Assignment 3

Federated Learning

Hongyi Hao

1. Questions

1. Initialize and train a new model using each of the specified hyperparameter configurations. For each run, plot the training loss versus the number of epochs, and report the training and test accuracies. Each run is expected to complete within 5 to 30 minutes, depending on your computer's specifications.

(a) 15pts: batch size = 32, num communications = 50, learning rate = 0.1, num workers = 2, num local steps = 5.

(b) 15pts: batch size = 32, num communications = 50, learning rate = 0.1, num workers = 4, num local steps = 5.

(c) 15pts: batch size = 32, num communications = 50, learning rate = 0.1, num workers = 8, num local steps = 5.

(d) 15pts: batch size = 32, num communications = 50, learning rate = 0.1, num workers = 4, num local steps = 20.

2. Coding Task

2.1 Finish the Training Function for Each Worker

```python
def train_model(args):
    model, optimizer, criterion, random_sampler, batch_size = args

    train_loader_random = torch.utils.data.DataLoader(
        dataset=random_sampler.data_source,
        batch_size=batch_size,
        sampler=random_sampler
    )

    total_loss = 0
    model.train()

    for i, (images, labels) in enumerate(train_loader_random):
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()

    return model, total_loss
```

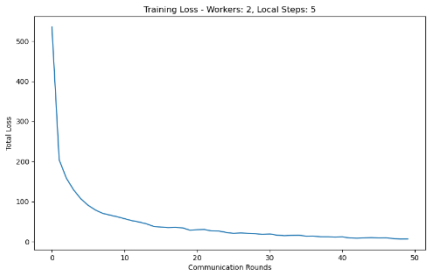
2.2 Finish the Training Loop

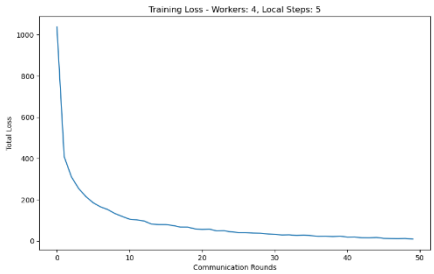
```python
train_loader = torch.utils.data.DataLoader(dataset=train_dataset, batch_size=batch_size, shuffle=True)
random_sampler = RandomSampler(
    data_source=train_dataset,
    replacement=True,
    num_samples=num_local_steps * len(train_loader)
)

losses_array = []
model = Net()

for _ in range(num_communications):
    models = duplicate_model(model, num_workers)
    optimizers = [optim.SGD(model.parameters(), lr=learning_rate) for model in models]

    args_list = [(models[i], optimizers[i], criterion, random_sampler, batch_size)
                 for i in range(num_workers)]

    with Pool(num_workers) as pool:
        results = pool.map(train_model, args_list)

    models = [result[0] for result in results]
    total_loss = sum(result[1] for result in results)

    ensemble_model_params = average_models(models)
    model.load_state_dict(ensemble_model_params)

    losses_array.append(total_loss)

plt.figure(figsize=(10, 6))
plt.plot(losses_array)
plt.xlabel('Communication Rounds')
plt.ylabel('Total Loss')
plt.title(f"Training Loss - Workers: {num_workers}, Local Steps: {num_local_steps}")
plt.show()
```

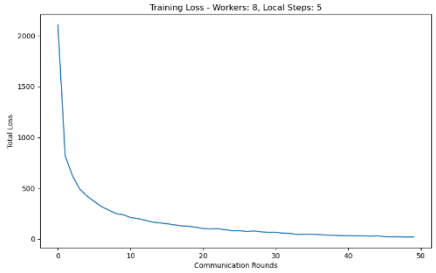
2.3 Testing the Model

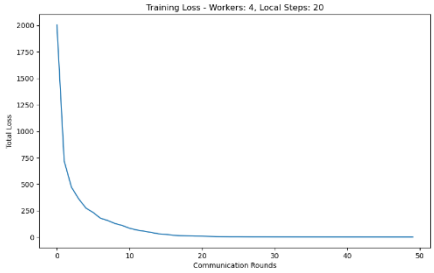
```python
test_loader = torch.utils.data.DataLoader(dataset=test_dataset, batch_size=batch_size, shuffle=False)
model.eval()
with torch.no_grad():
    correct = 0
    total = 0
    for images, labels in test_loader:
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
    test_accuracy = 100 * correct / total
```

3. Question Solution









Accuracy:

Configuration	**Training Accuracy**	**Test Accuracy**	**Loss Convergence Speed**	**Overfitting Risk**
a	99.81%	98.38%	Moderate	Low
b	99.93%	98.24%	**Fast**	Medium-Low
C	99.94%	**98.42%**	Fastest	Medium-Low
d	**100.00%**	98.38%	Slow	High