به نام خدا

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر

بهینه سازی توزیع شده و یادگیری
Distributed Optimization and Learning

پروژه 1
First Project

عرفان میرزایی
Erfan Mirzaei

810199289

دکتر کبریایی
Dr.Kebriaei

آذر ماه 1400
December 2021

Contents

Abstract

In this project, we implemented federated learning algorithms as samples of the decentralized optimization algorithms for the case when client datasets are i.i.d. or non-iid. Then, compare the results with the centralized version.

For this purpose, a convolution neural network and the cifar10 dataset are used. In the first part, we trained the network in a centralized manner, as in the standard cases in deep learning. We implemented the FedAvg algorithm, one of the most famous algorithms in federated learning, when the client datasets are iid. In the following, the iid assumption was discarded. Thus, we split datasets in a non-iid manner. Therefore for non-iid datasets, we implemented the FedProx algorithm and compared the results to the FedAvg Algorithm.

Section 1) Problem Definition

In this project, I implemented federated learning algorithms as one of the decentralized optimization algorithms for the case when client datasets are i.i.d. or non-iid—then compared the results with the centralized version.

The objective of training a neural network is to find a set of parameters that minimize the cost function over the training set. Usually, these cost functions are not convex, and Stochastic Gradient Descent is one of the most famous algorithms used for optimizing this objective. However, to attain good generalization and performance on test datasets, you should have a large centralized dataset, which is not the case for many applications. Therefore, there is so much desire for decentralized optimization methods to solve this problem. This situation is usually called Federating learning; when we have decentralized datasets and want to use all the data, however, we do not or can not access the whole data for privacy, communication cost, etc. In this case, the importance of a good decentralized optimization algorithm appears. The FedAvg algorithm[1] is one of the simple decentralized optimization algorithms which shows good performance, and as a part of this project, I implemented it.

The problem formulation for the centralized scenario is as follows:

$$\min_{w \in \mathbb{R}^d} f(w) \qquad \text{where} \qquad f(w) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

This relation for the decentralized scenario became to the below equation:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w)$$

Federated learning, like any other learning method, faces several challenges. The systems and statistical heterogeneity are two examples of these challenges. Statistical heterogeneity means client datasets have different distributions and are non-iid, which makes optimization harder. Systems heterogeneity is about when the storage, computational, and communication capabilities of each device in federated networks may differ due to variability in hardware (CPU and memory), network connectivity (3G, 4G, 5G, and Wi-Fi), and power (battery level). In the second part of this project, we tried to implement FedProx Algorithm [2] to address these problems.

Section 2) Network Architecture

In this project, we want to use a convolutional neural network for the classification of images for comparing centralized and decentralized versions of deep learning. For this purpose, we used the network in [1] for the cifar10 dataset.

The network has 2 Convolutional layers and two max-pooling layers, and on top of that, there are three fully connected layers for classifying. The below tables show the architecture.

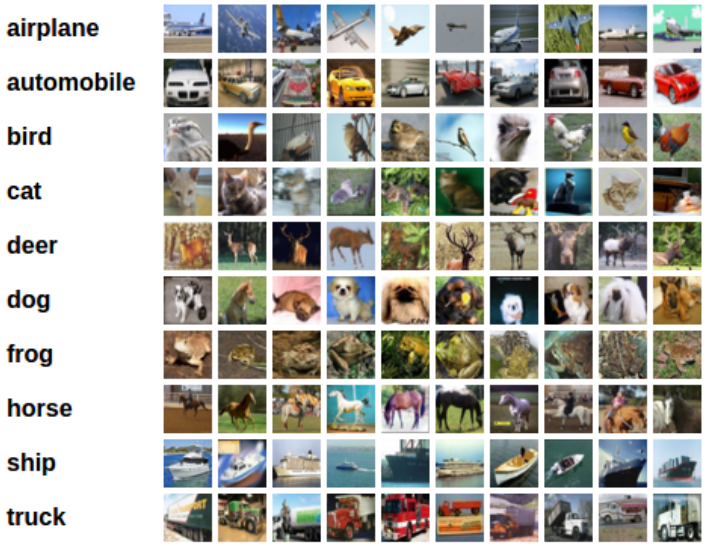
Operation Layer		# filters	Size of Each Filter	Stride value	Padding value
Layer 1	Convolution	6	5 * 5 * 3	1 * 1	-
	(BN), ReLU	-	-	-	-
	Max-Pooling[1]	1	2 * 2	2 * 2	-
Layer 2	Convolution	16	5 * 5 * 6	1 * 1	-
	(BN), ReLU	-	-	-	-
	Max-Pooling	1	2 * 2	2 * 2	-

Layer	Input Dimension	Output Dimension
FC1	400	120
FC2	120	84
FC3	84	# Classes(10)

[1] Max Pooling + downsampling

Section 3) Dataset

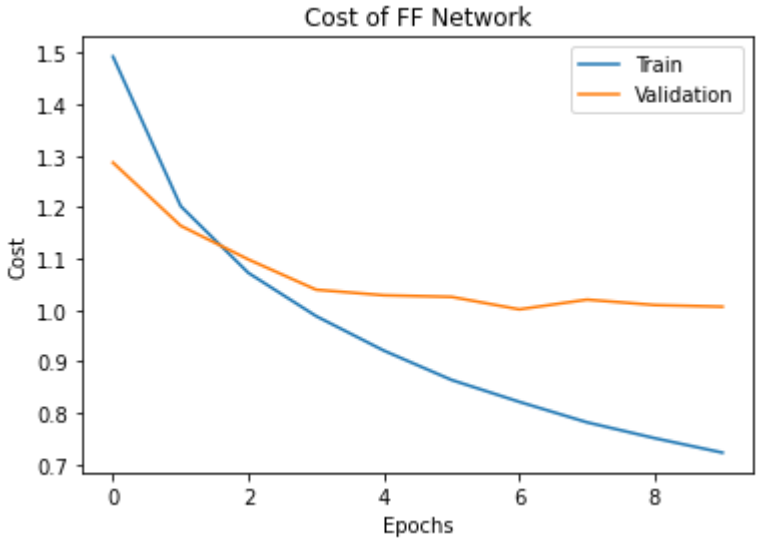
In this project, we used the cifar10 dataset [3]. It has the classes: 'airplane,' 'automobile,' 'bird,' 'cat,' 'deer,' 'dog,' 'frog,' 'horse,' 'ship,' and 'truck.' The images in CIFAR-10 are of size 3x32x32, i.e., 3-channel color images of 32x32 pixels in size.

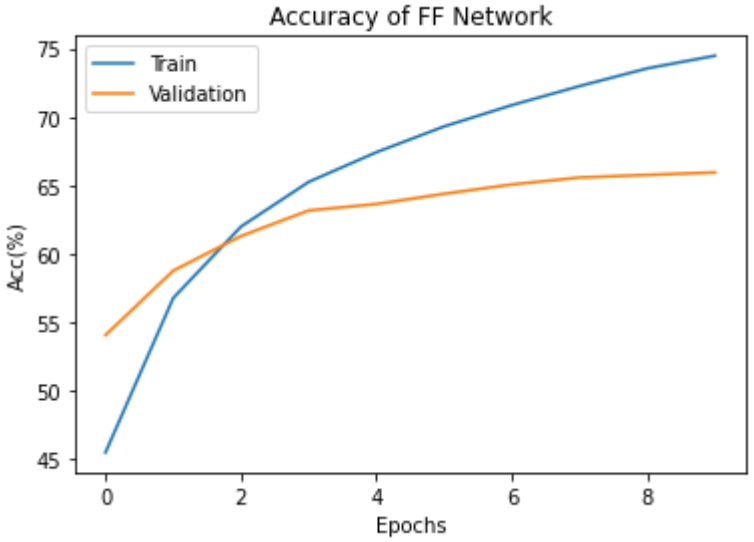


The cifar10 dataset includes 50,000 train images and 10,000 test images. The train images were distributed uniformly for classes; in other words, each class has precisely 5,000 images in the training set.

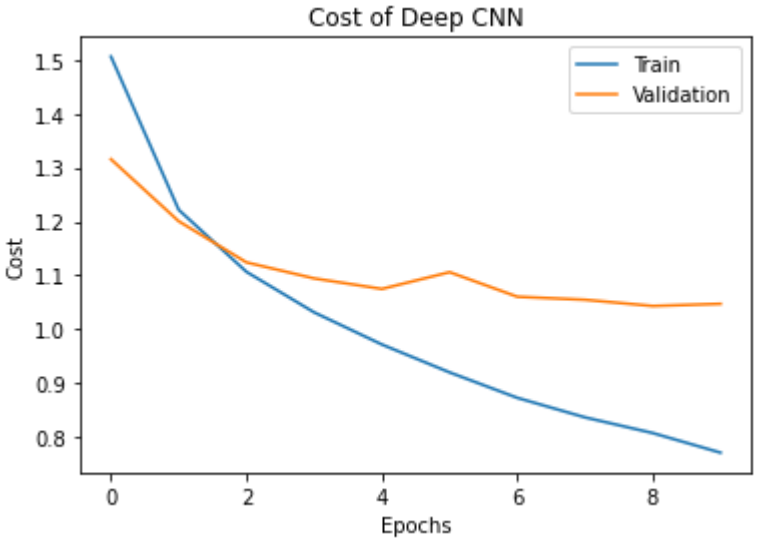
Section 4) Centralized Optimization

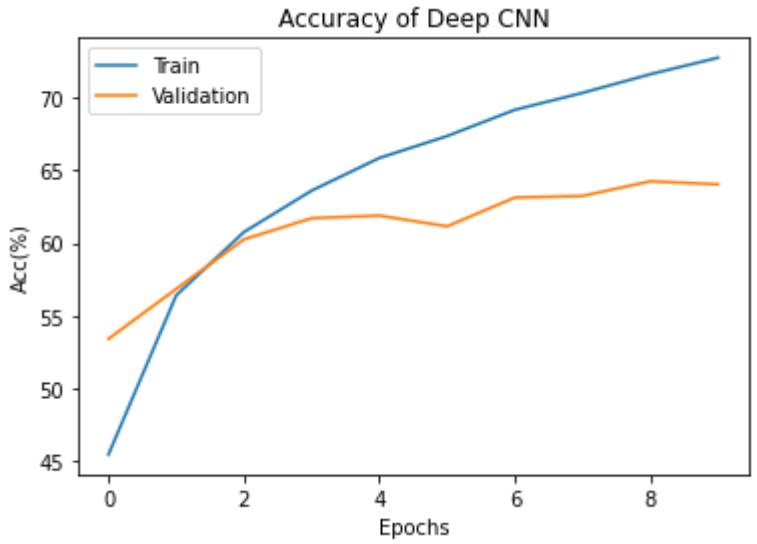
In this section, we trained the architecture described in the previous sections on the cifar10 training sets in a centralized manner. For training of the network, we passed data as batches with a size of 100, as in the paper[1]. We used the "Cross-Entropy" loss function and the "SGD" as an optimizer function with a learning rate of 0.05 and 0.01, and the network was trained for ten epochs on the train and validation datasets. The results are as follows:





learning_rate = 0.05





learning_rate = 0.01

Section 4) Decentralized Optimization (FedAvg)

In this section, we assumed that data are distributed over K clients uniformly and iid, e.g., each client has images from all classes and the same number of data. Furthermore, there is a server that communicates with clients. To solve this decentralized optimization, we implemented the FedAvg algorithm, which was first introduced in [1]. The Pseudo code of this algorithm is as follows:

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

initialize w_0

for each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of m clients)

for each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

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

ClientUpdate(k, w): *// Run on client k*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)

for each local epoch i from 1 to E **do**

for batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return w to server

As noticeable, there are a bunch of hyperparameters in the pseudocode. We used the same hyperparameters as in the original paper.

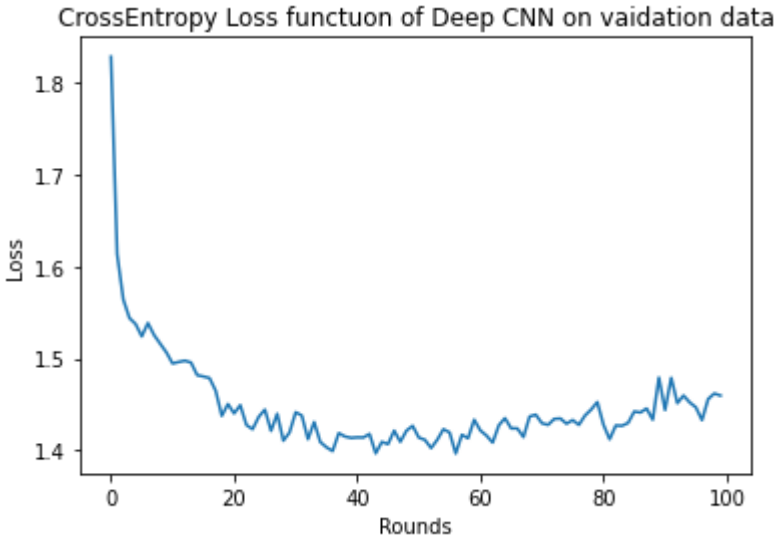
Hyperparameter	E	K	C	B
Value	5	100	0.1	50

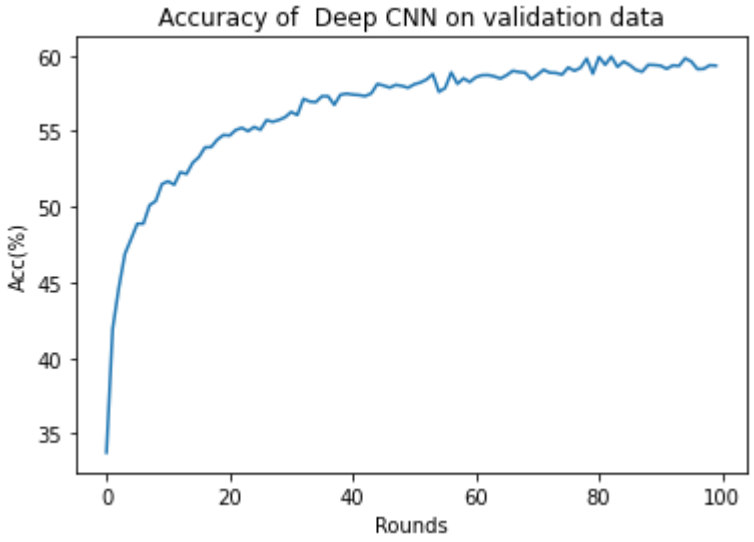
4.1. Splitting the Data

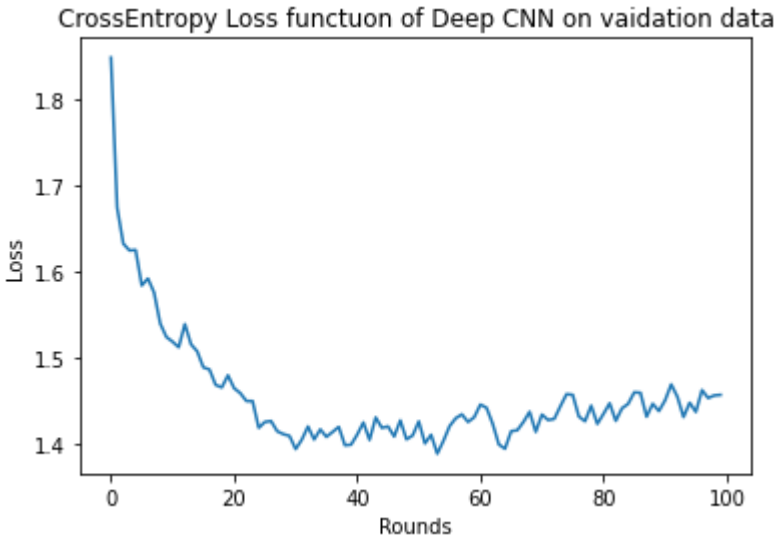
There are 50,000 train images in the cifar10 dataset and ten classes, each with 5,000 train images. Also, we assumed that there are 100 clients to be compatible with the paper. Additionally, for this section, the assumption is that client data are i.i.d; thus, we split the data equally between clients, and each client has 500 images from all ten classes uniformly. In other words, each client has 50 images from each category.

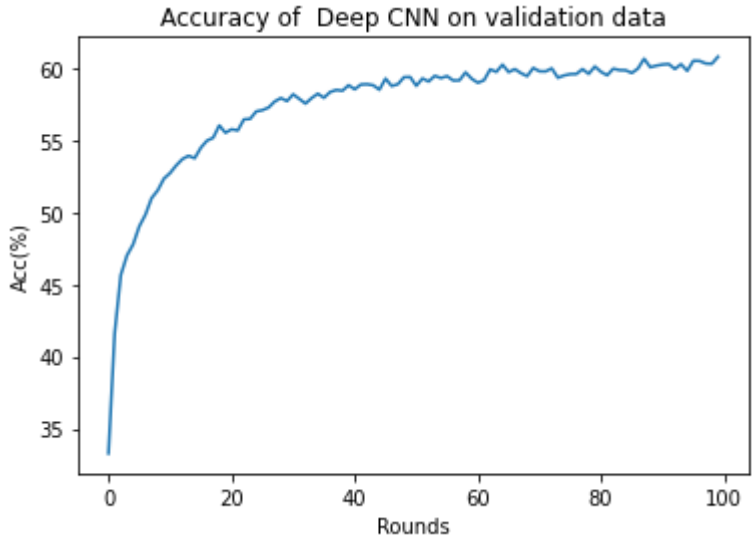
4.2. Implementation Results

We trained the previous architecture on the client dataset based on the FedAvg algorithm with the mentioned hyperparameters. We used the "Cross-Entropy" loss function and the "SGD" as an optimizer function with a learning rate of 0.01 and 0.02. The network was trained for 100 rounds on the train and validation datasets. The results are as follows:









As we can see in the plots, changing the learning rate did not substantially affect the results for these values. Also, the performance in 100 rounds is a bit lower compared to the centralized version, which we think can reach the centralized result if it is trained for more rounds.

Section 5) Decentralized Optimization (FedProx)

In this section, the assumption that client datasets are iid is discarded. However, the other assumption from the previous section is still valid here. As was stated in the problem formulation, the difference in the distribution of client datasets and systems heterogeneity are two examples of challenges for federated learning. For this purpose, for this section, We implemented the PredFox algorithm as stated in [2]. The Pseudo code of this algorithm is as follows:

Algorithm 2 `FedProx` (Proposed Framework)

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

The algorithm is similar to the FedAvg algorithm, but the main difference is in adding a regularization term to the loss function of each client. This regularization term has been used to prevent local weights from becoming very different from global weights. Additionally, in the client update section, each client uses a gamma-inexact minimizer, and the server averages the weights from clients. However, the last changes were ignored in the implementation part of the original paper, and so we do.

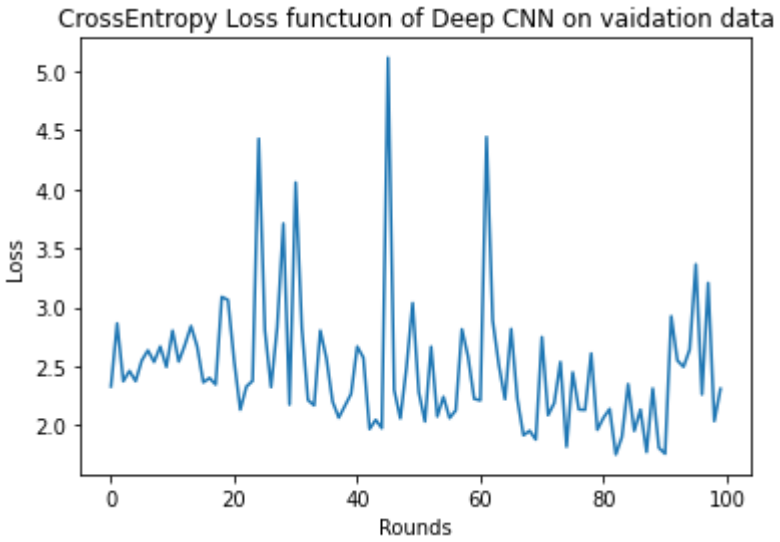
In this part, we used the same hyperparameters from the previous section. Furthermore, try different values for mu (1,2,3) as it has not been reported precisely in the paper.

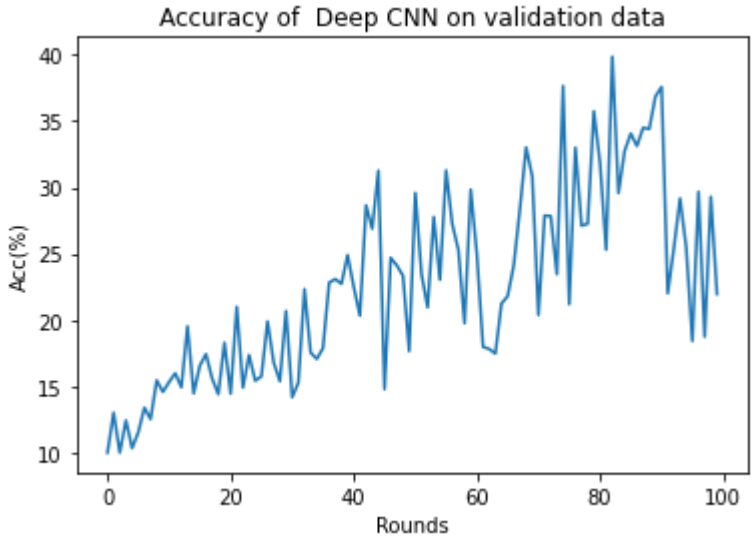
5.1. Splitting the Data

There are 50,000 train images in the cifar10 dataset and ten classes, each with 5,000 train images. Also, we assumed that there are 100 clients to be compatible with the paper. Additionally, this section assumes that client data are not i.i.d. Thus, we split the data differences between clients. Each client has 250 images of 2 classes. In other words, each client has 500 images from 2 classes.

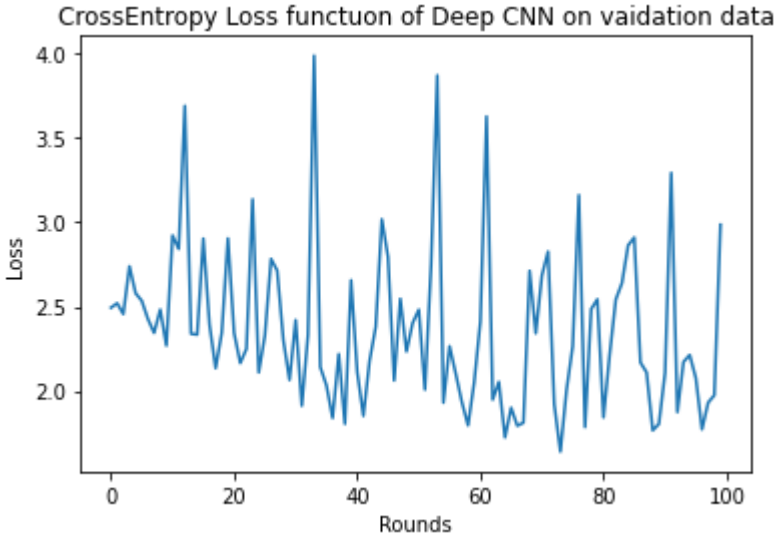
5.2. Implementation Results

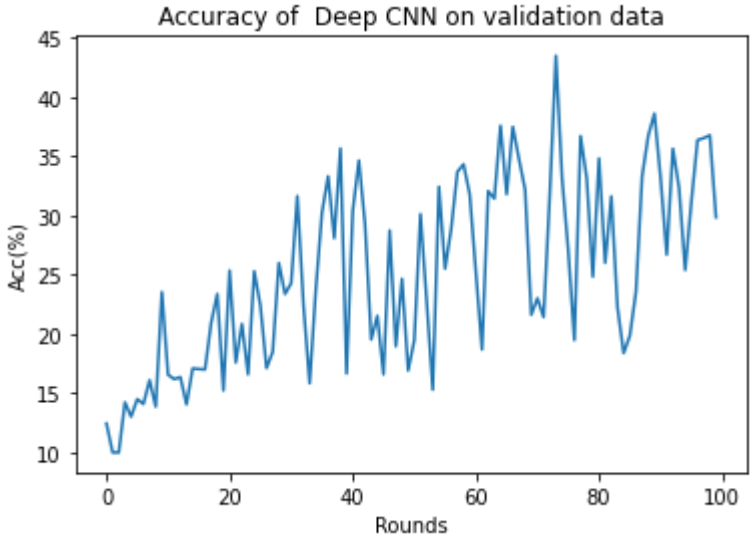
We trained the previous architecture on the client dataset based on the FedAvg and the FedProx algorithm with the mentioned hyperparameters. We used the "Cross-Entropy" loss function and the "SGD" as an optimizer function with a learning rate of 0.01. The network was trained for 100 rounds on the train and validation datasets. The results of the FedAvg algorithm are as follows:





The results of the FedAvg algorithm are as follows:





As we can see in the above plots, because of the differences in the distribution of clients' datasets, there is so much fluctuation in the network's performance. Furthermore, we can conclude that the FedProx algorithm improves the network's performance compared to the FedAvg algorithm. Nevertheless, there is still a considerable gap between when client datasets are i.i.d. and when they are not. These show that statistical heterogeneity can be a severe challenge for federated learning. The other contribution of the FedProx algorithm is when the clients have different systems, which was not tested in this project.

Section 5) Conclusion

In this project, we implemented a convolutional neural network on the cifar10 dataset in 3 three different scenarios:

1) When we have access to all of the training samples(Classic centralized Version)
2) When the data had been split in an i.i.d. manner through the clients
3) When the data had been split in a non-i.i.d. manner through the clients

For this purpose, we implemented the FedAvg and the FedProx algorithms. As we saw in the previous sections, how much we go from the first scenario to the last, the problem becomes more complicated and more complex, and the performance gap increases. Also, the FedProx algorithm showed better performance in the last scenario due to adding the regularizer term in the clients' loss functions.

References:

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

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

[3] Krizhevsky, Alex, Vinod Nair, and Geoffrey Hinton. "The cifar-10 dataset." *online: http://www. cs. toronto. edu/kriz/cifar. html* 55.5 (2014).

[4]Li, Tian, et al. "Federated learning: Challenges, methods, and future directions." *IEEE Signal Processing Magazine* 37.3 (2020): 50-60.