Big Data processing
Assignment2 Report

32217072 Mobile System Engineering 김도익

Big Data Processing Assignment 2

목차

1. Instruction

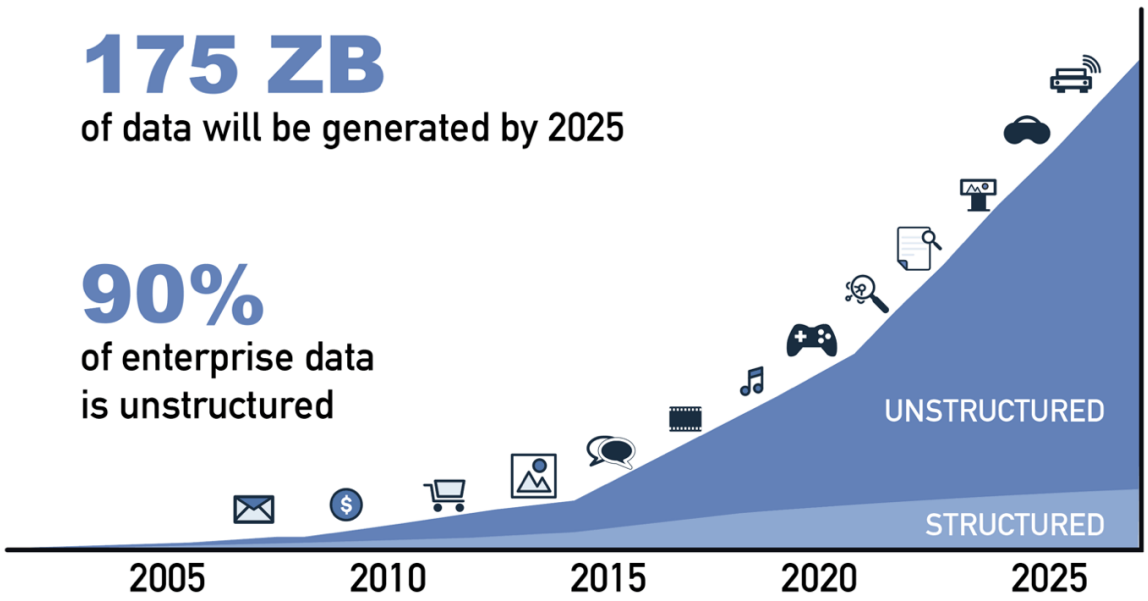

Fig 1. Growth of Global Data by 2025

현대 사회에서는 데이터의 폭발적인 증가가 이루어지고 있다. 특히, 머신 러닝 기술의 발전에 따라 사용할 수 있는 데이터의 양과 범위가 지속적으로 확장되고 있다. Fig 1과 같이, 2025년에는 175ZB의 데이터가 생성될 것으로 예상되며, 데이터의 90%가 sensing logs, images와 같은 비정형 데이터로 구성될 것으로 예상된다. 따라서 이러한 데이터를 효과적으로 처리하는 기술의 중요성이 점점 커지고 있다.

이 레포트에서는 비정형 데이터를 처리하여 여러 디바이스에서 학습을 수행한 후 이를 통합하는 연방학습(Federate Learning)에 대해 설명한다. 특히 연방학습의 대표적인 프레임워크인 FedAvg 중심으로 그 동작 원리와 특성을 설명한다.

또한, MNIST 데이터셋을 활용하여 연방학습을 구현하고, 하이퍼파라미터인 Epochs, Client 비율, Round수를 조정하며 각 하이퍼파라미터가 학습 결과에 미치는 영향을 분석하고, 성능을 분석한다. 이를 통해 연방학습의 개념과 실질적인 응용 방법을 이해하고, 비정형 데이터를 활용한 머신 러닝의 가능성을 탐구한다.

2. 배경지식

본 레포트에서 다루는 내용을 이해하기 위해 필요한 배경 지식으로 Federated Learning과 FedAvg에 대해 설명한다.

2.1) Federated Learning

Federated Learning이란 분산된 데이터를 갖고 있는 사용자가 데이터를 중앙 서버로 전송하지 않고, 각자의 디바이스에서 데이터를 로컬 학습을 통해 모델을 훈련하고, 이를 중앙 서버로 전송하여 글로벌 모델을 통합하는 머신 러닝 기술이다. 이 기술은 데이터의 프라이버시와 보안을 유지하면서도 협력적으로 학습을 가능하게 한다.

Federated Learning은 다음과 같은 단계로 작동하며, 이를 1라운드라고 한다.

1. **Client selection**: 학습에 참여할 device를 선택한다.
2. **Broadcast**: 서버에 저장된 initial model을 선택된 각 client의 디바이스에 분배한다.
3. **Local model update**: 각 클라이언트는 로컬 데이터를 활용하여 초기 모델을 학습시키고, 이를 업데이트한다.
4. **Aggregation**: 클라이언트에서 학습된 모델 파라미터를 서버로 전송하고, 서버는 이를 집계하여 통합 연산을 수행한다.
5. **Global model update**: 서버는 집계된 파라미터를 바탕으로 글로벌 모델을 업데이트하며, 다음 라운드 학습을 위한 새로운 초기 모델로 사용한다.

Federated Learning은 여러 디바이스에서 로컬 데이터를 활용하여 모델을 학습하고, 학습된 모델 파라미터만 중앙 서버로 전송한다. 이를 통해 통신 비용을 줄이며 학습을 수행할 수 있고, 데이터가 중앙 서버로 전송되지 않고 로컬 디바이스에 남아있으므로, 민감한 데이터를 보호할 수 있다.

또한, Federated Learning은 데이터 전체를 전송하지 않고 모델 파라미터만 주고받으므로 통신 비용을 크게 줄이고 네트워크 대역폭을 효율적으로 사용할 수 있다. 이를 통해 통신 비용을 절감하며, 디바이스의 배터리 소모와 네트워크 에너지 사용량도 감소시킬 수 있다.

이 기술은 각 디바이스가 자신의 데이터를 사용하여 학습하므로 사용자 맞춤형 학습(Personalization)이 가능하다. 이를 통해 스마트폰의 키보드 자동 완성이나 음성 인식 시스템처럼 개인화된 결과를 제공할 수 있으며, 글로벌 모델의 일반화된 지식과 로컬 모델의 개인화된 지식을 결합하여 더 나은 성능을 도출할 수 있다.

마지막으로, Federated Learning은 많은 디바이스에서도 확장 가능한 학습을 지원한다. 여러 디바이스가 동시에 로컬 학습을 수행할 수 있어 병렬 처리가 가능하며, 분산 환경에서도 효율적으로 글로벌 모델을 업데이트할 수 있다. 이를 통해 대규모 데이터 환경에서도 높은 확장성을 제공한다.

2.2) Fed Avg

FedAvg는 Federated Learning에서 가장 개선된 방식으로, 클라이언트들이 로컬 데이터를 사용해 학습한 후 모델의 가중치를 중앙 서버에서 평균화하여 글로벌 모델을 업데이트하는 방식이다. FedAvg에서 각 클라이언트는 자신의 데이터를 매니 배치 단위로 나누어 학습을 수행하며, 이를 통해 메모리사용량을 줄이고, 제한된 컴퓨팅 성능을 가진 디바이스에서도 효과적으로 학습을 진행할 수 있다.

또한 클라이언트가 로컬 데이터를 Local Epochs만큼 반복적으로 학습을 수행하도록 설계되어 있다. 이는 통신 빈도를 낮추는 동시에, 클라이언트가 자체 데이터를 통해 충분한 학습을 진행하도록 한다.

클라이언트에서 학습된 모델의 가중치를 중앙 서버에서 평균화하여 글로벌 모델을 업데이트하는 것도 FedAvg의 핵심 특징이다. 모델 가중치 평균화(Model Aggregation)는 각 클라이언트의 데이터 크기에 비례하여 가중치를 부여해 안정적인 글로벌 모델 학습을 가능하게 한다. 이를 통해 데이터 분포가 균등하지 않은 Non-IID 환경에서도 학습 성능을 유지할 수 있다.

마지막으로, FedAvg는 학습 과정에서 다양한 하이퍼파라미터(예: 배치 크기, 로컬 학습 횟수, 클라이언트 비율 등)를 조정할 수 있는 유연성을 제공한다. 이를 통해 각 클라이언트의 데이터 크기, 네트워크 상태, 디바이스의 계산 능력에 따라 최적의 학습 환경을 설정할 수 있다.

3.Implement

이 섹션에서는 MNIST 데이터셋을 활용해 FedAvg을 구현하여 학습을 진행한다. 다양한 학습 파라미터를 조정하면서 결과를 분석하고, 이를 그래프로 시각화하여 성능과 특성을 평가한다.

3.1) Task 1

3.1.1) average_weights

```python
def average_weights(selected_models):
    avg_state_dict = deepcopy(selected_models[0])
    for key in avg_state_dict.keys():
        for i in range(1, len(selected_models)):
            avg_state_dict[key] += selected_models[i][key]
        avg_state_dict[key] = torch.div(avg_state_dict[key], len(selected_models))

    return avg_state_dict
```

Fig 2. Average_weights

이 함수는 연방 학습과정에서 선택된 클라이언트들의 모델 가중치를 평균화한 후, 글로벌 모델 업데이트에 사용된다. Deepcopy를 통해 선택된 모델의 가중치를 복사하여 평균화에 사용할 초기 상태 딕셔너리를 생성한다. 각 키에 대해 반복문을 실행하여 선택된 클라이언트들의 가중치를 동일한 키에 누적하여 더한다. 클라이언트 수로 나누어 각 가중치의 평균값을 계산하여 반환한다.

3.1.2) federated_training

```python
def federated_training(num_rounds, num_clients, client_fraction, local_epochs, train_loaders, test_loader, lr=0.001):
    # 글로벌 모델 초기화
    global_model = SimpleCNN()
    global_model_state = global_model.state_dict()

    # 각 라운드별 정확도 기록
    round_accuracies = []

    # 각 라운드에서 선택할 클라이언트 수 계산
    num_selected_clients = max(1, int(client_fraction * num_clients))

    for round_num in range(num_rounds):
        print(f"Round {round_num + 1}/{num_rounds}")

        # 현재 라운드에서 클라이언트 무작위 선택
        selected_clients = random.sample(range(num_clients), num_selected_clients)

        # 로컬 모델 수집
        local_models = []
        for client_id in selected_clients:
            local_model = SimpleCNN()
            local_model.load_state_dict(global_model_state)  # 글로벌 모델 상태로 초기화

            # 로컬 학습 수행
            local_model_weights = train_local(local_model, train_loaders[client_id], epochs=local_epochs, lr=lr)
            local_models.append(local_model_weights)

        # 가중치 평균화
        global_model_state = average_weights(local_models)

        # 글로벌 모델 업데이트
        global_model.load_state_dict(global_model_state)

        # 글로벌 모델 평가
        test_accuracy = test_model(global_model, test_loader)
        print(f"Test Accuracy after Round {round_num + 1}: {test_accuracy:.2f}%")
        round_accuracies.append((round_num + 1, test_accuracy))  # 라운드별 정확도 저장

    return global_model, round_accuracies
```

Fig 3. Federated_training

이 함수는 Federated Learning에서 여러 라운드에 걸쳐 클라이언트들이 학습한 모델을 중앙 서버에서 집계하고 글로벌 모델을 업데이트하는 역할을 한다. 함수의 동작순서는 다음과 같다.

1. 글로벌 모델 초기화: 글로벌 모델을 초기화하고, 이를 state_dict형태로 저장한다. 초기화된 글로벌 모델은 각 라운드의 시작점으로 사용된다.
2. 클라이언트 선택: 매 라운드에서 전체 클라이언트 중 일부를 랜덤하게 선택하여 학습에 참여시킨다. 선택된 클라이언트의 수는 전체 클라이언트와 클라이언트 비율변수인 client_fraction을 통해 정해진다.
3. 로컬 학습 수행: 선택된 클라이언트는 초기화된 글로벌 모델을 복사하고, 각 로컬 데이터를 사용해 local_epochs만큼 학습을 수행한다.
4. 모델 가중치 평균화: 클라이언트에서 학습된 로컬 모델 가중치를 평균화한다.
5. 글로벌 모델 업데이트 및 평가: 평균화된 가중치를 글로벌 모델에 반영하여 업데이트하고, 테스트 데이터셋을 통해서 모델의 성능을 평가하여 정확도를 기록한다.

3.1.3) Settings

하이퍼파라미터	값
클라이언트 수	100
클라이언트 비율	0.01
로컬 학습 횟수	1
라운드 수	20
학습률	0.001

table 1. Hyper parameter

Task 1 실험에서는 하이퍼파라미터를 table1과 같이 설정하여 학습을 진행하였다.

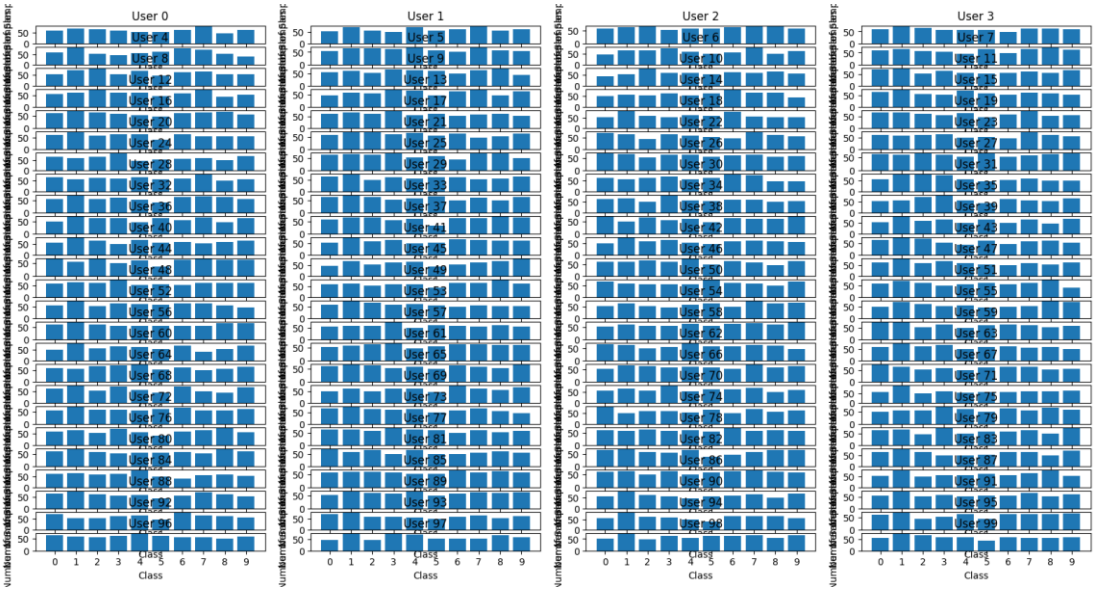

Fig 3. Data distribution

Fig 3에서 각 100명의 User가 보유한 MNIST class인 0부터 9까지의 분포가 비슷하게 유지되는 것

을 확인할 수 있다. 이는 데이터가 IID(Independent and Identically Distributed)형태로 분배되었음을 알 수 있고 이로 인해 각 사용자의 데이터에 대해 고르게 학습할 수 있어 글로벌 모델의 성능이 안정적으로 향상될 가능성이 있다.

3.1.4) 결과 분석

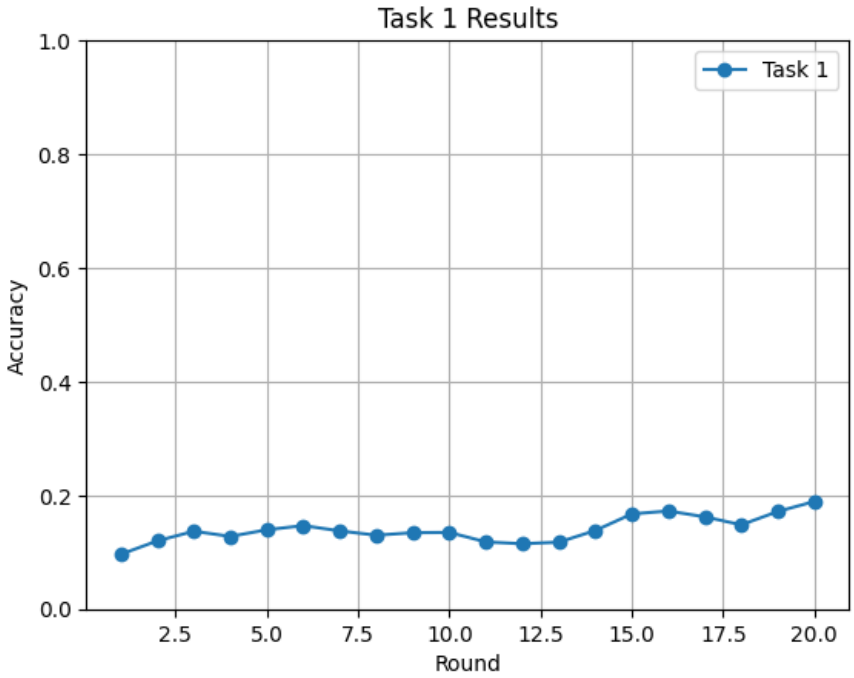

Fig 5. Task 1 Result

fig 5그래프는 각 라운드가 진행됨에 따라 글로벌 모델의 정확도 변화를 나타낸다. 초기 1라운드에서는 정확도가 9.68%로 초기에는 클라이언트 데이터에 대한 학습이 거의 이루어지지 않았음을 알 수 있다. 라운드가 진행됨에 따라 정확도가 증가하며 20라운드에서는 18.91%의 정확도에 도달한다. 이는 Federated Learning이 반복 학습을 통해 모델 성능을 향상시킨다는 것을 볼 수 있다.

그러나 최종 모델의 정확도가 18.91%에 불과하다는 점을 고려할 때, 이는 신뢰할 만한 모델 성능이 아니라고 할 수 있다. 이를 개선하기 위해 클라이언트 비율을 증가시키거나 로컬 학습 횟수를 늘리는 등 하이퍼파라미터를 조정해야 할 것으로 보인다.

3.2) Task 2

3.2.1) Settings

하이퍼파라미터	Task 2-Experiment 1	Task 2-Experiment 2	Task 2-Experiment 3
클라이언트 수	60	60	60
클라이언트 비율	0.01	0.01	0.01
로컬 학습 횟수	1	1	1
라운드 수	20	30	40
학습률	0.001	0.001	0.001

table 2. Hyperparameter

Task 2실험에서는 table 2와 같이 클라이언트 수를 60으로 설정하고, 라운드 수를 20,30,40으로 조정해가며 각 성능을 비교하고 분석한다.

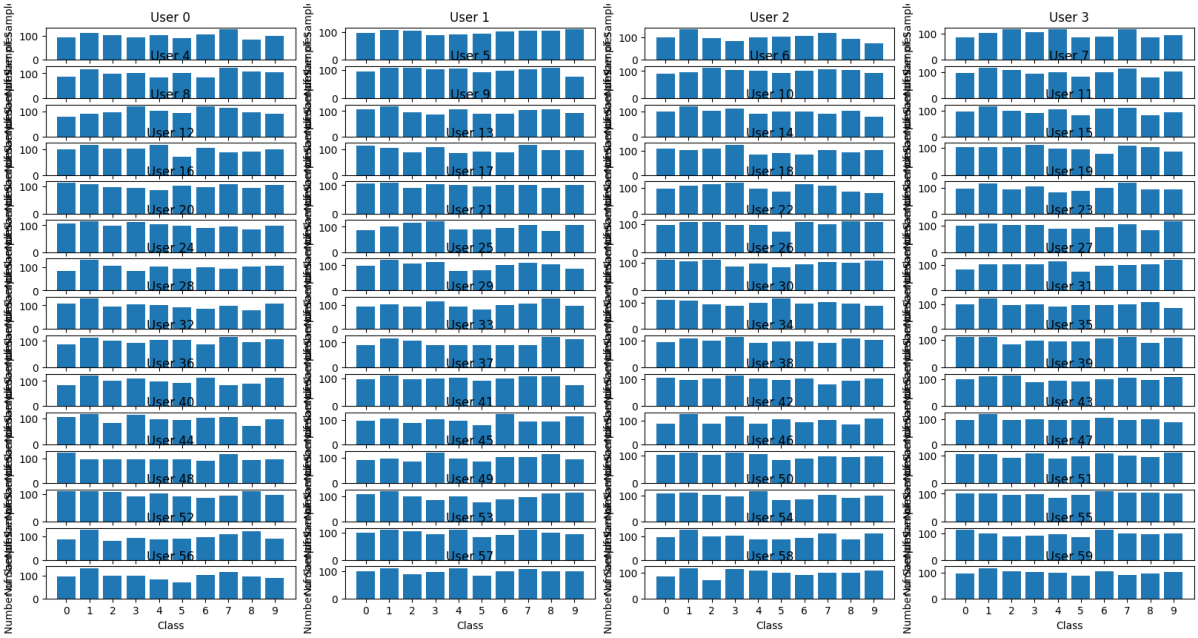

Fig 6. Data distribution

Task 2의 각 실험에서는 Fig 6과 같이 60명의 사용자가 보유한 MNIST 클래스(0부터 9까지)의 데이터 분포가 IID 형식으로 구성되어 있다. 이 데이터셋은 공정한 성능 평가를 위해 각 20라운드, 30라운드, 40라운드 실험에 공통적으로 사용된다.

3.2.2) 결과

각 실험의 결과는 Fig 7, 8, 9에 시각화 되어 있으며, 각각 20, 30, 40 라운드에 따른 정확도의 변화를 나타낸다.

1. Task 2-Experiment 1 (20라운드)

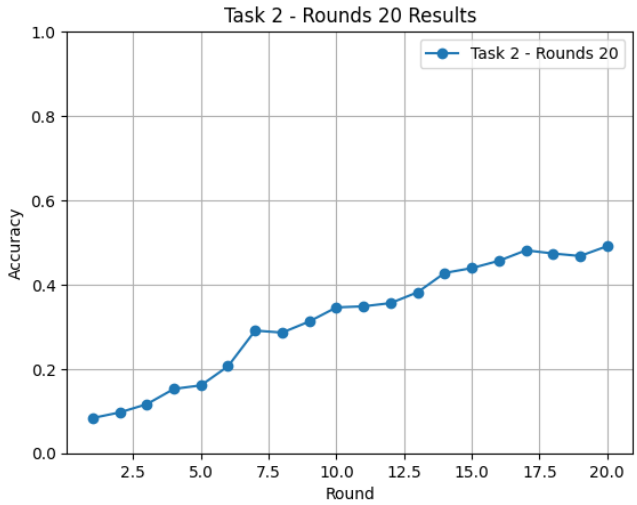

Fig 7. Task 2 – Round 20

초기 정확도는 8.35%로 시작하였으며, 20라운드 종료 시점에 49.12%의 정확도를 기록하였다.

2. Task 2-Experiment 2 (30라운드)

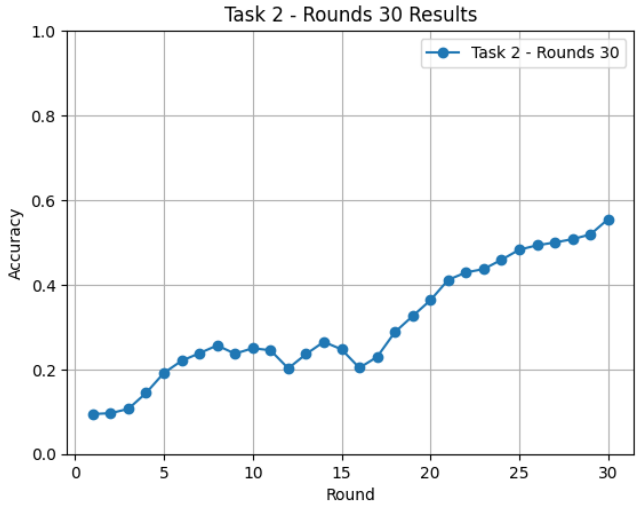

Fig 8. Task 2 – Round 30

초기 정확도는 9.44%로 Experiment 1과 비슷하게 시작하였으나, 30라운드 종료 시점에 55.49%의 정확도를 기록하였다.

3. Task 2-Experiment 3 (40라운드)

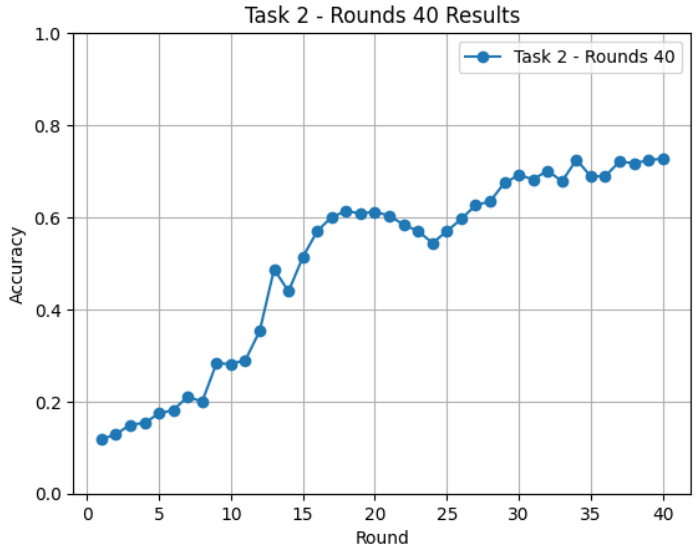

Fig 9. Task 2 – Round 40

정확도는 11.78%에서 시작하여 40라운드 종료 시점에 72.76%의 정확도까지 꾸준히 증가하였다. 30 라운드까지는 성능이 뚜렷하게 향상되었으나, 이후로는 성능 개선 폭이 점차 줄어드는 것을 볼 수 있다.

3.2.3) 결과 분석

라운드 수를 증가시킬수록 글로벌 모델의 가중치가 더 세밀하게 업데이트되며, 실험마다 높은 정확도를 보였다. 이는 충분한 라운드 수가 정확도 측면에서 중요한 역할을 한다는 점을 알 수 있다. 그러나 일정 라운드 이후에는 성능 개선 속도가 점차 느려지는 현상이 관찰되었다. 특히 30라운드에서 40라운드까지의 정확도 향상이 상대적으로 미미하였다. 높은 정확도가 요구되는 경우, 라운드 수를 추가하는 것이 유리하지만, 통신 비용 계산 비용의 균형을 고려할 때, 더 적은 라운드로 유사한 정확도를 달성하기 위해 다른 하이퍼파라미터를 조정하는 등의 개선방안이 필요할 것이다.

3.3) Task 3

3.3.1) Settings

하이퍼파라미터	Task 3-Experiment 1	Task 3-Experiment 2	Task 3-Experiment 3
클라이언트 수	60	60	60
클라이언트 비율	0.01	0.01	0.01
로컬 학습 횟수	1	5	10
라운드 수	20	20	20
학습률	0.001	0.001	0.001

table 3. Hyperparameter

Task 2실험에서는 table 3과 같이 클라이언트 수를 60으로 설정하고, 로컬 학습 횟수를 1, 5, 10으로 조정해가며 각 성능을 비교하고 결과를 분석한다.

Task 3의 각 실험에서는 Task2에서 사용된 데이터를 동일하게 사용하였다. 60명의 사용자가 보유한 MNIST 클래스(0부터 9까지)의 데이터 분포가 IID 형식으로 구성되어 있다. 이 데이터셋은 공정한 성능 평가를 위해 각 1로컬 학습 횟수, 5로컬 학습 횟수, 10로컬 학습 횟수 실험에 공통적으로 사용된다.

3.3.2) 결과

각 실험의 결과는 Fig 10, 11, 12에 시각화 되어 있으며, 각각 1, 5, 10 로컬 학습 횟수에 따른 정확도의 변화를 나타낸다.

1. Task 3-Experiment 1 (1로컬 학습 횟수)

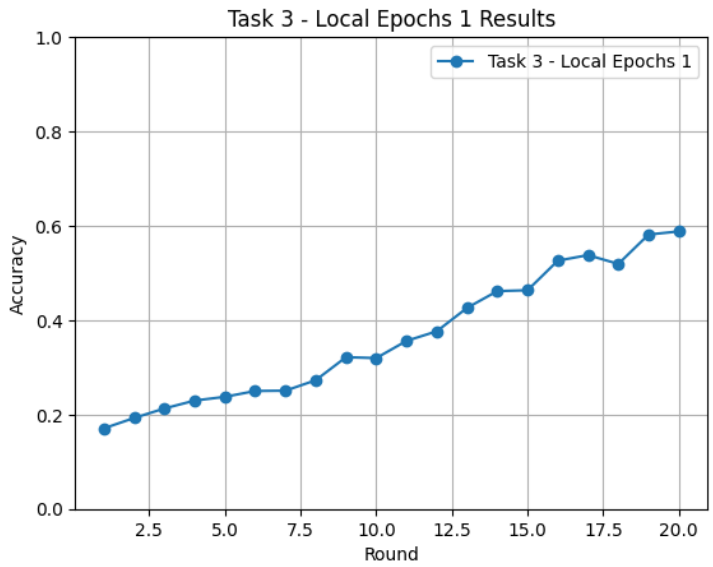

Fig 10. Task 3 – Local Epochs 1

로컬 학습 횟수가 1일 때 초기 정확도는 17.08%로 시작하여, 20라운드 종료 시점에서 최종 정확도는 58.81%로 나타났다. 초기부터 정확도가 점진적으로 상승하는 것을 볼 수 있다.

2. Task 3-Experiment 2 (5로컬 학습 횟수)

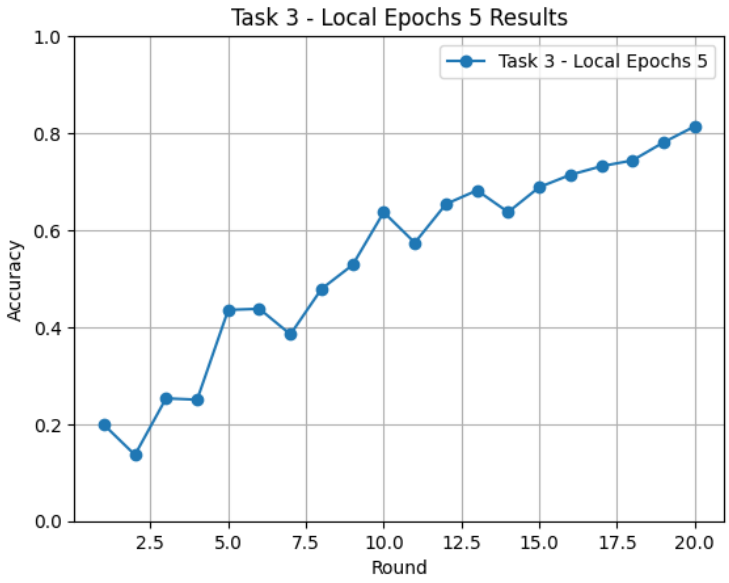

Fig 11. Task 3 – Local Epochs 5

로컬 학습 횟수가 5일 때 초기 정확도는 19.88로 시작하여, 20라운드 종료 시점에서 최종 정확도는 81.40%로 나타났다. 로컬 학습 횟수가 1 대비 정확도 상승 폭이 높으며, 최종 정확도도 확연히 높은 것을 볼 수 있다.

3. Task 3-Experiment 3 (10로컬 학습 횟수)

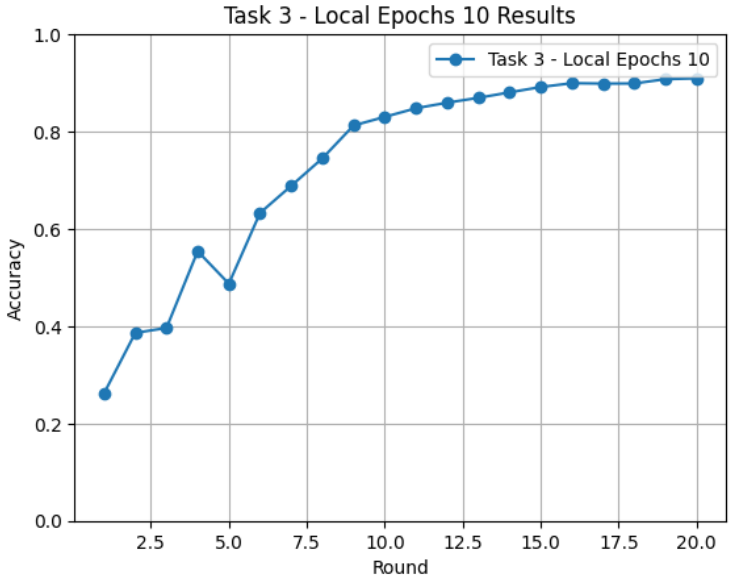

Fig 12. Task 3 – Local Epochs 10

초기 정확도는 26.25%로 시작하여, 20라운드 종료 시점에서 최종 정확도는 90.94%로 가장 높은 정확도를 보인다. 10라운드 이내에 빠르게 80% 이상의 정확도를 달성하는 것을 볼 수 있다.

3.3.3) 결과 분석

로컬 학습 횟수가 증가함에 따라 초기 정확도와 최종 정확도 모두 눈에 띄게 상승하였다. 이는 클라이언트가 로컬 데이터를 기반으로 충분히 학습하므로 글로벌 모델의 성능이 향상되었음을 알 수 있다. 로컬 학습 횟수 10에서 가장 빠르게 높은 정확도에 도달하였으며 5에서도 비교적 빠른 속도로 정확도가 상승하였다. 반면, 1에서는 정확도 향상 속도가 느리고 최종 성능도 낮았다.

학습 라운드가 증가할수록 정확도는 안정화되며, 특히 로컬 학습 횟수가 10일 때 15라운드 이후에는 큰 변화 없이 90% 이상의 높은 정확도를 유지하였다. 이는 라운드 수가 충분할 때 높은 로컬 학습 횟수가 모델 수렴을 가속화하고 안정적인 성능을 제공함을 보여준다.

높은 정확도가 요구되는 상황이라면 로컬 학습 횟수를 늘리는 것이 적합하다. 특히 로컬 학습 10은 빠른 학습 속도와 높은 최종 성능을 제공하였다. 그러나 로컬 학습 횟수를 증가할 수 록 클라이언트의 계산 비용이 증가할 수 있다. 따라서 통신 비용과 계산 부하의 균형을 맞추기 위해서는 꼭 로컬 학습 수를 늘리는 것만이 효율적인 대안이 될 수는 없다.

로컬 학습 횟수가 낮다면, 충분한 라운드 수를 제공하여 모델의 성능을 보완하거나, 높은 로컬 학습 횟수를 가지고 있다면 적은 라운드 수로도 높은 정확도에 도달할 수 있으므로, 시스템의 조건에 따라서 적절한 하이퍼파라미터 설정을 하는 것이 중요한 것으로 보인다.

3.4) Task 4

3.4.1) Settings

하이퍼파라미터	Task 4-Experiment 1	Task 4-Experiment 2	Task 4-Experiment 3
클라이언트 수	100	100	100
클라이언트 비율	0.01	0.05	0.1
로컬 학습 횟수	1	5	10
라운드 수	20	20	20
학습률	0.001	0.001	0.001

table 4. Hyperparameter

Task 4실험에서는 table 4와 같이 클라이언트 수를 100으로 설정하고, 매 라운드 학습에 사용되는 클라이언트 수를 조정하는 하이퍼파라미터인 클라이언트 비율을 0.01, 0.05, 0.1의 순서로 조정하며 실험하였다.

Task 4의 각 실험에서는 Task1에서 사용된 데이터를 동일하게 사용하였다. 100명의 사용자가 보유한 MNIST 클래스(0부터 9까지)의 데이터 분포가 IID 형식으로 구성되어 있다. 이 데이터셋은 공정한 성능 평가를 위해 각 0.01, 0.05, 0.1의 클라이언트 비율 실험에 공통적으로 사용된다.

3.4.2) 결과

각 실험의 결과는 Fig 13, 14, 15에 시각화 되어 있으며, 각각 0.01, 0.05, 0.1 클라이언트 비율에 따른 정확도의 변화를 나타낸다.

1. Task 4-Experiment 1 (0.01 클라이언트 비율)

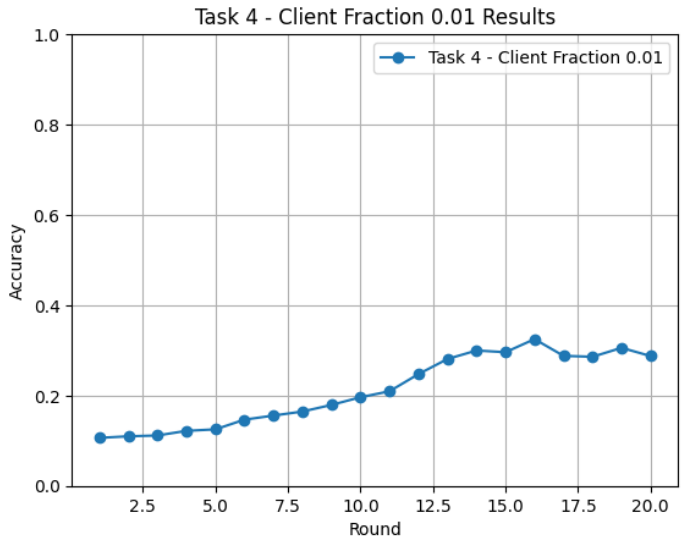

Fig 13. Task 4 – Client Fraction 0.01

초기 라운드에서 정확도가 천천히 증가하며, 20라운드 종료 시점에 28.75%의 정확도를 달성했다.

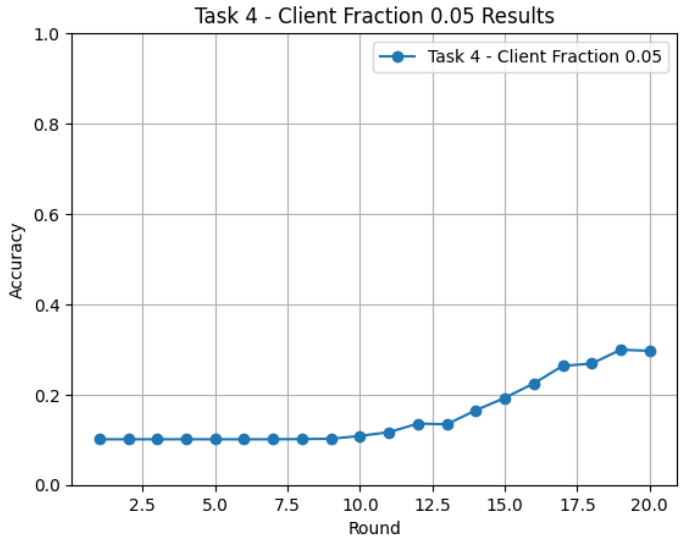

Fig 14. Task 4 – Client Fraction 0.05

초기 라운드에서 정확도가 천천히 증가하며, 20라운드 종료 시점에 29.68%의 정확도를 달성했다. 0.01 비율에 비해 최종 정확도가 상승되긴 하였으나, 그 차이가 미미하다.

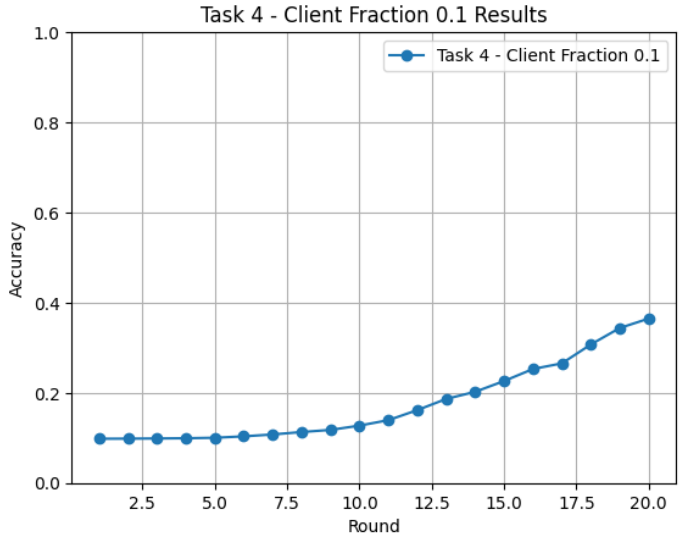

Fig 14. Task 4 – Client Fraction 0.1

초기 라운드의 정확도는 약10%로, 라운드가 증가할수록 정확도는 천천히 증가하며 다른 비율과 비슷한 수준이지만 최종 정확도가 36.47%로 가장 높게 나타났다.

3.4.3) 결과 분석

클라이언트 비율이 낮을수록 매 라운드 학습에 참여하는 클라이언트의 수가 적어 정확도가 상대적 상대 낮은 것을 볼 수 있다. 이는 클라이언트 비율이 적을수록 데이터의 다양성을 글로벌 모델이 충분히 반영하지 못한 것으로 보인다. 클라이언트 비율이 증가함에 따라 학습에 더 많은 클라이언트가 참여하면서 데이터 분포가 글로벌 모델에 더 잘 반영되어, 정확도가 증가하였다.

하지만 다른 파라미터 조정에 비해 정확도의 상승폭이 크지 않은 것을 보아, 클라이언트의 비율을 늘려도, 로컬 학습 횟수가 충분하지 않다면 개별 클라이언트에서 학습된 정보가 글로벌 모델에 효과적으로 반영되지 않는 것으로 보이고, 라운드 수가 충분하지 않다면 클라이언트 비율 증가만으로는 극적인 정확도의 향상을 기대하긴 어려워 보인다.

4. 결론

본 레포트에서는 Federated Learning의 성능에 영향을 미치는 주요 하이퍼파라미터를 조정하여 글로벌 모델의 학습 성능을 분석하였다. 클라이언트 비율, 라운드 수, 로컬 학습 횟수, 클라이언트 수 등 다양한 조건에서 실험들 진행한 결과 각 하이퍼파라미터가 글로벌 모델의 성능에 서로 다른 방식으로 기여함을 확인할 수 있었다.

높은 성능의 글로벌 모델을 얻기 위해서는 각 하이퍼파라미터 간의 균형을 적절히 조정하는 것이 중요하다. 라운드 수, 로컬 학습 횟수, 클라이언트 비율은 모두 성능에 영향을 미치지만, 지나치게 증가시키면 통신 비용과 계산 비용이 과도하게 증가할 수 있다. 따라서 시스템의 제약 조건과 목표에 따라 적절한 하이퍼파라미터를 선택하는 것이 Federated Learning 구현의 핵심이라고 할 수 있다.