=1 (lr=0.1)**

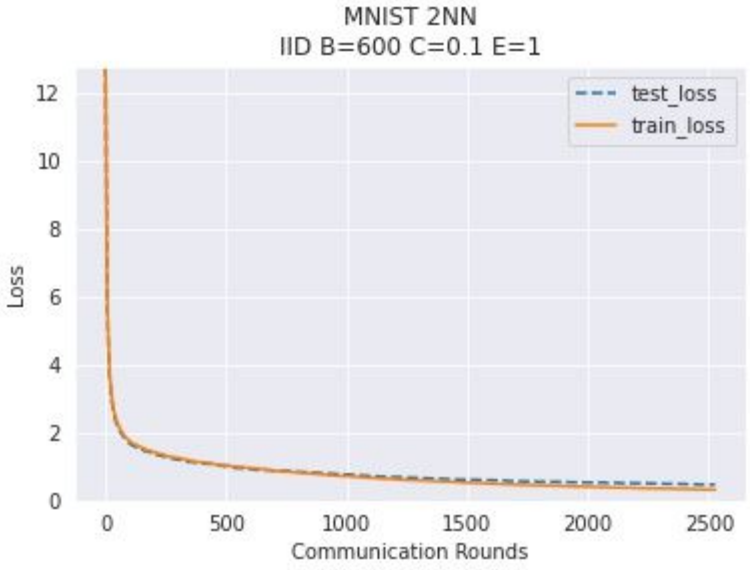


# Experiment Result - Task 2

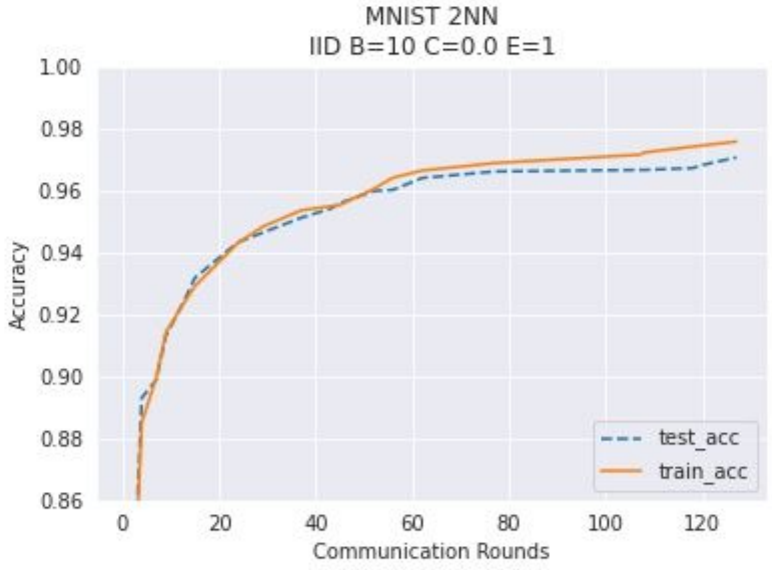
- 2NN, IID, B=600, C=0.1, E=1 (lr=0.1)

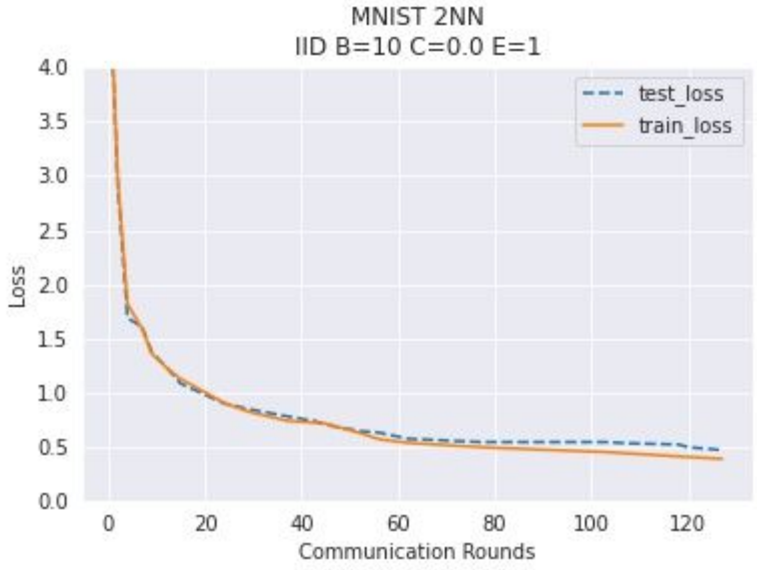


# Experiment Result - Task 2

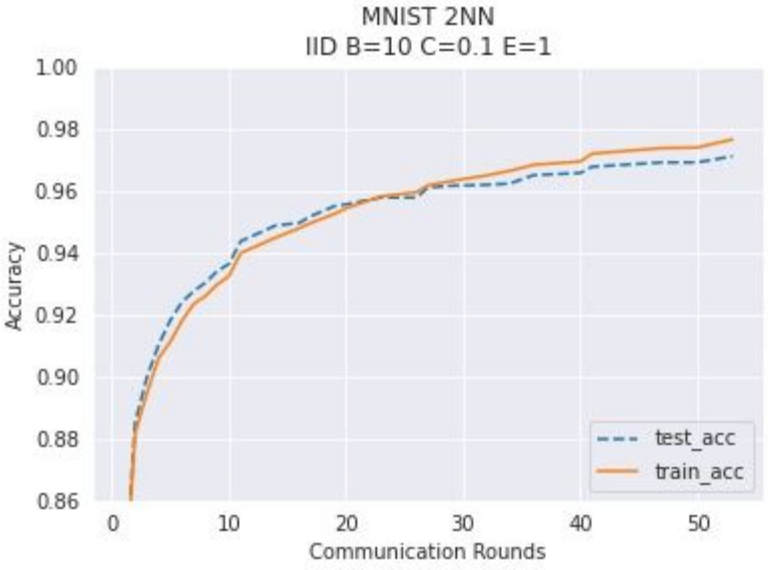
- **2NN, IID, B=10, C=0.0, E=1 (lr=0.1)**

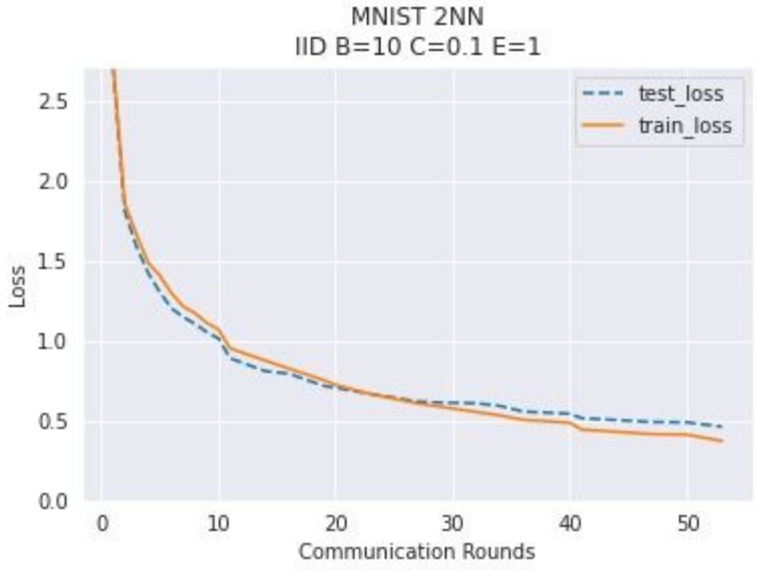


# Experiment Result - Task 2

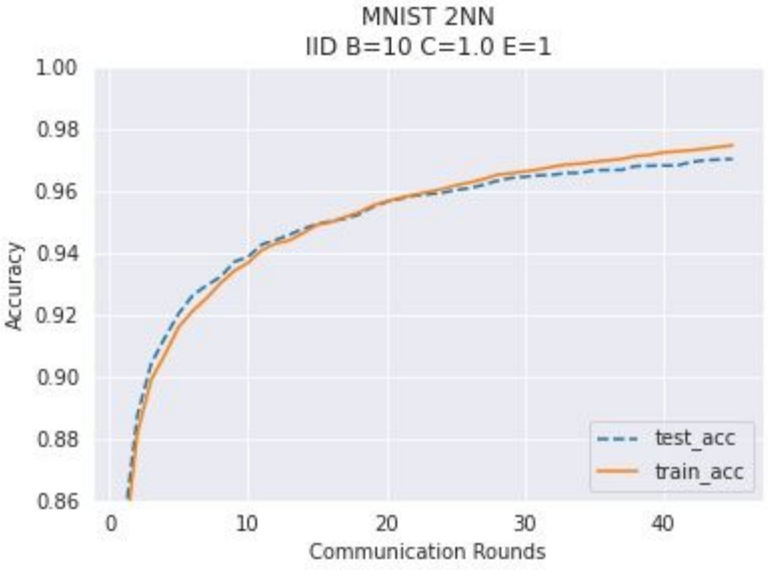
- **2NN, IID, B=10, C=0.1, E=1 (lr=0.1)**

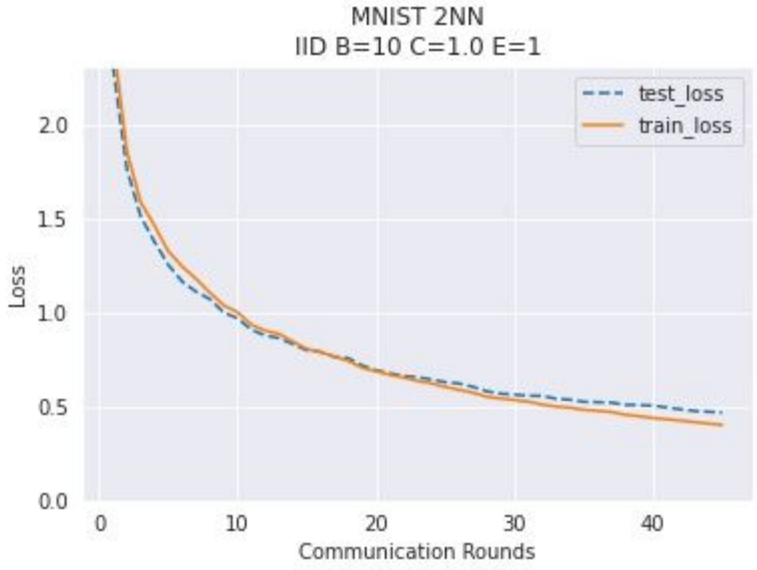


# Experiment Result - Task 2

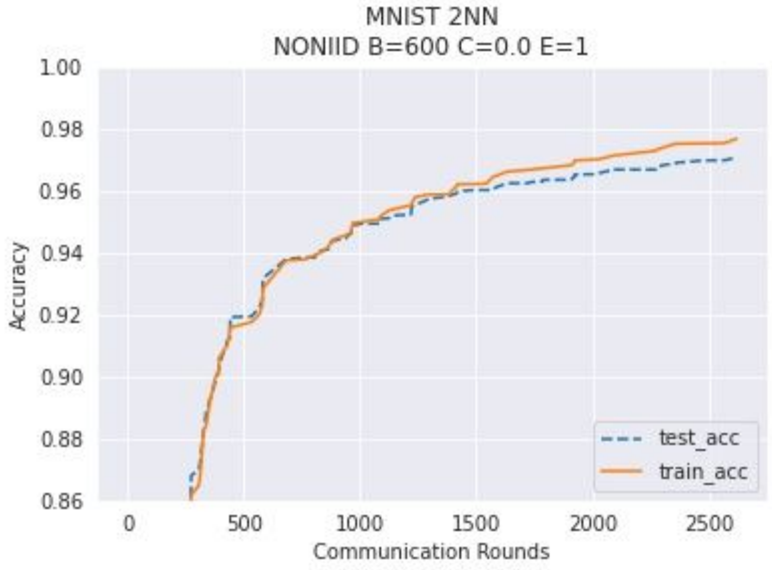
- 2NN, IID, B=10, C=1.0, E=1 (lr=0.1)

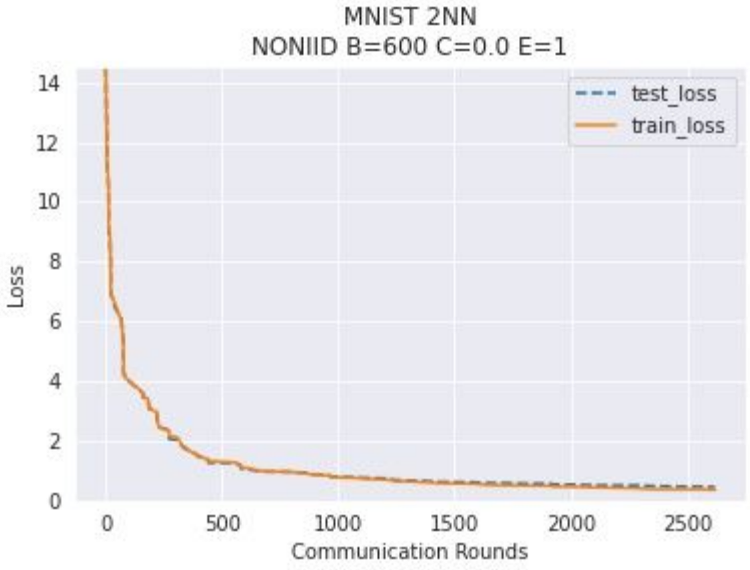


# Experiment Result - Task 2

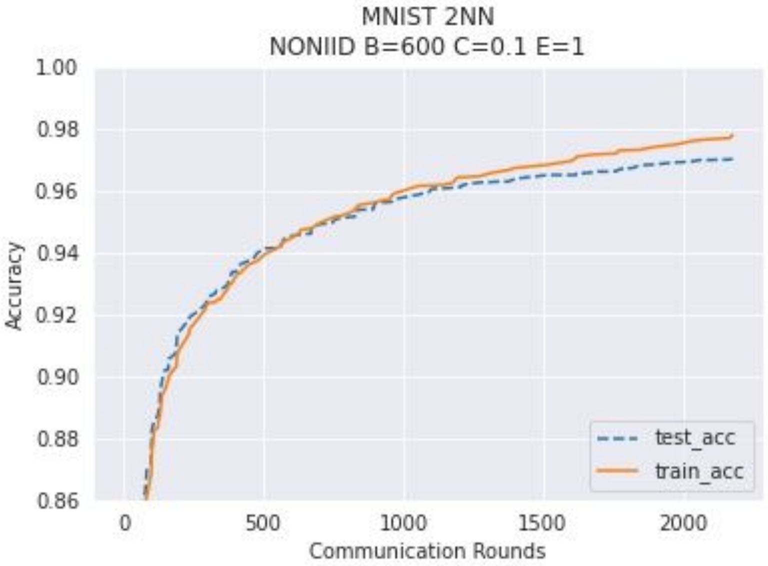
- **2NN, NONIID, B=600, C=0.0, E=1 (lr=0.1)**

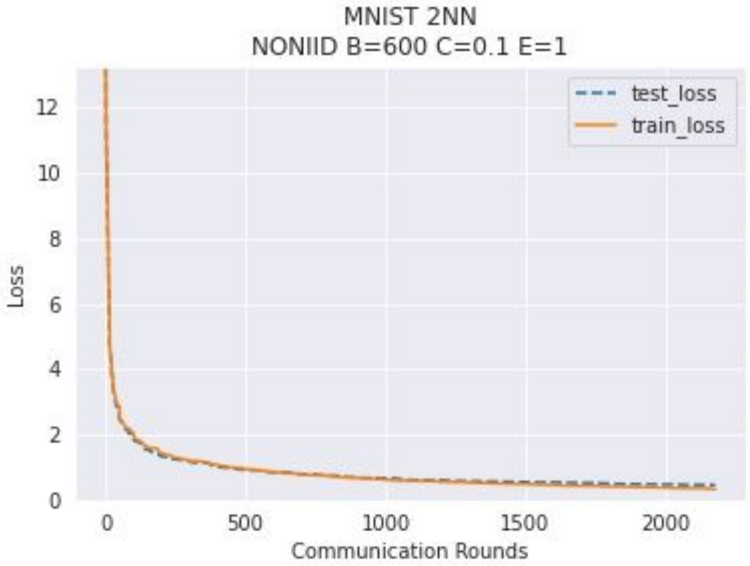


# Experiment Result - Task 2

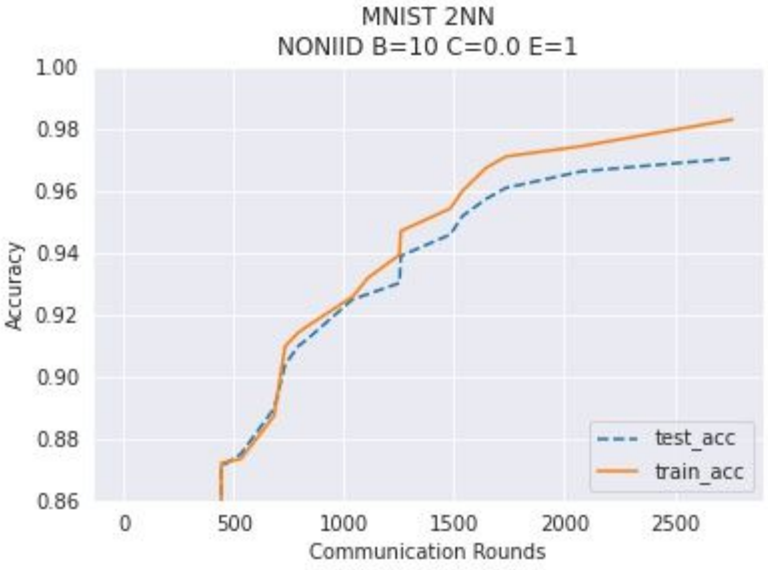
- **2NN, NONIID, B=600, C=0.1, E=1 (lr=0.1)**

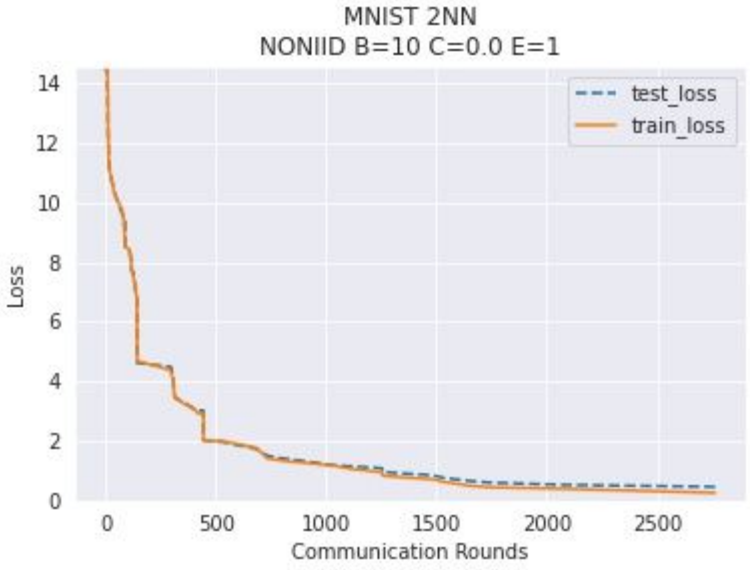


# Experiment Result - Task 2

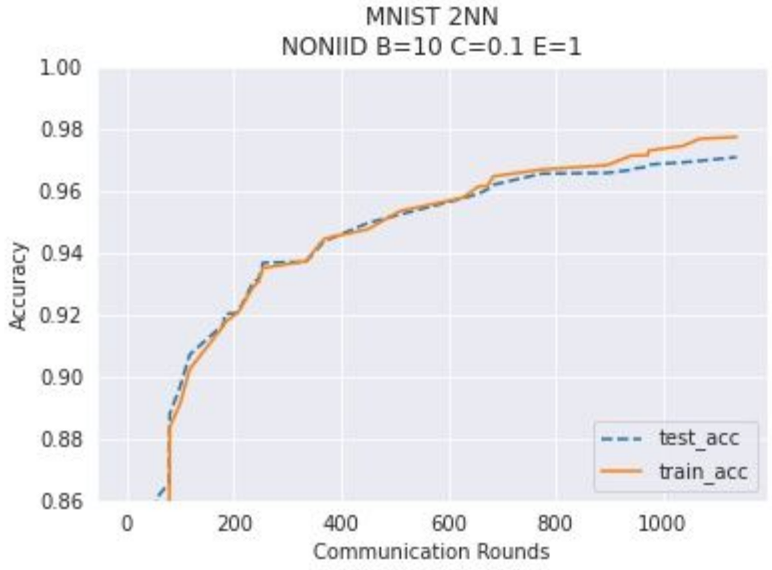
- **2NN, NONIID, B=10, C=0.0, E=1 (lr=0.1)**

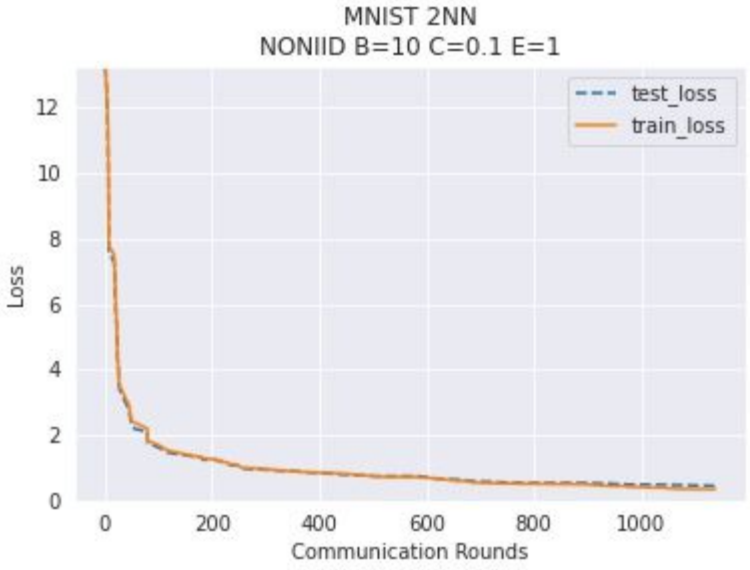


# Experiment Result - Task 2

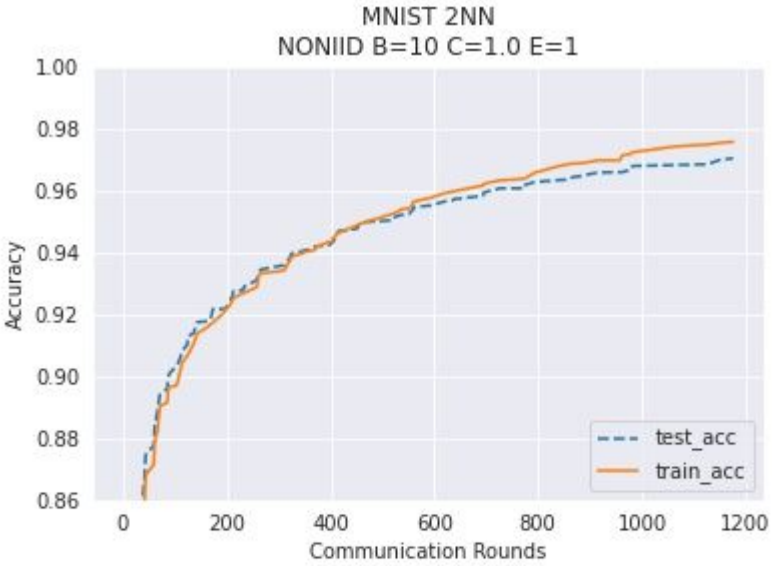
- **2NN, NONIID, B=10, C=0.1, E=1 (lr=0.1)**

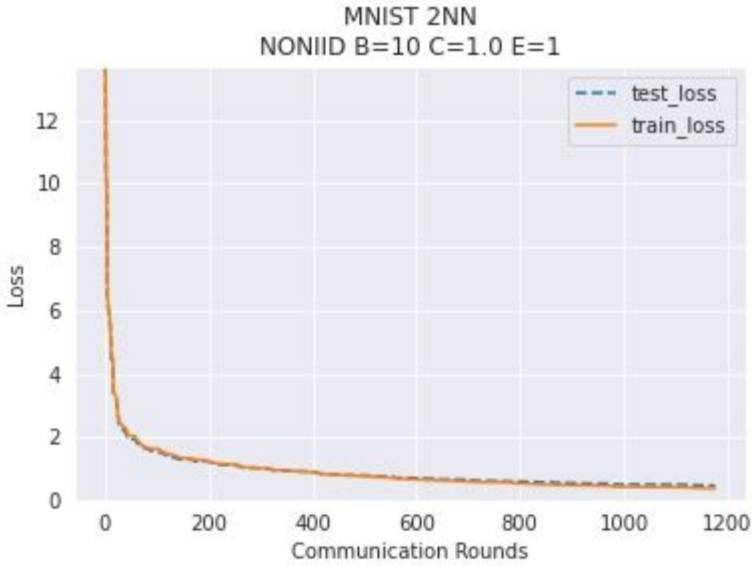


# Experiment Result - Task 2

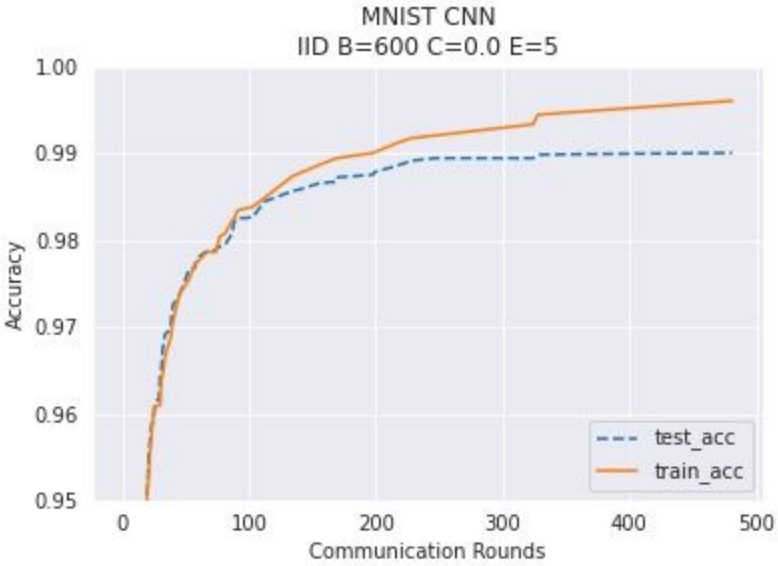
- **2NN, NONIID, B=10, C=1.0, E=1 (lr=0.1)**

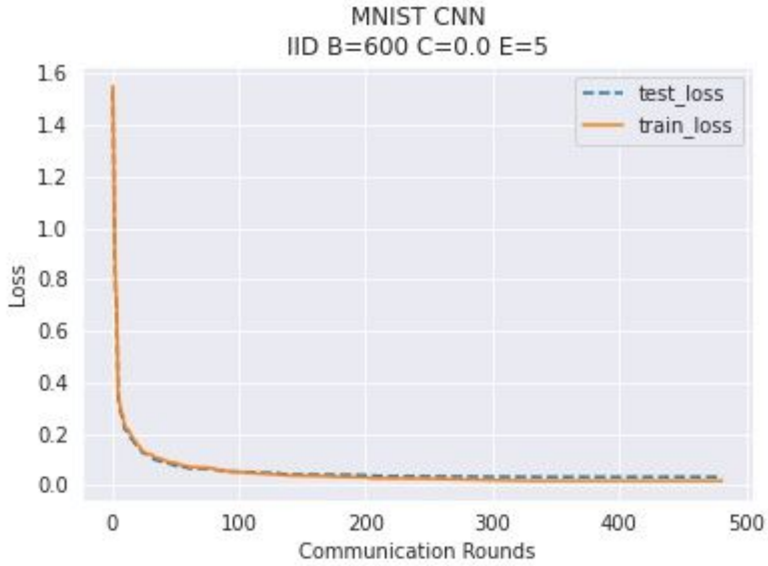


# Experiment Result - Task 2

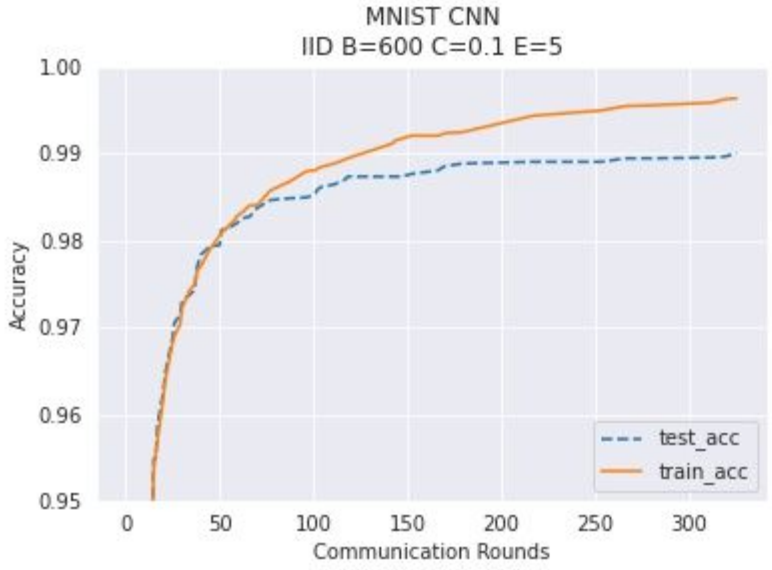
- **CNN, IID, B=600, C=0.0, E=5 (lr=0.215)**

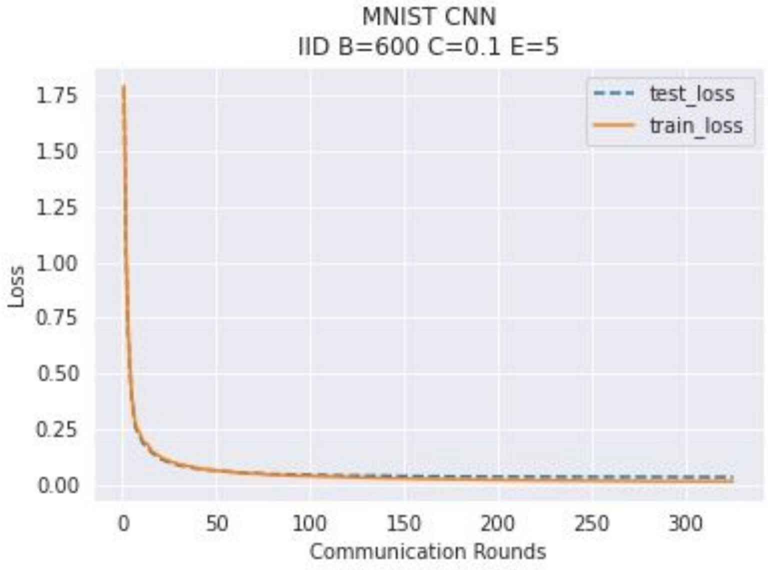


# Experiment Result - Task 2

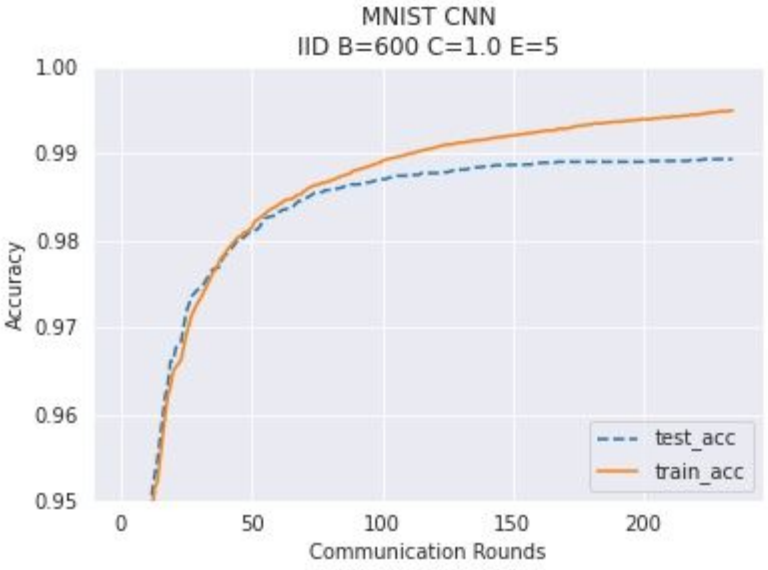
- **CNN, IID, B=600, C=0.1, E=5 (lr=0.215)**

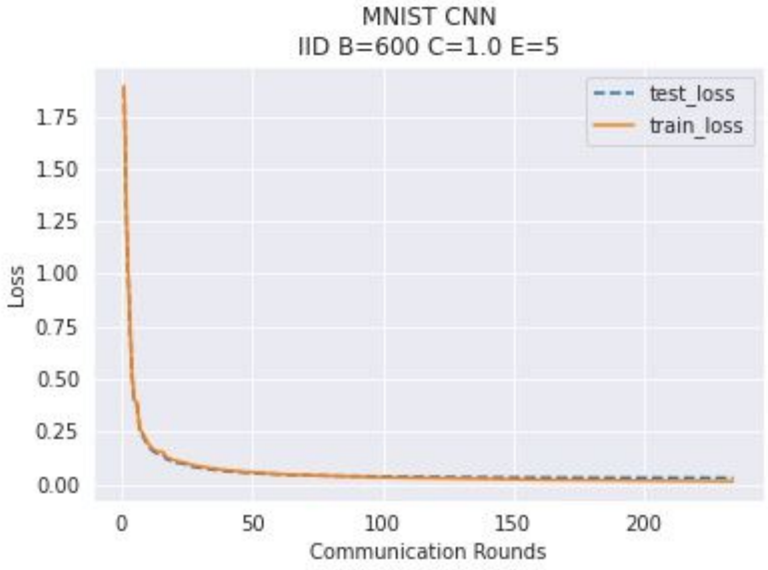


# Experiment Result - Task 2

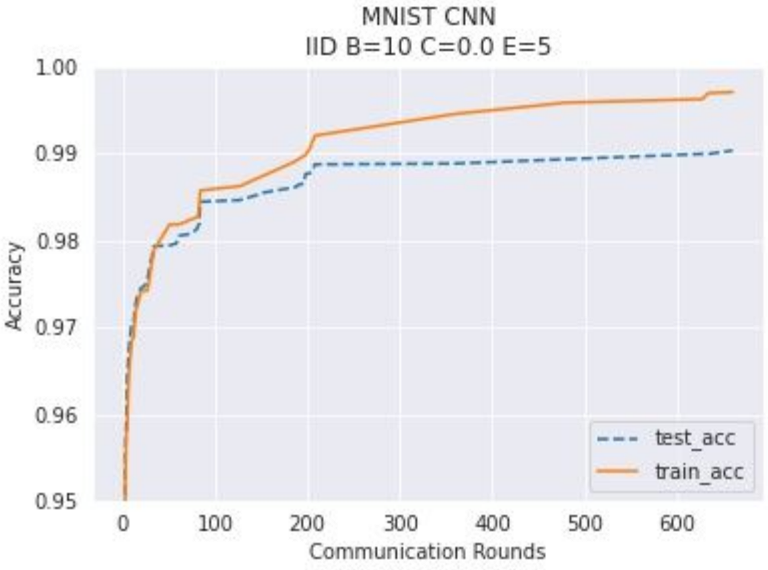
- **CNN, IID, B=600, C=1.0, E=5 (lr=0.215)**

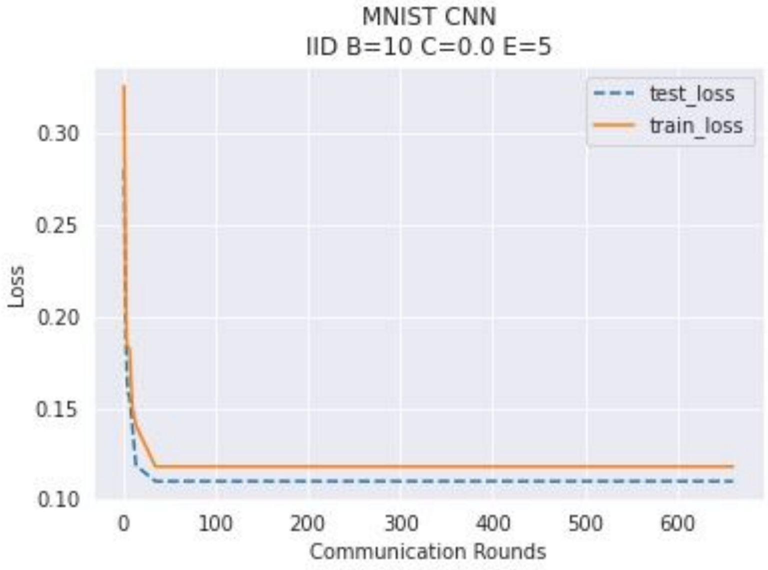


# Experiment Result - Task 2

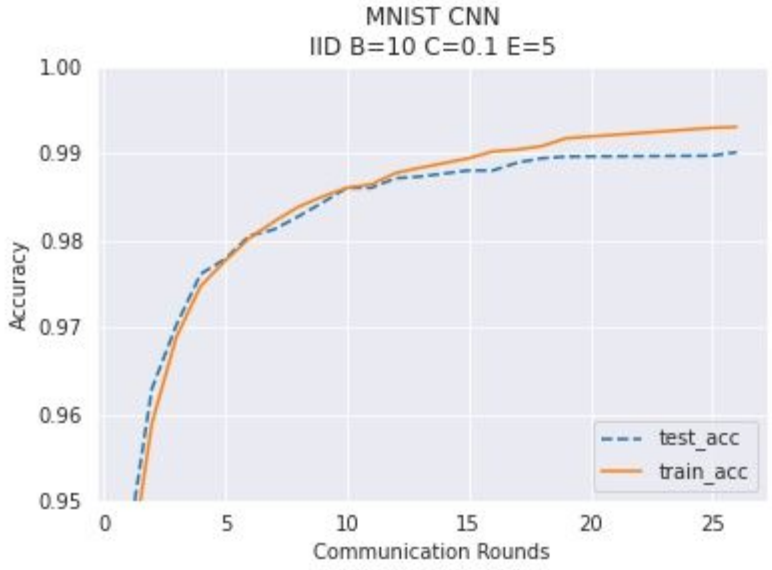
- **CNN, IID, B=10, C=0.0, E=5 (lr=0.215)**

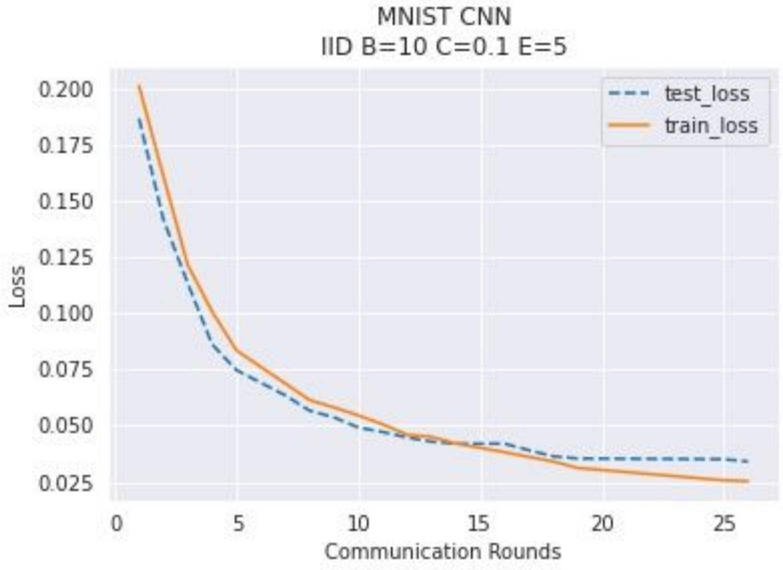


# Experiment Result - Task 2

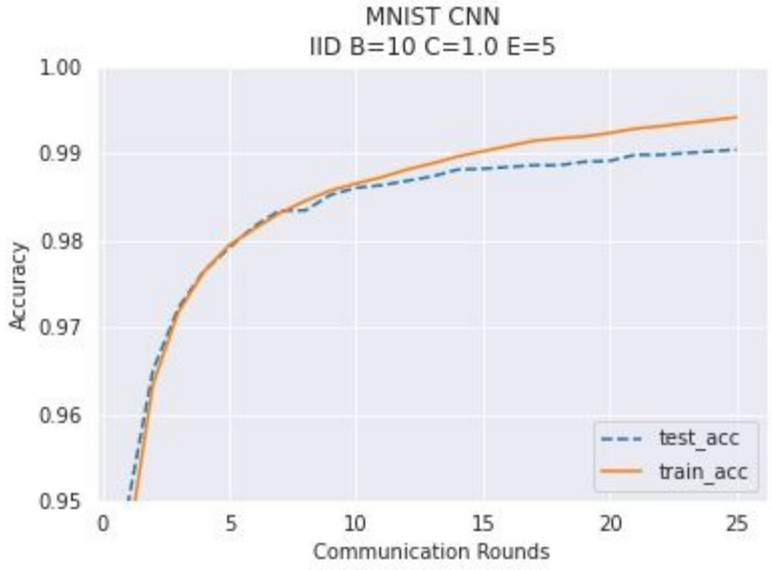
- **CNN, IID, B=10, C=0.1, E=5 (lr=0.215)**

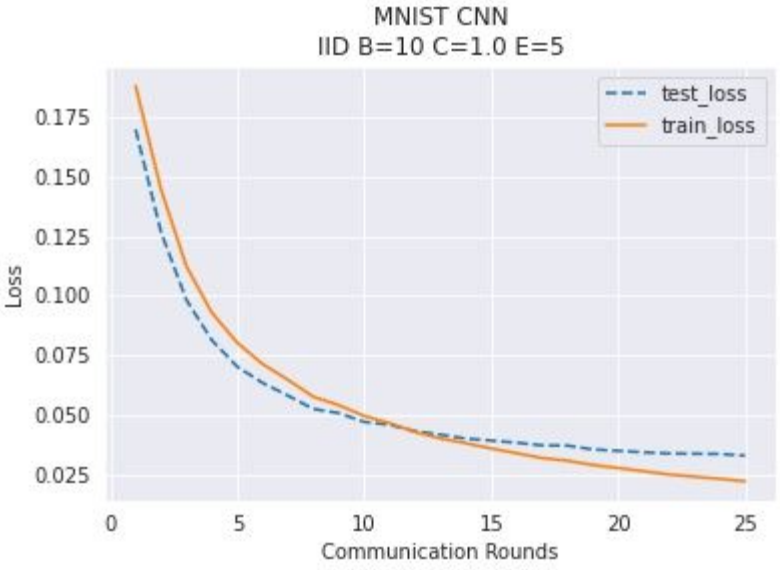


# Experiment Result - Task 2

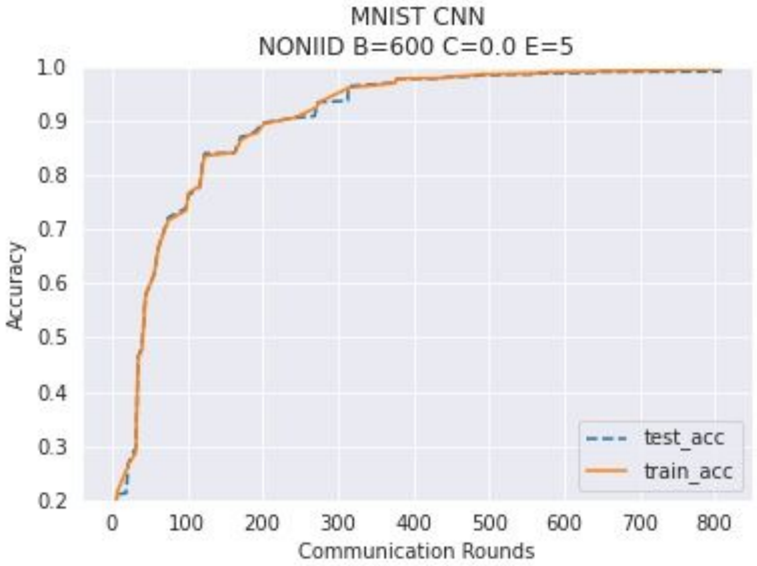
- **CNN, IID, B=10, C=1.0, E=5 (lr=0.215)**

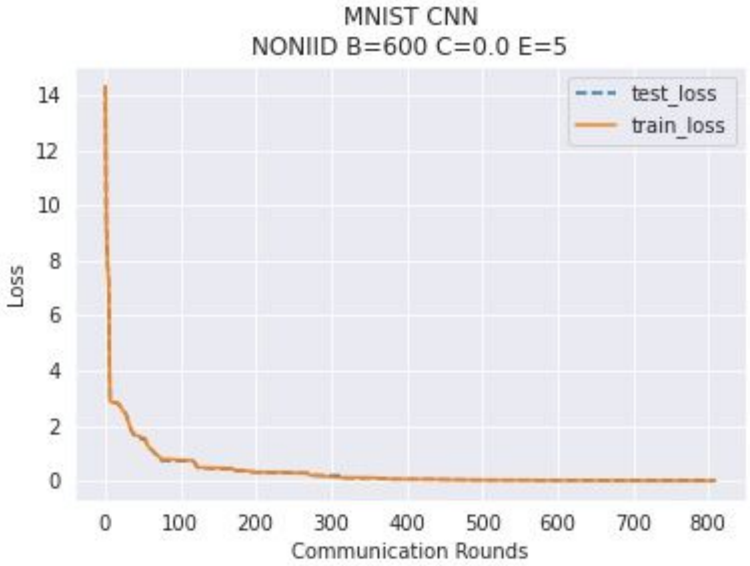


# Experiment Result - Task 2

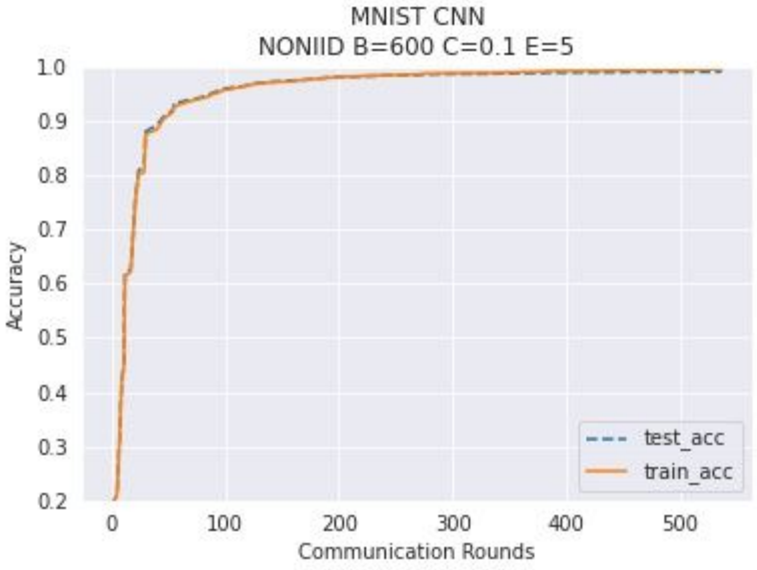
- **CNN, NONIID, B=600, C=0.0, E=5 (lr=0.215)**

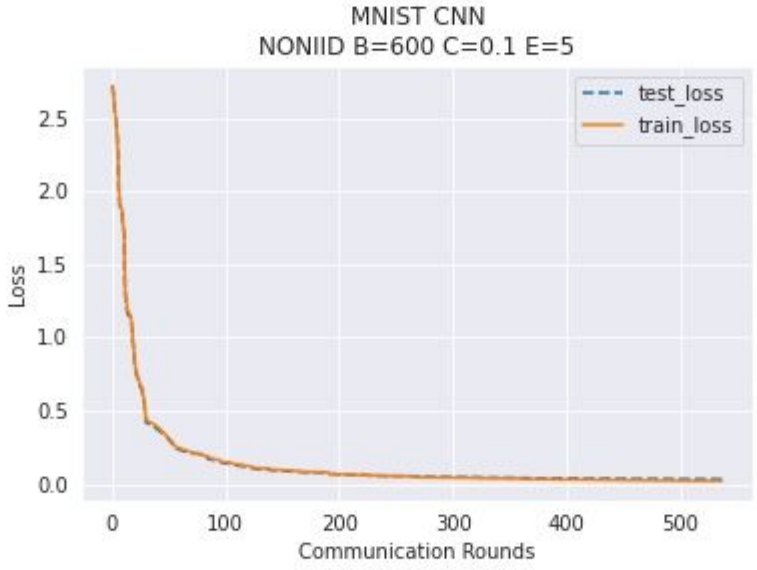


# Experiment Result - Task 2

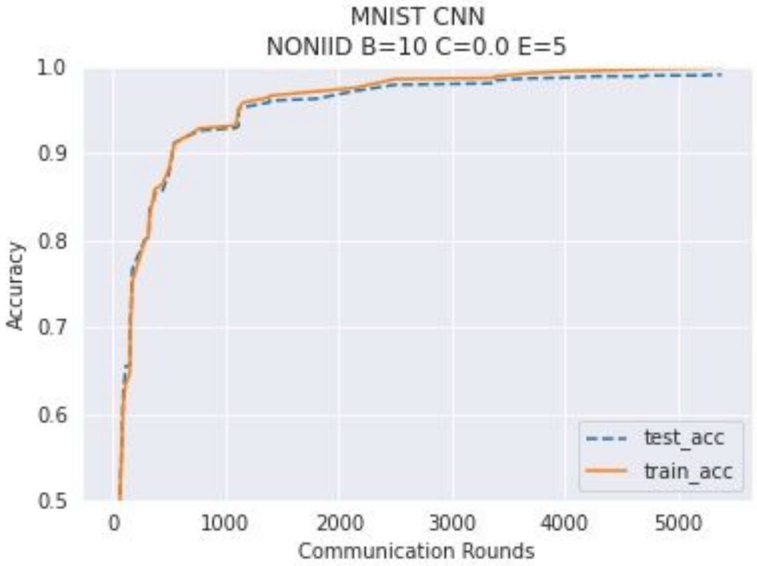
- **CNN, NONIID, B=600, C=0.1, E=5 (lr=0.215)**

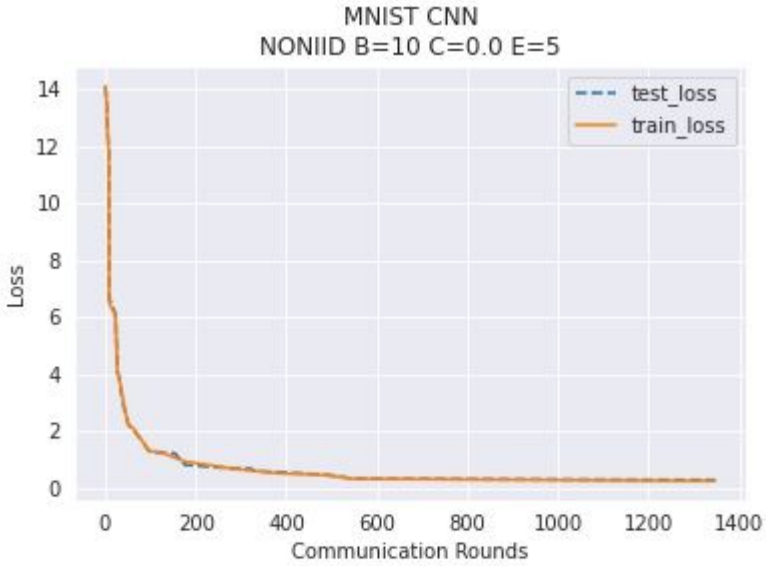


# Experiment Result - Task 2

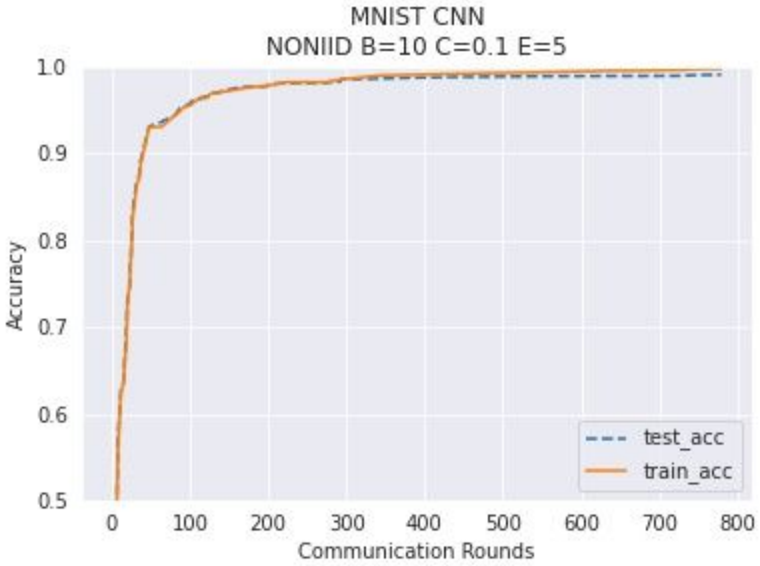
- **CNN, NONIID, B=10, C=0.0, E=5 (lr=0.215)**

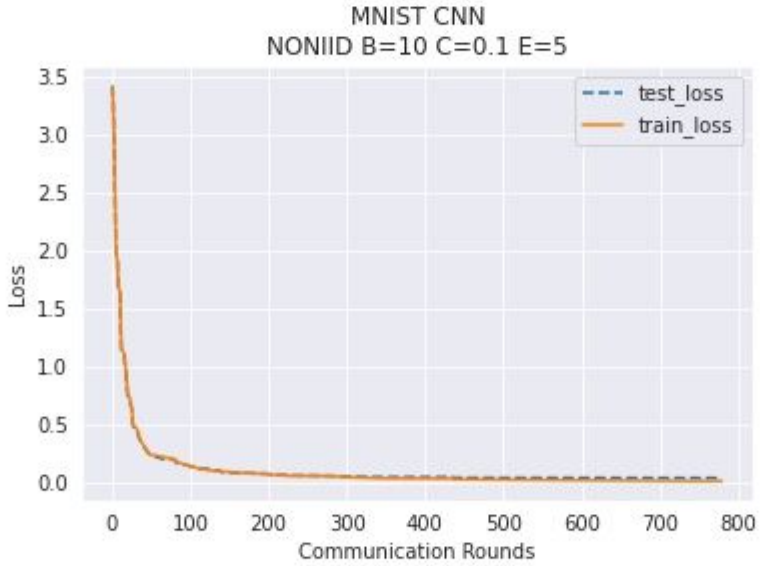


# Experiment Result - Task 2

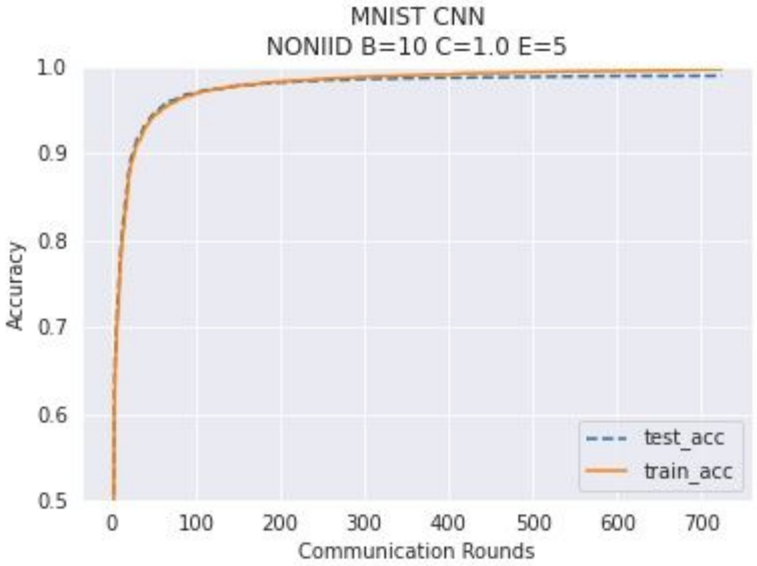
- **CNN, NONIID, B=10, C=0.1, E=5 (lr=0.215)**

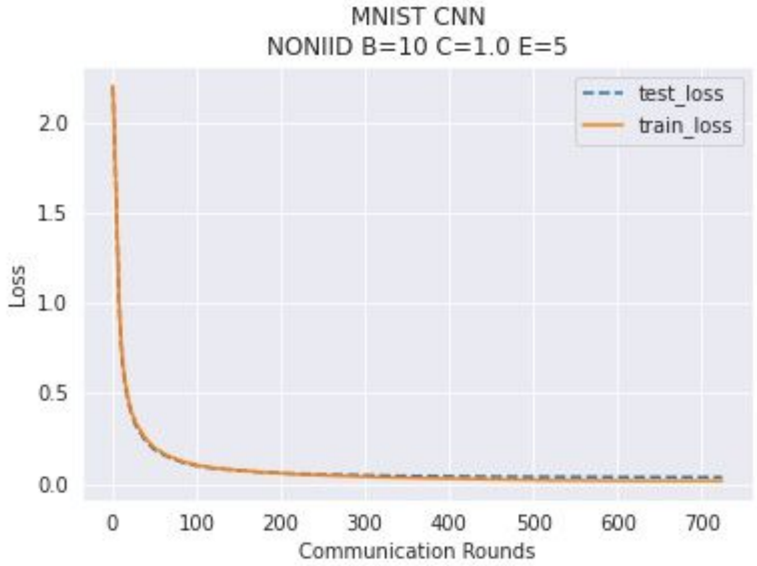


# Experiment Result - Task 2

- **CNN, NONIID, B=10, C=1.0, E=5 (lr=0.215)**



# Experiment Result - Task 2

- **Reproduce Table [1] : Comparison**

**Table [1]: Original**

| 2NN | IID | | NONIID | |
|---|---|---|---|---|
| C | B=600 | B=10 | B=600 | B=10 |
| 0.0 | 1455 | 316 | 4278 | 3275 |
| 0.1 | 1474 (1.0x) | 87 (3.6x) | 1796 (2.4x) | 664 (4.9x) |
| 1.0 | (—) (—) | 70 (4.5x) | (—) (—) | 380 (8.6x) |
| CNN, E=5 | | | | |
| 0.0 | 387 | 50 | 1181 | 956 |
| 0.1 | 339 (1.1x) | 18 (2.8x) | 1100 (1.1x) | 206 (4.6x) |
| 1.0 | 246 (1.6x) | 16 (3.1x) | (—) (—) | 97 (9.9x) |

**Table [1]: Reproduce**

| 2NN | IID | | NONIID | |
|---|---|---|---|---|
| C | B=600 | B=10 | B=600 | B=10 |
| 0.0 | 2449 | 127 | 2620 | 2764 |
| 0.1 | 2526 (1.0x) | 53 (2.4x) | 2176 (1.2x) | 1138 (2.4x) |
| 1.0 | (—) (—) | 45 (2.8x) | (—) (—) | 1176 (2.4x) |
| CNN, E=5 | | | | |
| 0.0 | 481 | 660 | 809 | 5380 |
| 0.1 | 325 (1.5x) | 26 (25x) | 536 (1.5x) | 779 (6.9x) |
| 1.0 | 234* (2.1x) | 25 (25x) | (—) (—) | 724* (7.4x) |

# Experiment Result - Task 2

- **Key Takeaways: 2NN**
  - Identical to the original experiment, **smaller batch size produces significant improvement** both in IID and non-IID case.
  - With B=10, **a speed up of 2.4x-2.8x** was recorded in both IID and non-IID case.

# Experiment Result - Task 2

- **Key Takeaways: CNN**
  - Identical to the experiment, in both IID and non-IID cases, **smaller batch size** led to a **better** speed up.
  - **Increasing parallelism** in all cases, did **speed up the communication rounds** needed.
  - Cases that mark with asterisk (*) failed to converge to exact value 0.99 in this experiment. Both achieved a maximum test accuracy of 0.9893, 0.9889 respectively.

# Experiment Result - Task 2

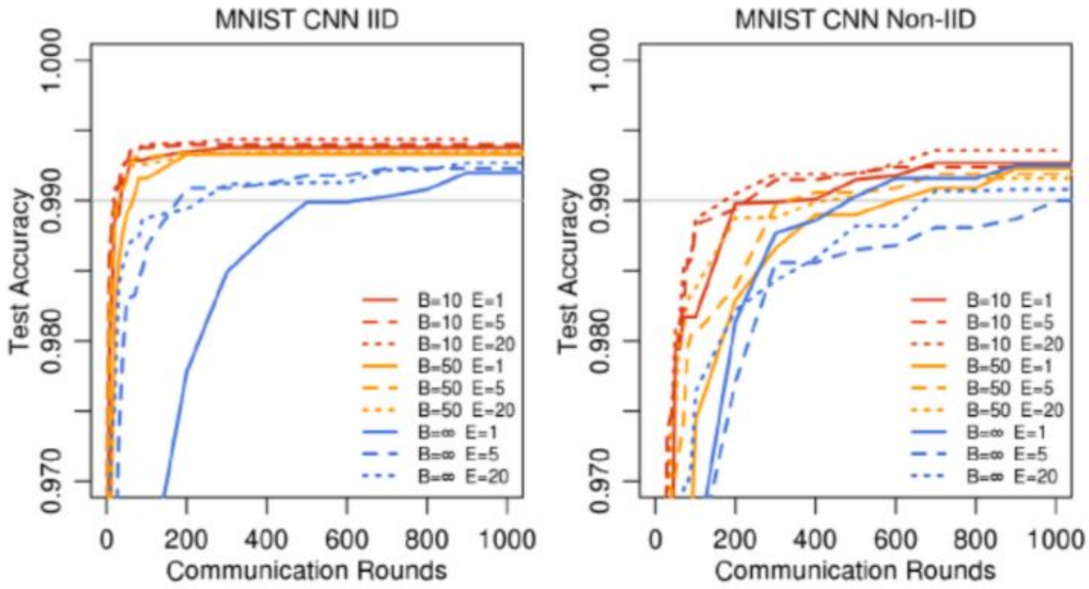
- **Key Takeaways: Discrepancies**
  - **Some cases failed to converged.**
    - Some cases **with C=1.0 failed to converge** to a test accuracy of 0.99.
    - Training using all data from all clients with a high learning rate could cause an **overfitting** or cause the model to converge to a **suboptimal solution**.
    - Yet, the achieved test accuracy was still decent at **0.9893** and **0.9889**.

# Experiment Result - Task 2

- **Key Takeaways: Discrepancies**
  - **Using B=10 in CNN, produce quite different results from the original experiment.**
    - Major difference observed when the **C is set to 0.0**, communication rounds tend to be much larger than the original experiment.
    - This probably due to the **gradient norm clipping** that I applied to prevent exploding gradient problem  when using a high learning rate. This cause the gradient value to scale down and causing the **rate of convergence slower**.
    - Putting aside the differences, the experiment still shows that **increasing parallelism did speed up** the communication rounds needed.

# Experiment Result - Task 3

3) Reproduce the following figures for $(B,E) = (10,1),(10,20),(50,1),(50,20),(\infty,1),(\infty,20)$.




- Do not use the validation set.

# Experiment Result - Task 3

- **Reproduce Figure [2]: CNN with Various (B, E)**
  - This experiment done to show the effect of **increasing computation** to the number of communication rounds.
  - Dataset used for this experiment is [MNIST Digit Recognition]() from Tensorflow. **Train data** contains **60,000 images** and **test data** contains **10,000 images**.
  - Threshold used for stopping the training process of the CNN is **1000** rounds.
  - A learning rate of **0.215** for both IID and NON-IID was used.

# Experiment Result - Task 3

- **Reproduce Figure [2]: CNN with Various (B, E)**
  - The training graph presented is **plotted monotonically improving**, done by taking the best test accuracy among the obtained values from previous communication rounds.
  - **B** is the local minibatch size, **E** is the number of local epochs, and **u** is the number of expected updates per round.
  - Metrics used:
    - Accuracy: AccuracyScore*
    - Loss: SparseCategoricalCrossentropy

*calculated by comparing the ground truth label and index of the maximum probability output by each prediction

# Experiment Result - Task 3

- **Reproduce Figure [2]: Comparison**

**Figure [2]: Original**

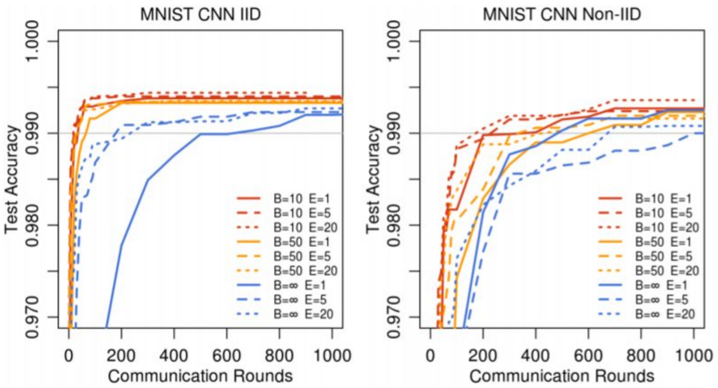
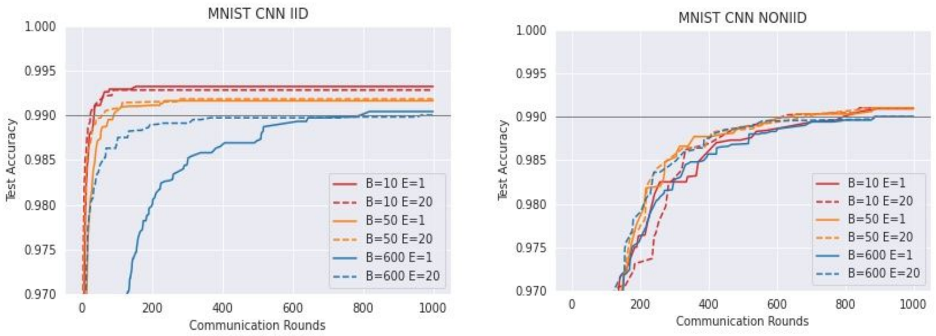



**Figure [2]: Reproduce**



# Experiment Result - Task 3

- **Reproduce Table [2]: Comparison**

**Table [2]: Original**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | **MNIST CNN,** 99% Accuracy | | | | |
| **CNN** | **E** | **B** | **u** | **IID** | | **NON-IID** | |
| FEDSGD | 1 | 600 | 1 | 626 | | 483 | |
| FEDAVG | 1 | 50 | 12 | 65 | (9.6x) | 600 | (0.8x) |
| FEDAVG | 20 | 600 | 20 | 234 | (2.7x) | 672 | (0.7x) |
| FEDAVG | 1 | 10 | 60 | 34 | (18.4x) | 350 | (1.4x) |
| FEDAVG | 20 | 50 | 240 | 32 | (19.6x) | 426 | (1.1x) |
| FEDAVG | 20 | 10 | 1200 | 18 | (34.8x) | 173 | (2.8x) |

**Table [2]: Reproduce**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | **MNIST CNN,** 99% Accuracy | | | | |
| **CNN** | **E** | **B** | **u** | **IID** | | **NON-IID** | |
| FEDSGD | 1 | 600 | 1 | 793 | | 885 | |
| FEDAVG | 1 | 50 | 12 | 92 | (8.6x) | 658 | (1.3x) |
| FEDAVG | 20 | 600 | 20 | 965 | (0.8x) | 890 | (1.0x) |
| FEDAVG | 1 | 10 | 60 | 35 | (22.7x) | 795 | (1.1x) |
| FEDAVG | 20 | 50 | 240 | 58 | (13.7x) | 721 | (1.2x) |
| FEDAVG | 20 | 10 | 1200 | 25 | (31.7x) | 632 | (1.4x) |

# Experiment Result - Task 3

- **Key Takeaways:**
  - Identical to the paper results, **increasing the number of local updates** proven to **decrease the amount of communication rounds** needed both in IID and non-IID cases.
  - The decrease could be as large as **22.7x** and **31.7x** in IID setting yet smaller in non-IID setting with only a maximum **1.4x** decrease.

# References (1)

Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V., Mao, M. Z., Ranzato, M. A., Senior, A., Tucker, P., Yang, K., & Ng, A. Y. (2012). Large scale distributed deep networks. *Advances in Neural Information Processing Systems*, *2*, 1223–1231.

McDonald, R., Hall, K., & Mann, G. (2010). Distributed training strategies for the structured perceptron. *NAACL HLT 2010 - Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference.*

Neverova, N., Wolf, C., Lacey, G., Fridman, L., Chandra, D., Barbello, B., & Taylor, G. (2016). Learning Human Identity from Motion Patterns. *IEEE Access*, *4*, 1810–1820. https://doi.org/10.1109/ACCESS.2016.2557846

# References (2)


Povey, D., Zhang, X., & Khudanpur, S. (2015). Parallel training of DNNs with natural gradient and parameter averaging. *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, 1–28.

Shokri, R., & Shmatikov, V. (2016). Privacy-preserving deep learning. *2015 53rd Annual Allerton Conference on Communication, Control, and Computing, Allerton 2015*, 909–910. https://doi.org/10.1109/ALLERTON.2015.7447103

Zhang, S., Choromanska, A., & Lecun, Y. (2015). Deep learning with elastic averaging SGD. *Advances in Neural Information Processing Systems, 2015-January*, 685–693.

# Thank You!

# Appendix

- **Source code:** https://drive.google.com/file/d/1a_lYLuqKpLiS_8u-v4wYXozcl4lzlb14/view?usp=sharing
  - Readme File
  - Training Logs
  - Training Plots
  - Runner Template
- **Reference paper**: Communication-Efficient Learning of Deep Networks from Decentralized Data