ASSIGNMENT#2

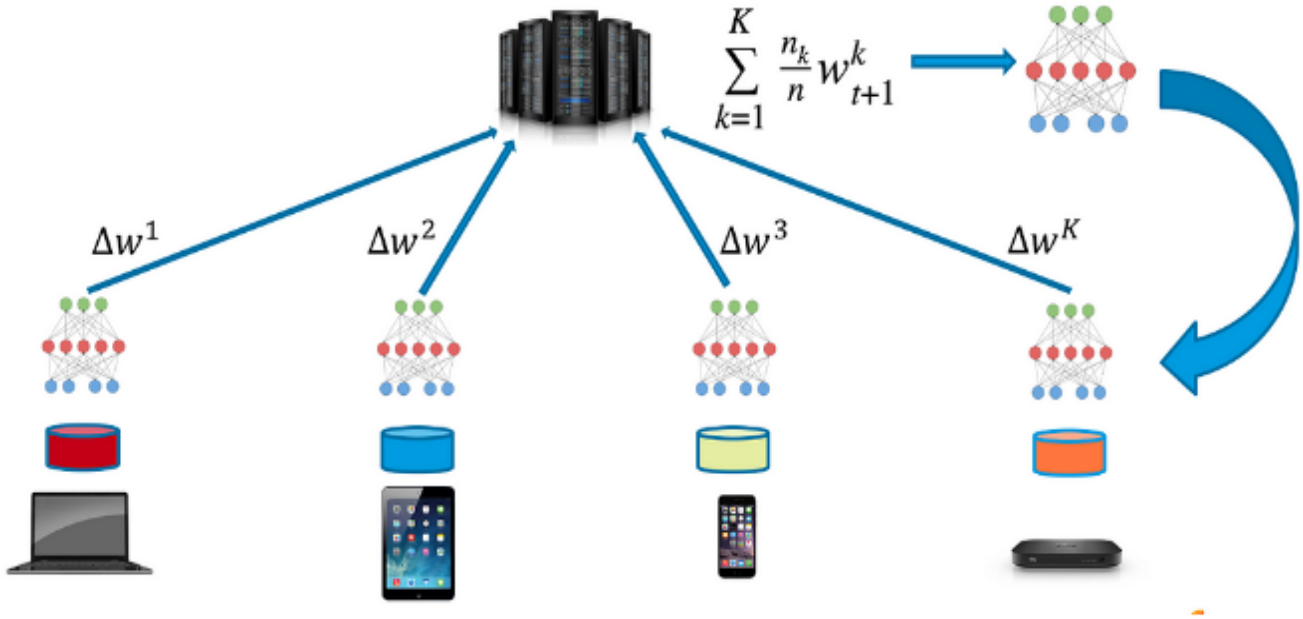
Federated Machine Learning

Submitted by :-
Praneet Karna (2020A7PS1202P)
Jaysheel Shah (2020A7PS0083P)
Nishal Shah (2020A3PS0321P)

Introduction

Federated learning (also known as collaborative learning) is a machine learning technique that trains an algorithm via multiple independent sessions, each using its own dataset. In contrast to conventional centralized machine learning methods, where local datasets are amalgamated for a singular training session, federated learning stands out by acknowledging the diversity of local data distributions.

This innovative approach empowers various stakeholders to collaboratively develop a shared and resilient machine learning model without the necessity of sharing raw data,thus addressing critical issues such as data privacy, data security, data access rights and access to heterogeneous data.



As depicted in the figure above, rather than training the model on a centralized server, the model is executed independently by various clients using their own data, subsequently updating the model weights. The resulting model weights constitute the weighted average of all individual model weights. This process occurs within a single epoch, and the model can undergo multiple epochs.

Implementation of Federated Learning

1) **Dataset Taken**:

   In order to illustrate the working of Federated Learning, we have used an animal dataset which contain 4 types of animals named cat, deer, dog and horse. We have a total of 2800 images comprising of all 4 classes.

   Dataset of the battery life and signal of each client in each epoch is also used as metadata which is randomly generated for each client. The current criteria for selecting clients is more than or equal to 30% battery life and signal strength of more than or equal to 2/5.

   Preprocessing : As the images taken are all of different sizes, before feeding them into the model for training, we resize them to a standard size of 256x256 for easier training. We have also used python scripts to separate the images of each class into different folders according to the client number and the epoch number.

   For our current implementation, we have considered 4 total epochs and 10 available clients for every epoch, thus requiring the dataset(training) to be divisioned into 40 different folders.

2) **Model used:**

   We have used a CNN(Convolutional Neural Network) as the model. It is a multilayer neural network which outputs of 1 of the 4 animal classes - Dog, Cat, Horse or Deer.

The CustomCNN model we created is a Convolutional Neural Network designed for animal classification with 4 classes (cat, dog, horse, deer). It consists of two convolutional layers (conv1, conv2) with ReLU activation, max-pooling layers (pool1, pool2), and two fully connected layers (fc1, fc2). The model learns hierarchical features to distinguish between classes during training using CrossEntropyLoss and SGD optimizer. The architecture is well-suited for image classification tasks, leveraging convolutional and fully connected layers for feature extraction and classification.

```
class CustomCNN(nn.Module):
    def __init__(self, num_classes=4):
        super(CustomCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, stride=1, padding=1)
        self.relu1 = nn.ReLU()
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1)
        self.relu2 = nn.ReLU()
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)
        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(32 * 64 * 64, 128)
        self.relu3 = nn.ReLU()
        self.fc2 = nn.Linear(128, num_classes)
```

3) **Client selection process**:

We have taken 10 clients with particular battery life and signal strength. The battery life ranges from 0 to 100 and signal strength ranges from 1 to 5.

If the battery life of a client is less than 30 or the signal strength of a client is less than 2, then the client is not allowed to train the model.
The model runs for 4 epochs.

We have created a separate function which takes the client number and epoch number as input and checks the metadata.csv file to obtain the metadata details for that particular client in that epoch. Then, the values are compared to our criteria set earlier and if both values are appropriate, the corresponding client is approved to train the model for that epoch.

```
[ ]  def approve_client(client_number, epoch_number):
       meta_data_path = '/content/drive/My Drive/DividedLabeledDataset/metadata.csv'
       meta_data = pd.read_csv(meta_data_path)
       index = epoch_number*10 + client_number - 1

       if(meta_data.at[index, 'battery life']>=30 and meta_data.at[index,'signal strength']>=2):
         return True
       return False
```

4) **Main implementation process:**

In every epoch, we first use the above defined function to approve the clients out of the 10 existing clients based on their metadata. Then, the base model is copied for each approved client and is then trained by each client on their own dataset separately. Finally, the averages of weights of all the models trained in this epoch are taken and fed to the main centralised model, and updated. Then, we use the testing dataset of ~700 images to test the accuracy of the model after training in this epoch and print it out, along with the confusion matrix.

Following is the function we have used to take the average of weights of different models. The arrays of weights of each individual model are passed into this function as arguments and the final array of average weights is returned.

```
[ ]  def get_average_of_models(models):
         n = len(models)
         average = models[0].state_dict()

         for i in range(1, n):
             current = models[i].state_dict()
             average = {x: average[x] + current[x] for x in average}

         average = {x: parameter / n for x, parameter in average.items()}

         return average
```

Results :

Along with federated learning, we have also trained a model on the same training dataset without using federated learning(centrally) in order to compare the accuracies achieved after same number of epochs of training. The following are the details of the clients approved in each epoch, along with the specific testing accuracies after each epoch, with the confusion matrices :-

Results for the training on federated learning model - 4 epochs

```
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_1
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_3
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_5
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_6
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_8
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_9
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_10
Training completed for epoch 0
Testing Accuracy for epoch 0: 20.96%
Confusion Matrix:
[[ 65 135   0   0]
 [ 31  84   0   0]
 [ 46 187   0   0]
 [ 35 128   0   0]]
Epoch 1/4 completed
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_11
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_13
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_14
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_16
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_17
Training completed for epoch 1
Testing Accuracy for epoch 1: 22.22%
Confusion Matrix:
[[ 72 120   8   0]
 [ 34  76   5   0]
 [ 51 172  10   0]
 [ 41 120   2   0]]
Epoch 2/4 completed
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_22
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_23
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_24
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_30
Training completed for epoch 2
Testing Accuracy for epoch 2: 17.30%
Confusion Matrix:
[[  0 190  10   0]
 [  0 110   5   0]
 [  0 220  13   0]
 [  0 158   5   0]]
Epoch 3/4 completed
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_33
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_38
Training started for /content/drive/My Drive/DividedLabeledDataset/Part_40
Training completed for epoch 3
Testing Accuracy for epoch 3: 28.97%
Confusion Matrix:
[[ 57  50  93   0]
 [ 29  44  42   0]
 [ 33  95 105   0]
 [ 32  52  79   0]]
Epoch 4/4 completed
```

Final accuracy achieved on testing dataset - 28.97%

Results for training on centralized model(without federated learning) - 4 epochs

```
Testing Accuracy without federated learning: 32.77%
```

Thus, we can see that federated learning model accuracy is almost equal to the conventional training model's accuracy, and it takes less time and computation power from a single machine.