University of Tehran

Faculty of Engineering

School of Electrical and Computer Engineering

Distributed Optimization and Learning

Project 2

Gossip Training for Image Classification with Deep Learning

Hafez Ghaemi

810199239

Winter 2022

Abstract

In this project, we will consider the problem of gossip training of distributed deep neural networks for image classification. Gossip learning is a fully-decentralized alternative to federated learning (FL), in which the central controller that is in charge of weight aggregation in FL is removed, and the agents perform a variant of random gossip algorithms to reach consensus and convergence. We apply gossip learning to image classification with convolutional neural networks with different graph models (weight matrices), and compare the results to the ones obtained using centralized and federated learning.

1. Introduction

Although federated learning (FL) solvers some of the problems in centralized training of machine learning (ML) and deep learning (DL), it still faces some challenges in areas such as privacy, data protection, and scalability. Even when the agents share only a gradient with a controller who is in charge of aggregating them and building a global model [3], the privacy concerns regarding user data still remains [6]. Furthermore, maintaining a global server for aggregating models with proper communication channels may be costly.

Therefore, a fully distributed alternative to federated learning, in which the users (agents) communicate directly with each other over a graph is favorable [2]. Specifically for distributed deep learning, gossip training has been proposed [1]. The proposed GoSGD algorithm is based on the gossip protocol, and therefore completely asynchronous [4]. There is no central controller in the setup and during the algorithm, and only at the end, the agents may share their model weights to be aggregated to build a global model. The network architecture is the same for all agents so that the agents are able to share their weights with their neighbors for local aggregation.

We will train all the models on the Fashion MNIST [4] benchmark for different types of weight graphs, noisy links, and both iid and non-iid data distributions. The network used is a convolutional neural network that is typically used for image classification. In section 2, we will discuss the Fashion MNIST dataset. Section 3 lays out the network architecture. The GoSGD algorithm is explained in Section 4, and the evaluation results (for centralized, federated, and gossip learning) are given in Section 5. Finally, Section 6 offers a brief discussion around and comparison of the results and concludes this report.

2. Dataset

The dataset we used for training and evaluation is Fashion MNIST [4], which consists of 60000 training samples alongside a test set with 10000 samples. The dataset consists of ten classes of clothing items and it is completely balanced in terms of class distribution. Figure. 1 shows a sample of each class belonging to this dataset.

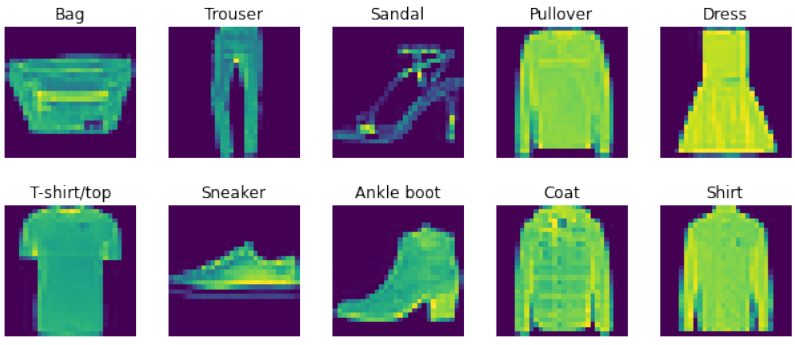


Figure 1. Class samples of the Fashion MNIST dataset

For iid splits, we shuffle the dataset and randomly sample the required fraction of each dataset. For non-iid splits, we split each class into shards, and for each split randomly select data shards from only two random classes.

3. Network Architecture

We use the convolutional neural network architecture as proposed by McMahan et al. [3] in federated learning paper for all agents in the graph.

Layer	Dimension	Properties
Input	(B, 1,28,28)	
Convolution	(B, 32,26,26)	Padding = 1, Stride = 1 Kernel Size = 5, Activation = ReLU
Max Pooling	(B, 32,14,14)	Padding = 1, Stride = 1, Kernel Size = 2
Convolution	(B, 64, 12, 12)	Padding = 1, Stride = 1 Kernel Size = 5, Activation = ReLU
Max Pooling	(B, 64,7,7)	Padding = 1, Stride = 1, Kernel Size = 2
Flatten	(B, 3136)	
Dense	(B, 512)	Activation = ReLU
Dense	(B, 10)	Activation = Softmax

The total number of trainable parameters in the model is 1,663,370. We set batch size to 128 and the number of training epochs to 20 for centralized training, and 10 for the local models in federated learning. The optimization algorithm used in all models is stochastic gradient descent with a learning rate of 0.05 and momentum equal to 0.9.

4. Methodology

In this section, we will describe the GoSGD algorithm, and the gossip update functions used in the gossip training process. The pseudocode of the algorithm and the functions are given below,

Algorithm 1 GoSGD: workers Pseudo-code

Input: p: probability of exchange, M: number of threads, η : learning rate
Initialize: x is initialized randomly, $x_i = x$, $\alpha_i = \frac{1}{M}$
repeat
  PROCESSMESSAGES(msg$_i$)
  $x_i \leftarrow x_i - \eta^t v_i^t$
  if $S \sim B(p)$ **then**
    $j = Random(M)$
    PUSHMESSAGE(msg$_j$)
  end if
until Maximum iteration reached
return $\frac{1}{M}\sum_{m=1}^{M} x_m$

Algorithm 2 Gossip update functions

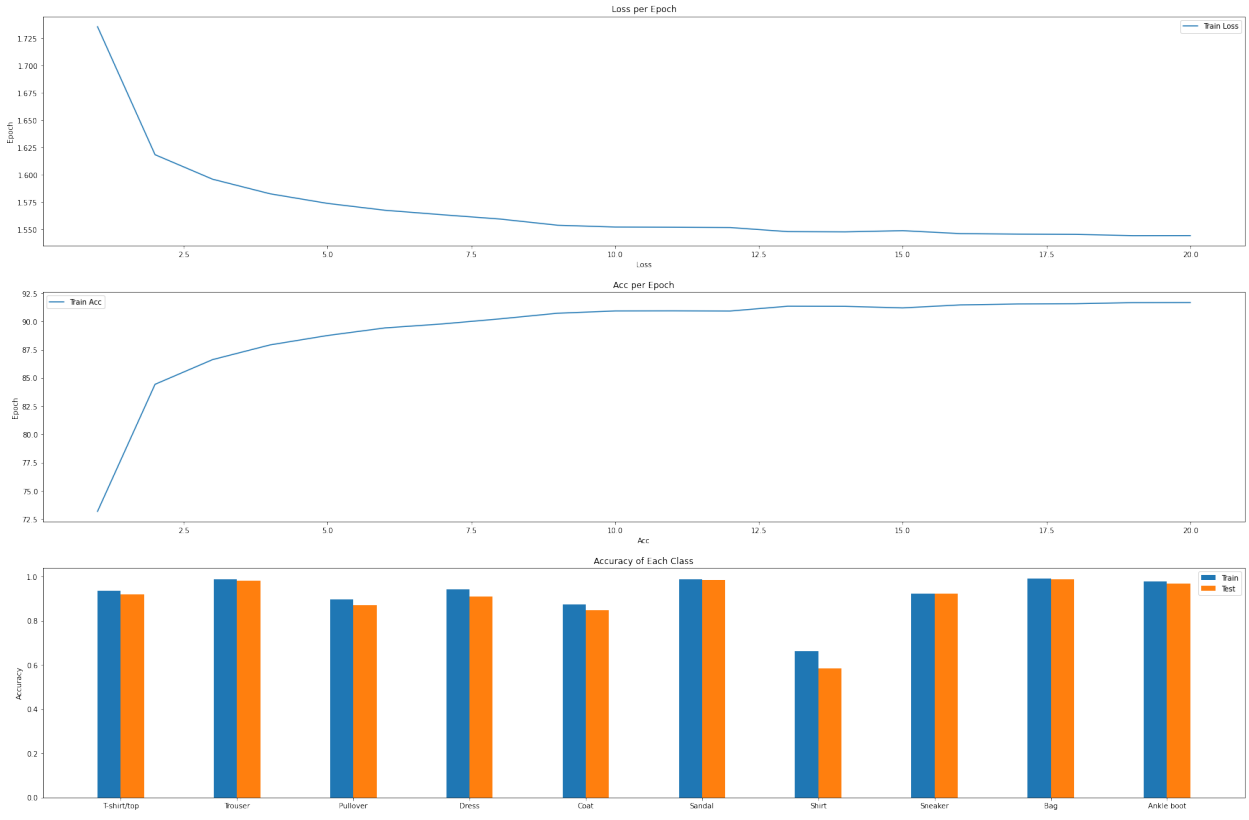
function PUSHMESSAGE(queue msg$_j$)
  $x_i \leftarrow x_i$
  $\alpha_i \leftarrow \frac{\alpha_i}{2}$
  msg$_j$.push((x_i, α_i))
end function
function PROCESSMESSAGES(queue msg$_i$)
  repeat
    $(x_j, \alpha_j) \leftarrow$ msg$_i$.pop()
    $x_i \leftarrow \frac{\alpha_j}{\alpha_i+\alpha_j} x_j + \frac{\alpha_i}{\alpha_i+\alpha_j} x_i$
    $\alpha_i \leftarrow \alpha_j + \alpha_i$
  until msg$_i$.empty()
end function

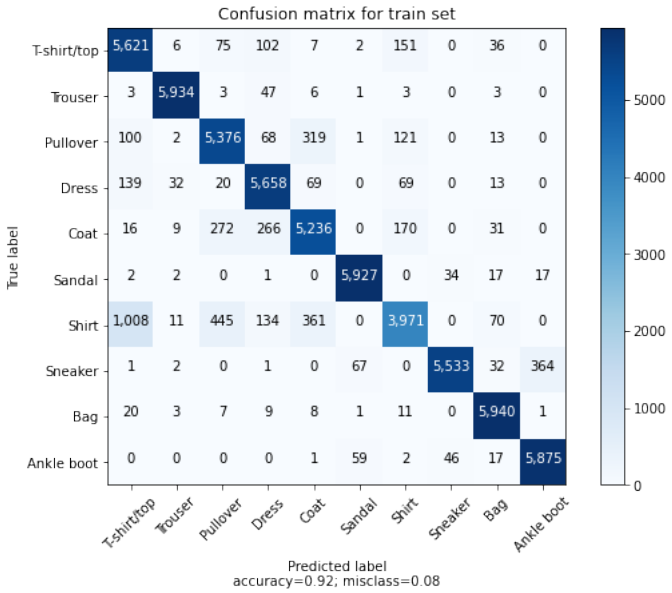
The algorithm consists of a loop in which each agent first processes the asynchronous messages received from her neighbors and aggregates the received weights with its own based on the link weights. Afterwards, she performs gradient descent on a given batch in its dataset. Finally, with a probability p, she chooses a random neighbor and sends her a message consisting of its model weights and the weight of the connection between her and the neighbor. At the end, if needed a global aggregator may aggregate the model weights to build a global model.

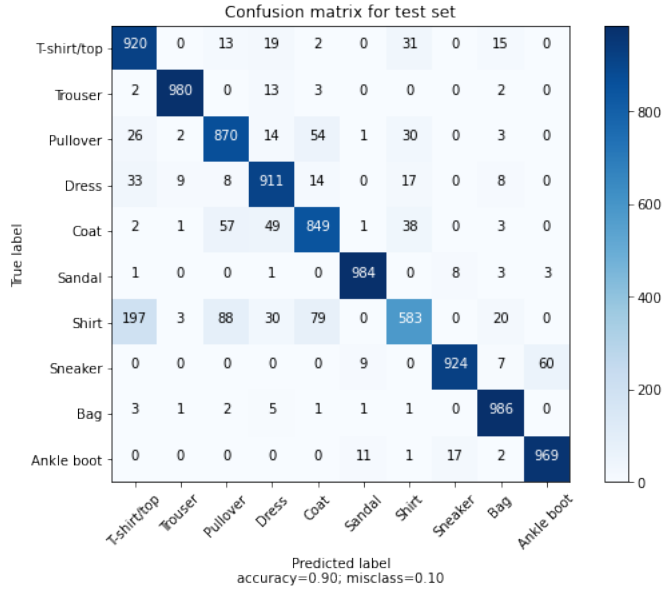
5. Results

5.1. Centralized Training

For centralized training, the hyperparameters are the ons mentioned in Section 3. Batch size is set to 128 and the number of training epochs is set to 20. The optimization algorithm used is stochastic gradient descent with a learning rate of 0.05 and momentum equal to 0.9. The results are given below,

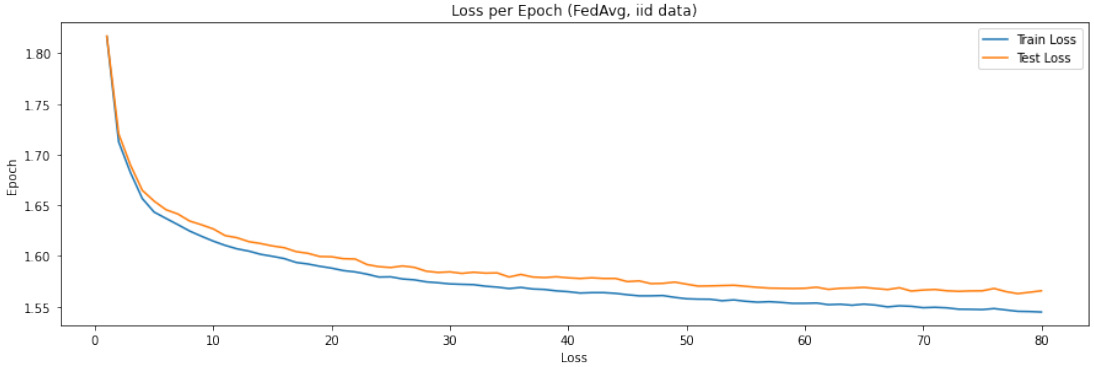


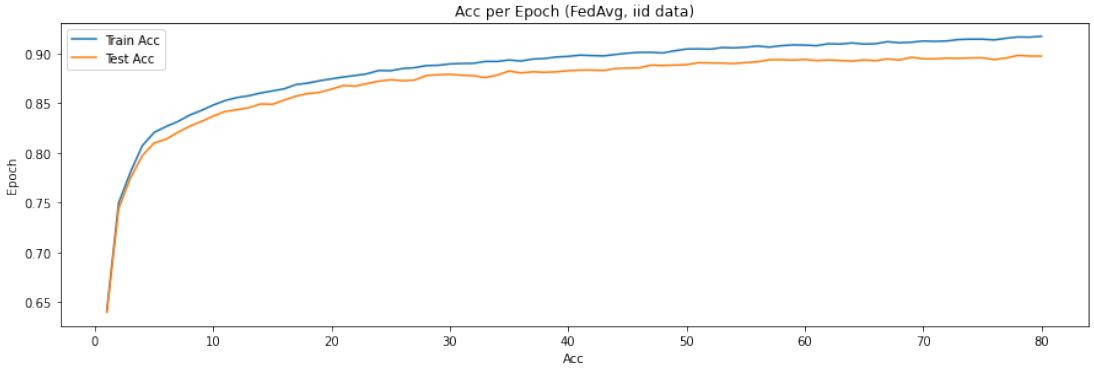


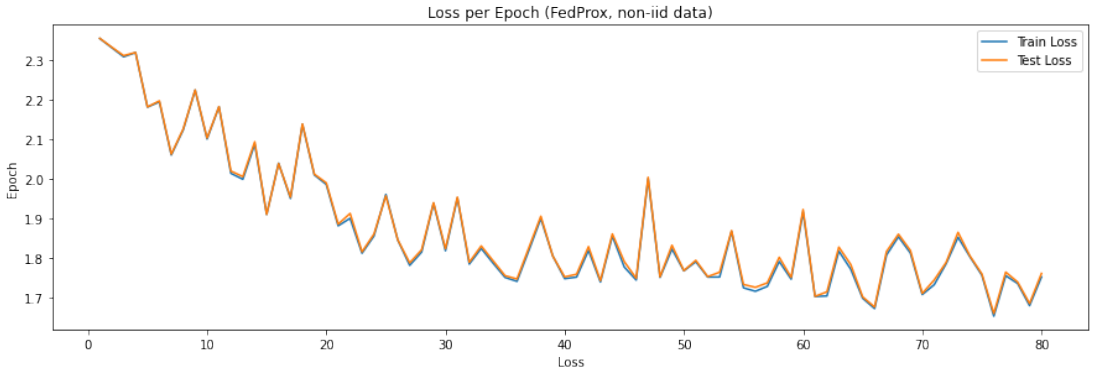


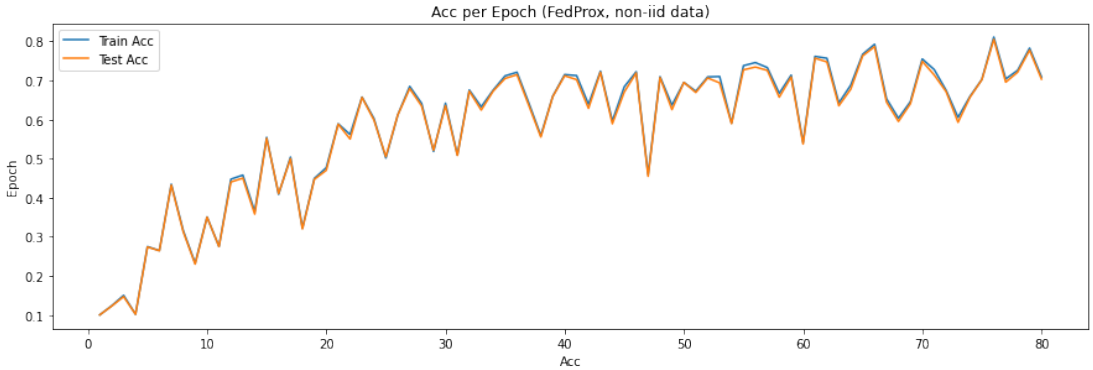
5.2. Federated Learning

Here we will provide our results obtained previously on iid and non-iid Fashion MNIST dataset to be able to compare them with the results of gossip learning [7]. The best iid results are obtained using FedAvg [3], and the best non-iid results are obtained using FedProx [8]. Apart from the common hyperparameters mentioned in Section 3, the number of rounds (E) was set to 80, and the fraction of clients chosen at each round (k) was set to 0.1 for FedAvg and FedProx, and the regularization parameter μ for FedProx was set to 3. The train and test loss accuracy for the global model at each round are plotted below for these two algorithms.


Loss per Epoch (FedAvg, iid data)
Train Loss
Test Loss
Epoch
Loss


Acc per Epoch (FedAvg, iid data)
Train Acc
Test Acc
Epoch
Acc

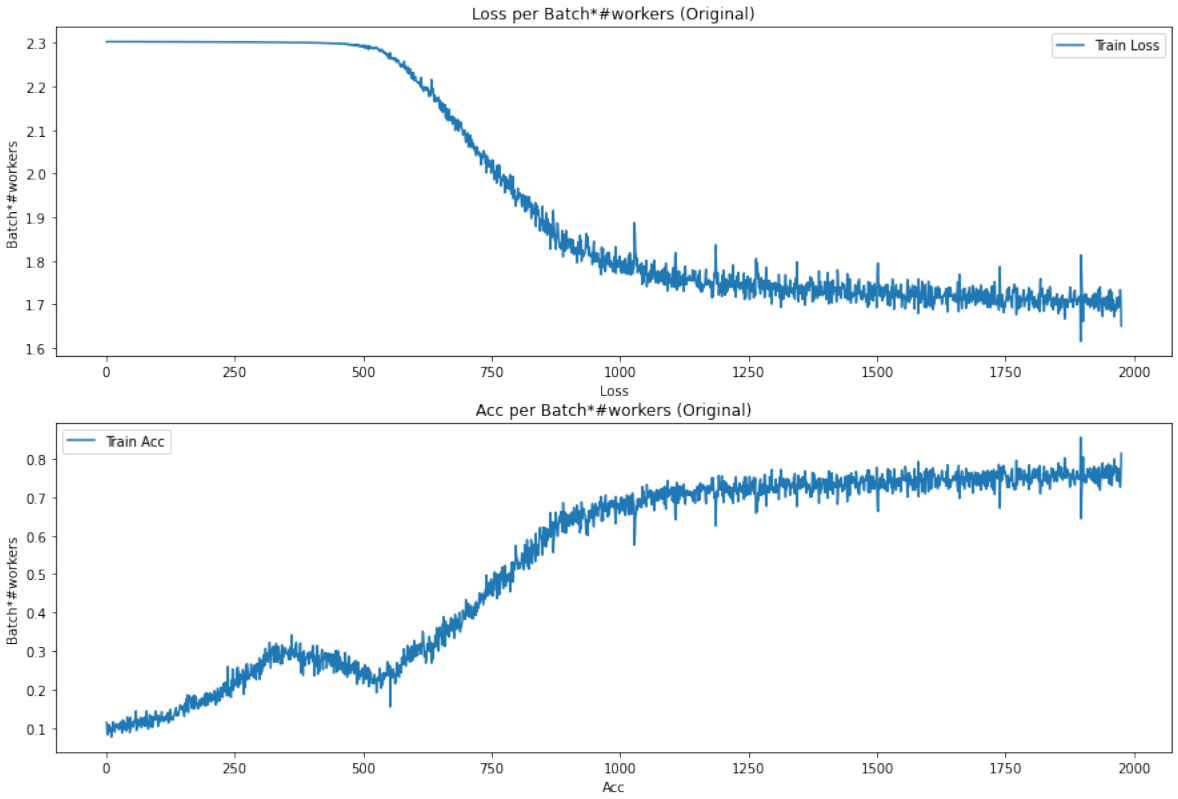

Loss per Epoch (FedProx, non-iid data)
Train Loss
Test Loss
Epoch
Loss

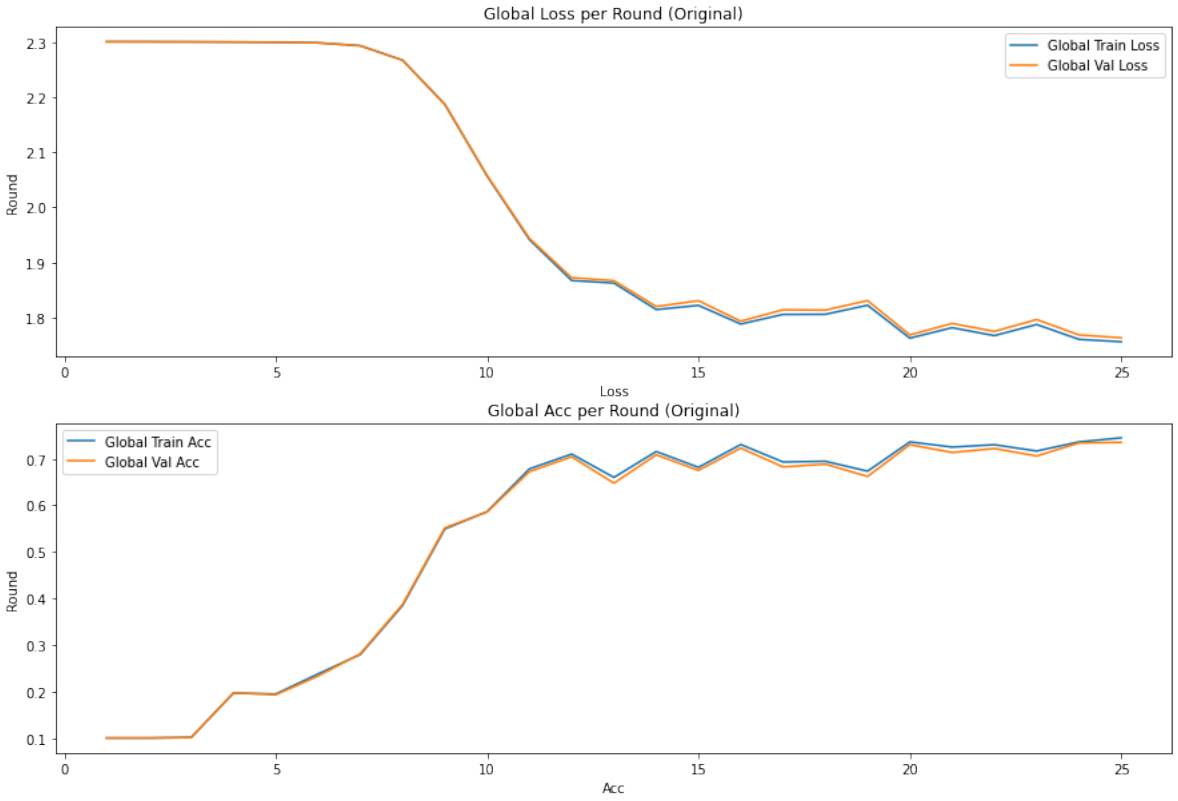

Acc per Epoch (FedProx, non-iid data)
Train Acc
Test Acc
Epoch
Acc

5.3. Gossip Training

Original Gossip Training with iid Data Distribution

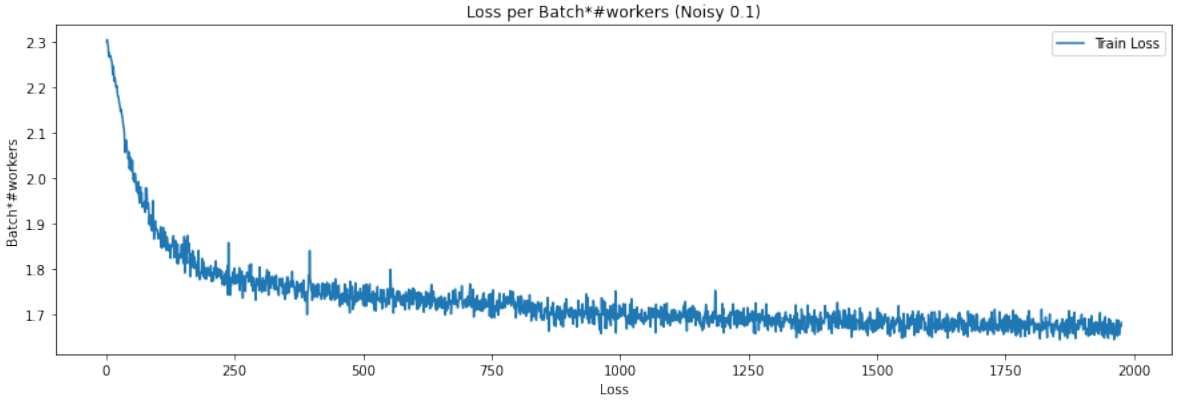
We first implemented the original gossip training algorithm [1] using a varying stochastic weight matrix (see Algorithm 2) which is initialized as a doubly stochastic matrix (see Algorithm 1). The data distribution is iid, and we set the number of agents in the graph to six. The algorithm is performed for 25 rounds (during each round we go over the number of batches in the local dataset for each worker). The network architecture and the model hyperparameters are given in section 3. The average loss and accuracy per worker for each batch along with the global loss and accuracy of the aggregate model for each round are plotted below.

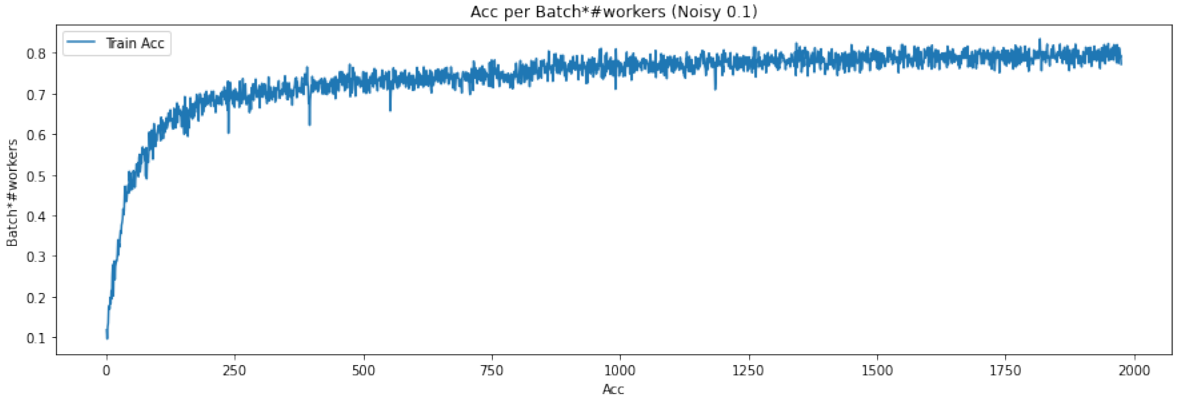


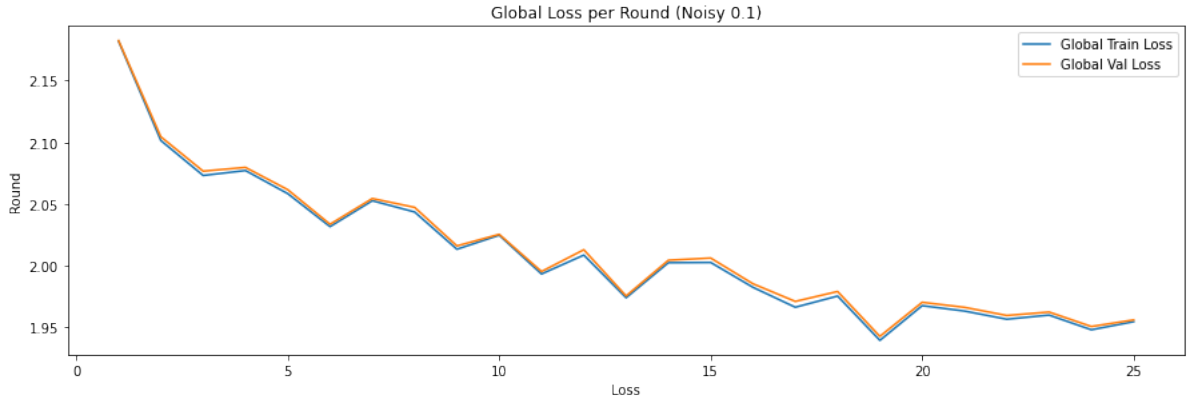


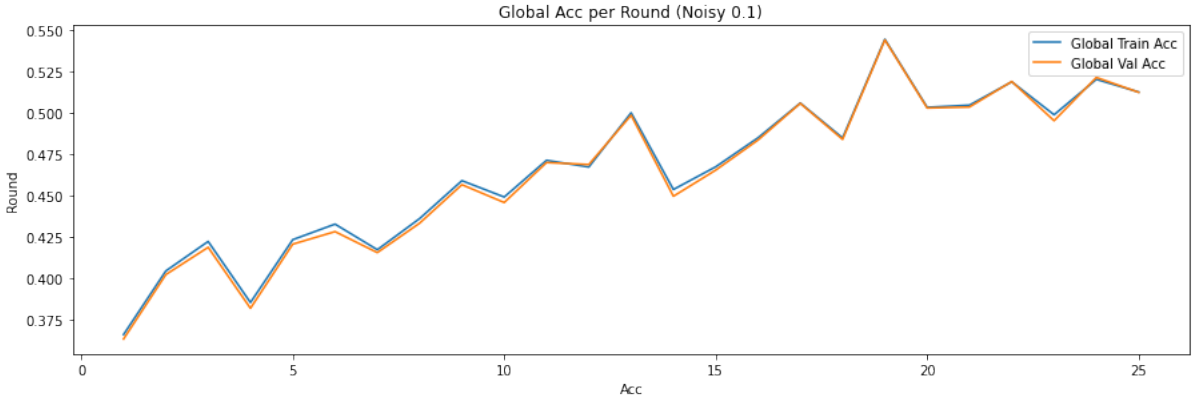
Gossip Training with Noisy Links (Noise = 0.1)

We added a normal random communication noise multiplied by 0.1 when the agents are communicating their weights in the original gossip learning algorithm. The data distribution is iid, and we set the number of agents in the graph to six. The algorithm is performed for 25 rounds (during each round we go over the number of batches in the local dataset for each worker). The network architecture and the model hyperparameters are given in section 3. The average loss and accuracy per worker for each batch along with the global loss and accuracy of the aggregate model for each round are plotted below.


Loss per Batch*#workers (Noisy 0.1)
Train Loss
Batch*#workers
Loss

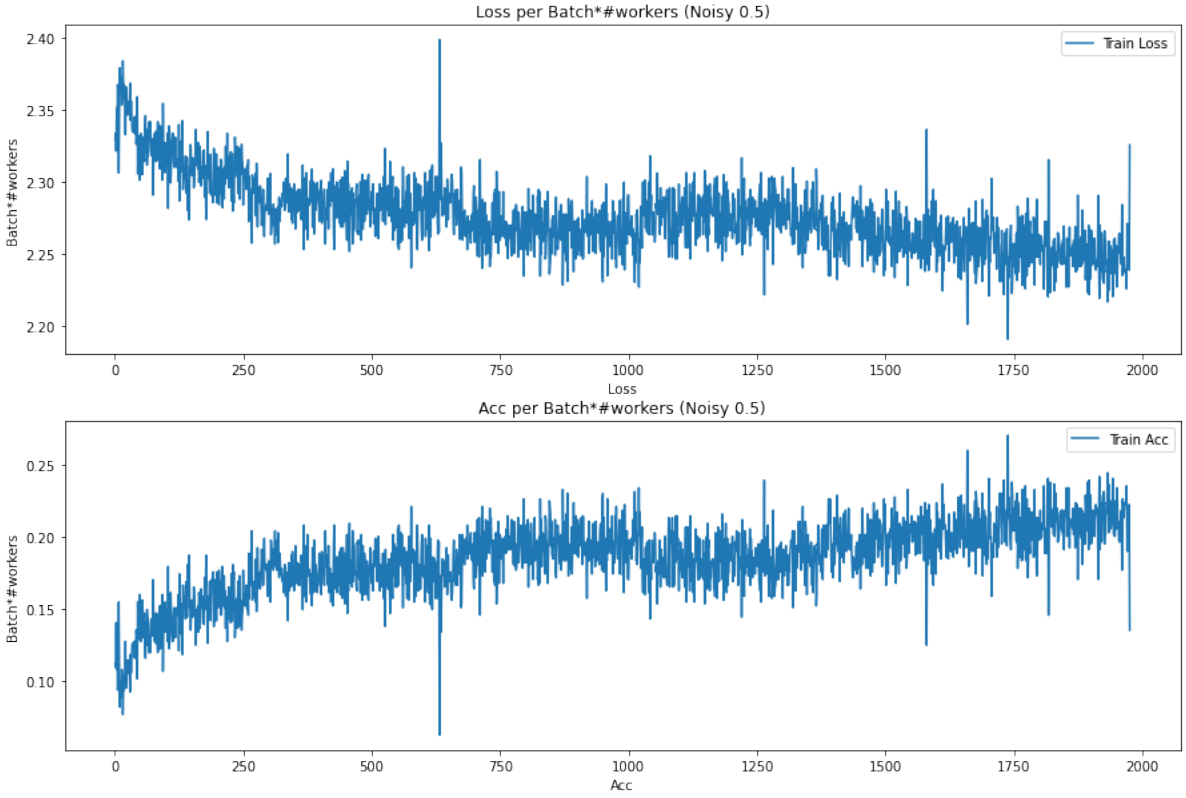

Acc per Batch*#workers (Noisy 0.1)
Train Acc
Batch*#workers
Acc

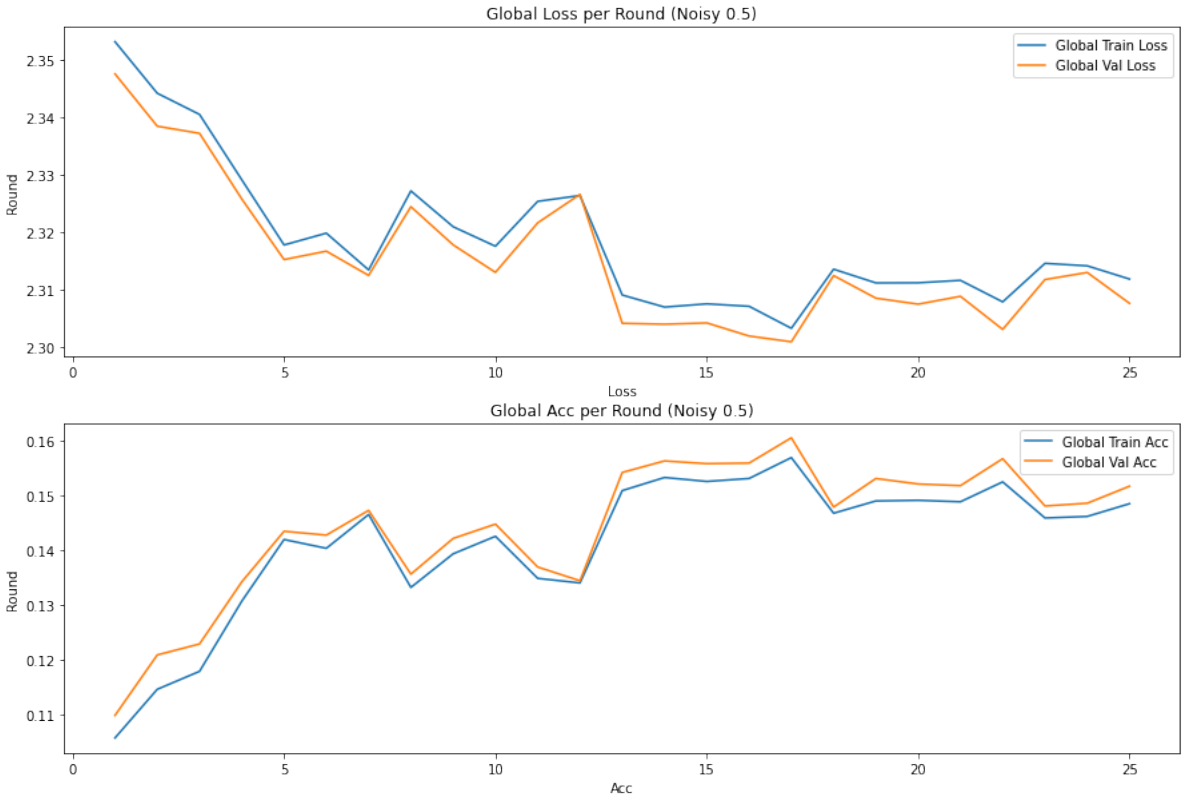

Global Loss per Round (Noisy 0.1)
Global Train Loss
Global Val Loss
Round
Loss


Global Acc per Round (Noisy 0.1)
Global Train Acc
Global Val Acc
Round
Acc

Gossip Training with Noisy Links (Noise = 0.5)

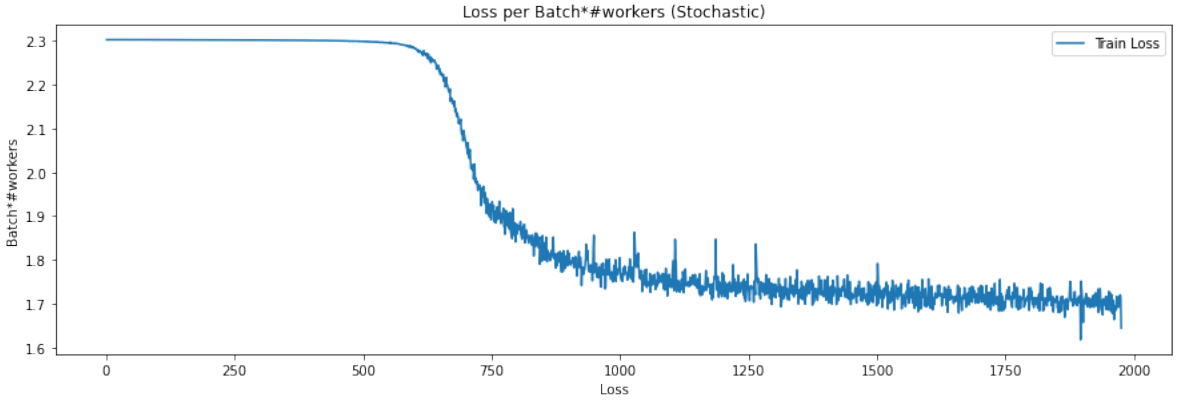
We added a normal random communication noise multiplied by 0.5 when the agents are communicating their weights in the original gossip learning algorithm. The data distribution is iid, and we set the number of agents in the graph to six. The algorithm is performed for 25 rounds (during each round we go over the number of batches in the local dataset for each worker). The network architecture and the model hyperparameters are given in section 3. The average loss and accuracy per worker for each batch along with the global loss and accuracy of the aggregate model for each round are plotted below.

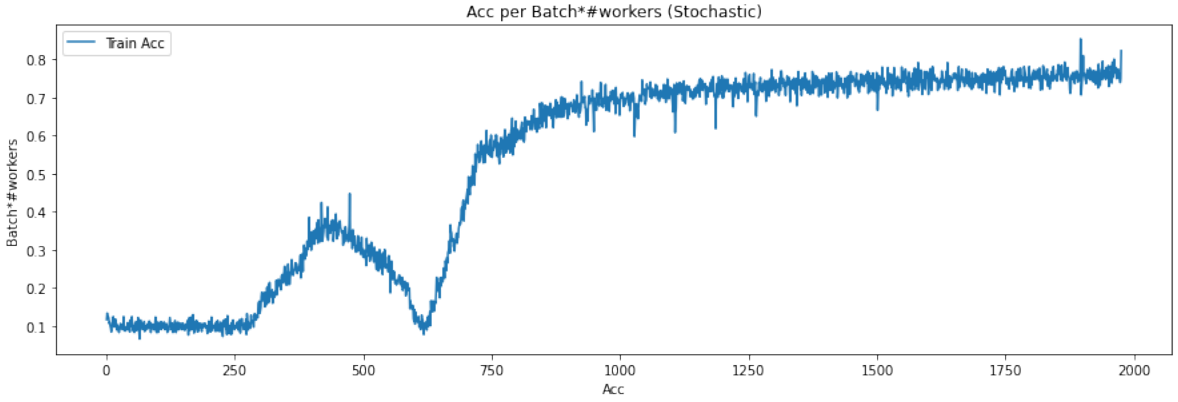


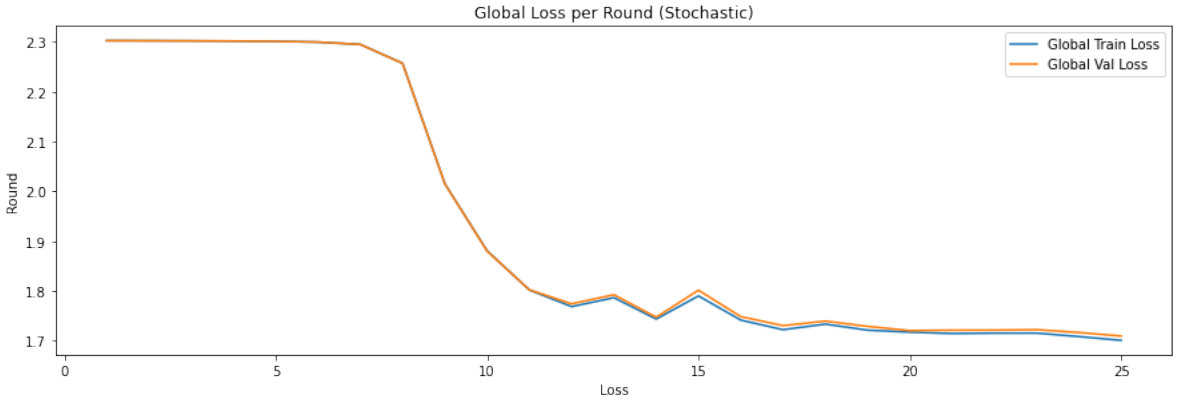


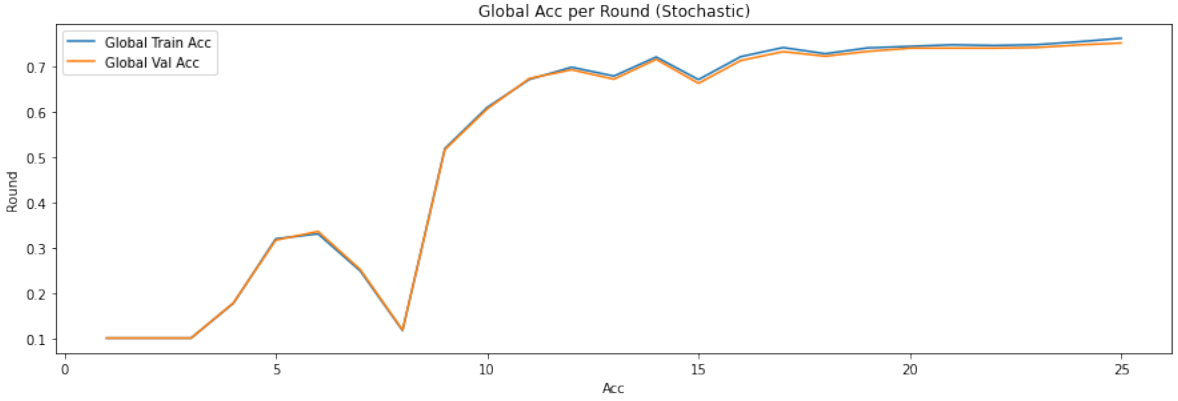
Gossip Training with a Constant Stochastic Weight Matrix

We used a constant stochastic weight matrix instead of the varying matrix in the original gossip learning algorithm. The data distribution is iid, and we set the number of agents in the graph to six. The algorithm is performed for 25 rounds (during each round we go over the number of batches in the local dataset for each worker). The network architecture and the model hyperparameters are given in section 3. The average loss and accuracy per worker for each batch along with the global loss and accuracy of the aggregate model for each round are plotted below.


Loss per Batch*#workers (Stochastic)
Train Loss
Batch*#workers
Loss

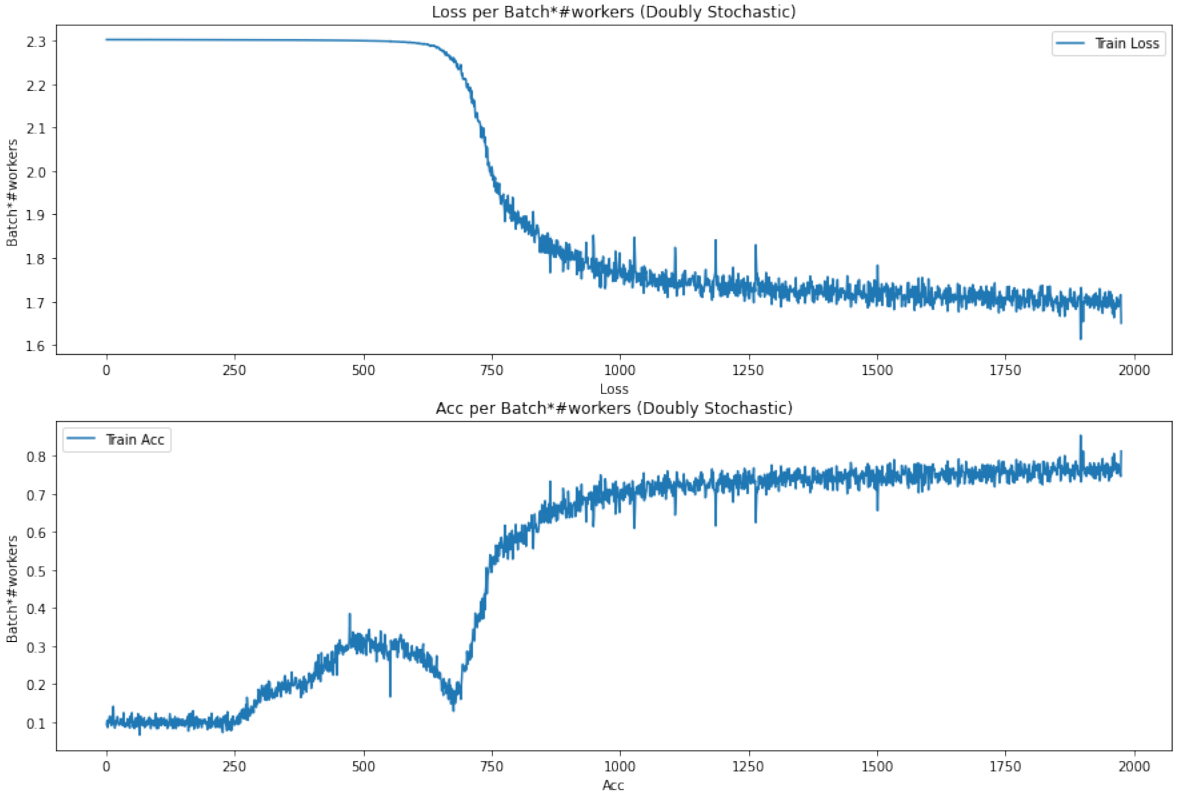

Acc per Batch*#workers (Stochastic)
Train Acc
Batch*#workers
Acc

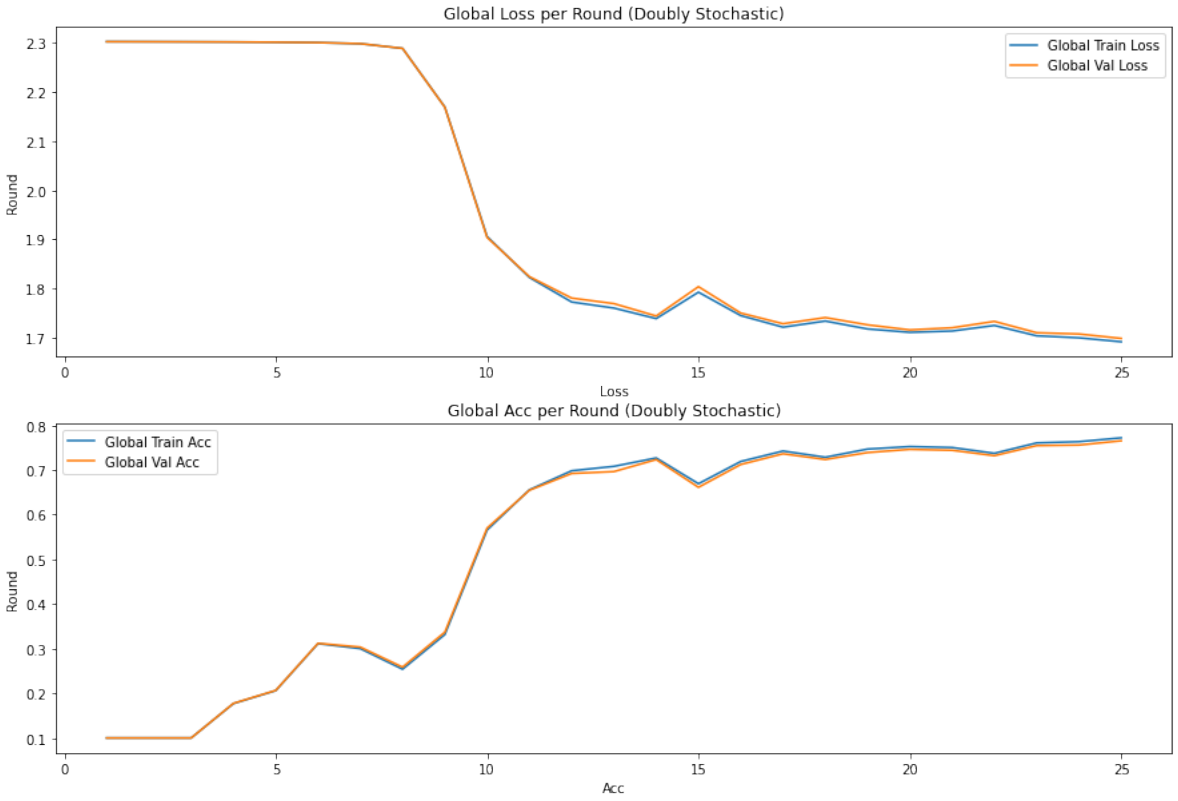

Global Loss per Round (Stochastic)
Global Train Loss
Global Val Loss
Round
Loss


Global Acc per Round (Stochastic)
Global Train Acc
Global Val Acc
Round
Acc

Gossip Training with a Constant Doubly Stochastic Weight Matrix

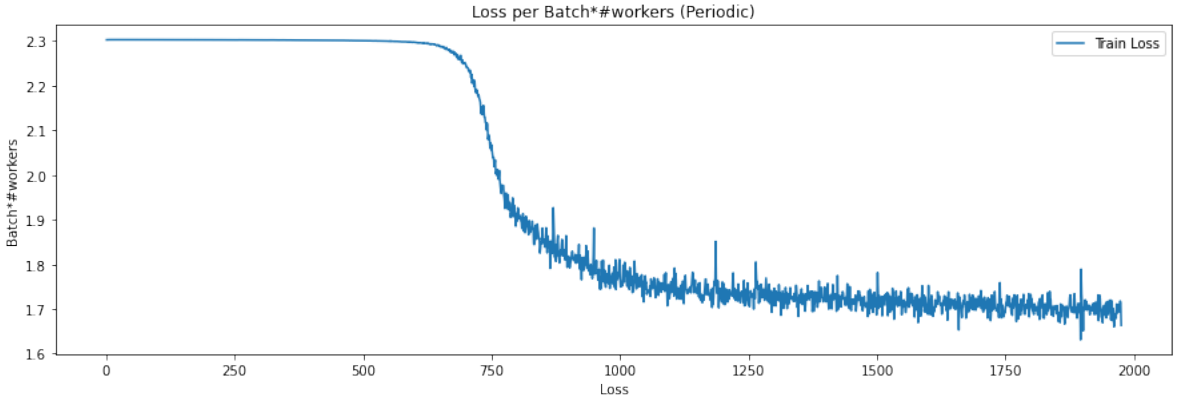
We used a constant doubly stochastic weight matrix instead of the varying matrix in the original gossip learning algorithm. The data distribution is iid, and we set the number of agents in the graph to six. The algorithm is performed for 25 rounds (during each round we go over the number of batches in the local dataset for each worker). The network architecture and the model hyperparameters are given in section 3. The average loss and accuracy per worker for each batch along with the global loss and accuracy of the aggregate model for each round are plotted below.

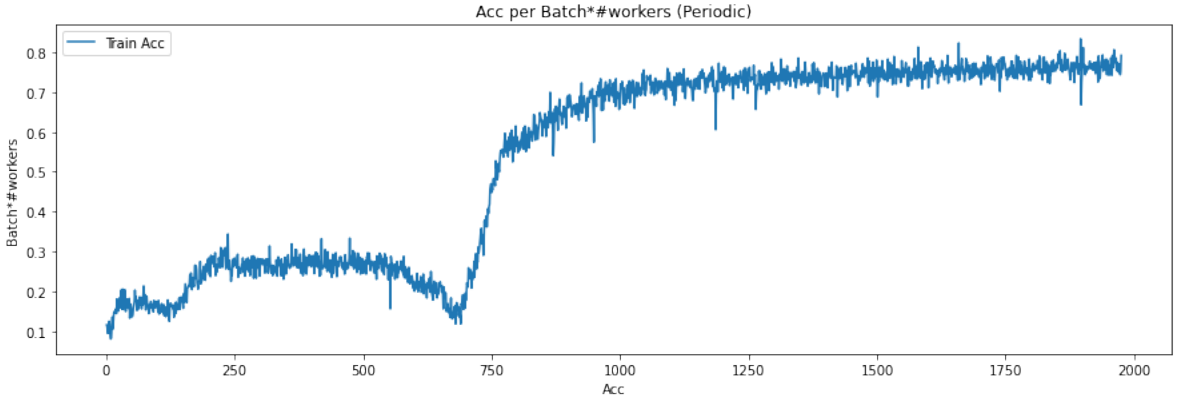


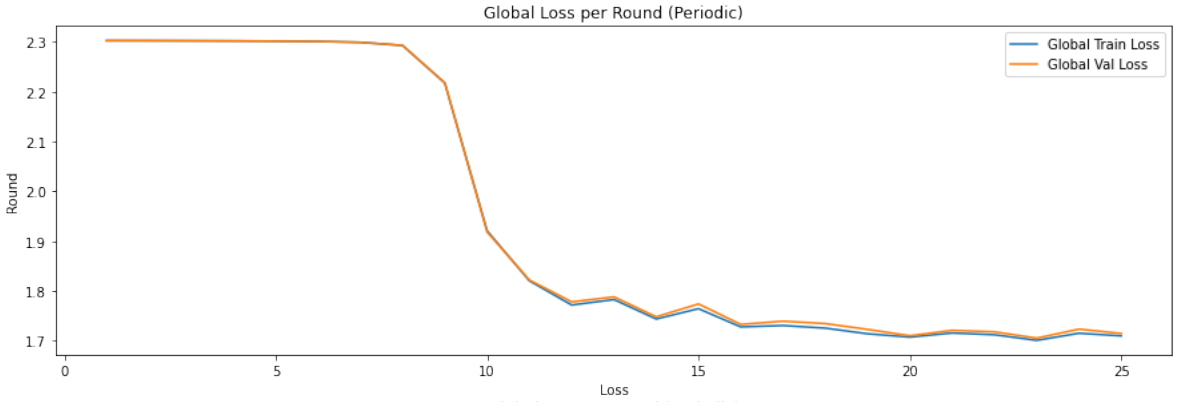


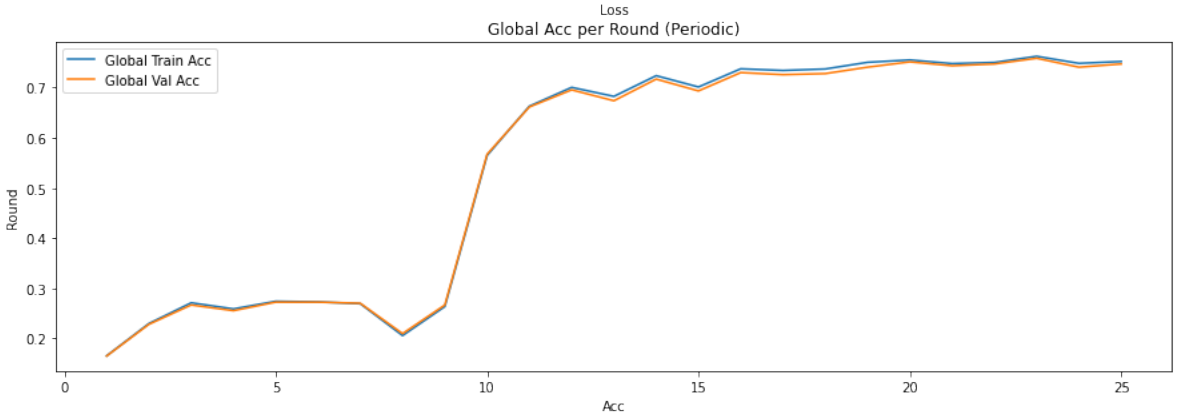
Gossip Training with a Periodically Strongly Connected Weight Matrix

We used five different weight matrices that are periodically used as the communication matrix. The matrices are in a way that the graph is periodically strongly connected with a period of five. The data distribution is iid, and we set the number of agents in the graph to six. The algorithm is performed for 25 rounds (during each round we go over the number of batches in the local dataset for each worker). The network architecture and the model hyperparameters are given in section 3. The average loss and accuracy per worker for each batch along with the global loss and accuracy of the aggregate model for each round are plotted below.


Loss per Batch*#workers (Periodic)
Train Loss
Batch*#workers
Loss

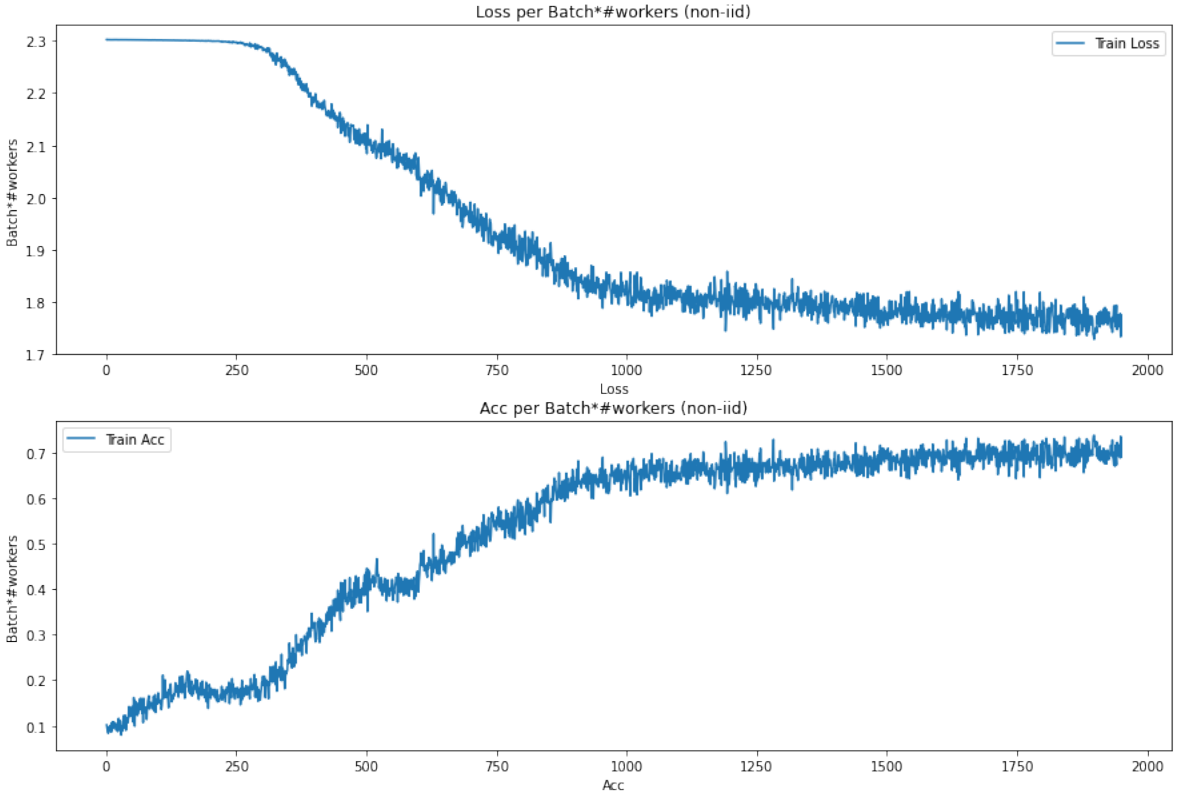

Acc per Batch*#workers (Periodic)
Train Acc
Batch*#workers
Acc

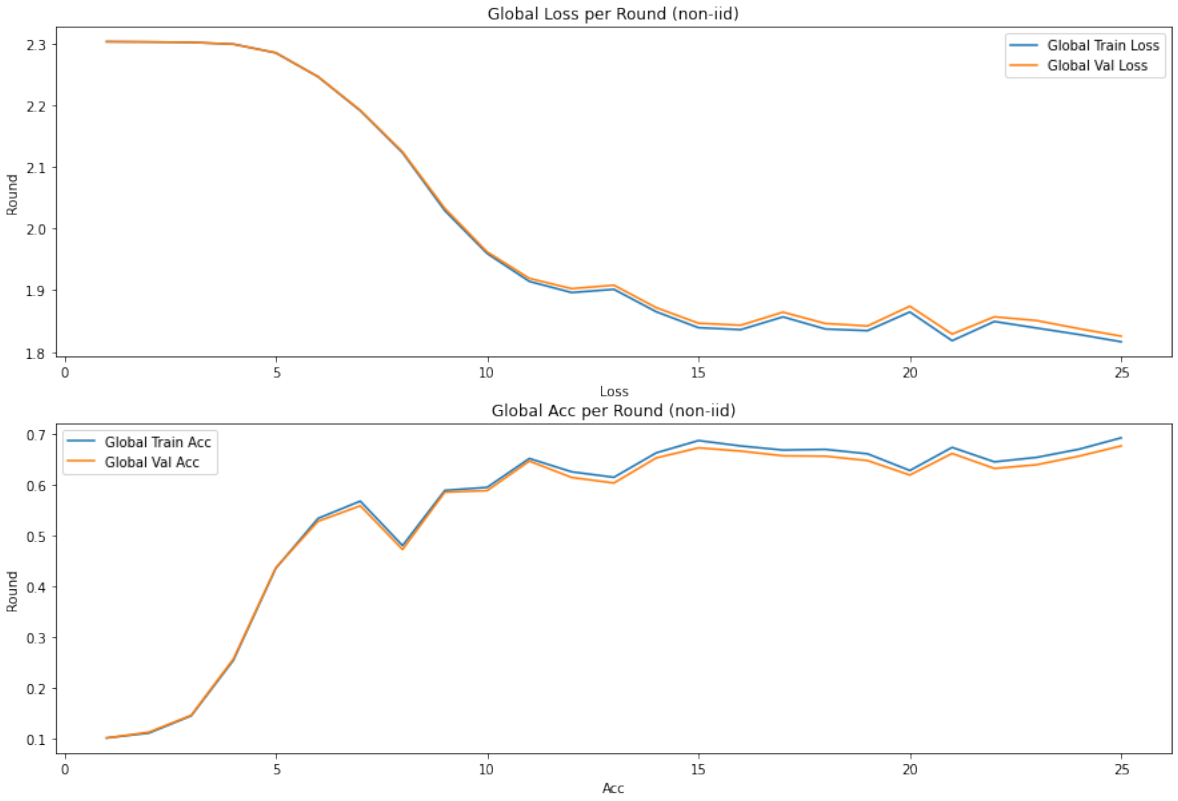

Global Loss per Round (Periodic)
Global Train Loss
Global Val Loss
Round
Loss


Global Acc per Round (Periodic)
Global Train Acc
Global Val Acc
Round
Acc

Original Gossip Training with Non-iid Data Distribution

We added a normal random communication noise multiplied by 0.1 when the agents are communicating their weights. The data distribution is iid, and we set the number of agents in the graph to six. The algorithm is performed for 25 rounds (during each round we go over the number of batches in the local dataset for each worker). The network architecture and the model hyperparameters are given in section 3. The average loss and accuracy per worker for each batch along with the global loss and accuracy of the aggregate model for each round are plotted below.





6. Discussion and Conclusion

In this project, we applied different variants of gossip training to the problem of image classification for iid and non-iid dataset distributions. We can have a few observations from the results:

- The results on the train and test sets are not much different, as the global model is not directly trained on the train set, yet, it is the aggregation of weights from the models that have been trained on portions of the train set.
- The addition of noise worsens the performance of gossip learning, and if this noise is large, it can completely disrupt the training process.
- As long as the weight matrix is periodically strongly connected, gossip training is successful, and different variants of weight matrices do not affect the speed of convergence and accuracy of the models.
- When the datasets are iid, gossip training's performance is similar to FedProx, yet the loss and accuracy plots show less fluctuation. The stable performance may be attributed to the fully-distributed nature of training.
- The gossip learning performance in general is on a par with federated learning and they are both slightly worse than the centralized training performance.

References

[1] Blot, Michael, et al. "Gossip training for deep learning." arXiv preprint arXiv:1611.09726 (2016).

[2] Hegedűs, István, Gábor Danner, and Márk Jelasity. "Gossip learning as a decentralized alternative to federated learning." IFIP International Conference on Distributed Applications and Interoperable Systems. Springer, Cham, 2019.

[3] McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." Artificial intelligence and statistics. PMLR, 2017.

[4] Boyd, Stephen, et al. "Randomized gossip algorithms." IEEE transactions on information theory 52.6 (2006): 2508-2530.

[5] Xiao, Han, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." arXiv preprint arXiv:1708.07747 (2017).

[6] Geiping, Jonas, et al. "Inverting gradients-how easy is it to break privacy in federated learning?." Advances in Neural Information Processing Systems 33 (2020): 16937-16947.

[7] https://github.com/hafezgh/Federated-Deep-Learning-for-Image-Classfication

[8] Li, Tian, et al. "Federated optimization in heterogeneous networks." *Proceedings of Machine Learning and Systems* 2 (2020): 429-450.