

|                    |                    |
|--------------------|--------------------|
| Course Title (과목명) | 인공지능통신             |
| HW Number (HW 번호)  | Project 2          |
| Submit Date (제출일)  | 2021-05-26         |
| Grade (학년)         | 4학년                |
| ID (학번)            | 20151483, 20161482 |
| Name (이름)          | 이창헌, 박준용           |

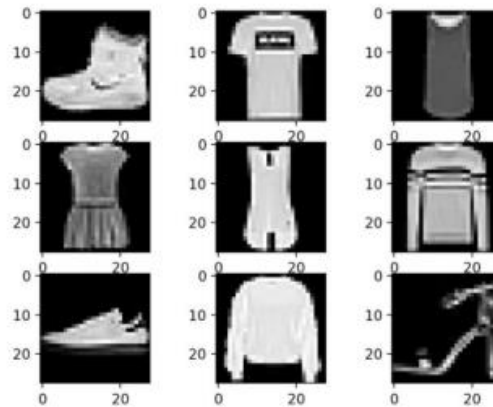
## Project 1

### 1. 과제 목표

- 본 프로젝트는 3명의 유저가 있는 상황에서 연합학습(FL)을 사용하여 신경회로망을 학습하는 것을 목표로 함.
- 학습 데이터와 테스트 데이터가 모두 포함된 2가지 데이터가 (IID, non-IID) 제공됨.
- Centralized Learning (CL)과 FL 비교해보기

### 2. 모델 구조

#### 1) Dataset



- 0 T-shirt/top • 3 Dress • 6 Shirt • 9 Ankle boot
- 1 Trouser • 4 Coat • 7 Sneaker
- 2 Pullover • 5 Sandal • 8 Bag

프로젝트 1과 동일하게 흑백으로 된 60000개의 28 x 28 크기의 패션 관련 데이터 셋으로 구성되어 있다. 각각 sample은 0~9의 10가지 class로 label 되어 있고 흑백 이미지는 1 channel임으로 1x28x28의 크기를 가지고 있다.

이번 프로젝트에서는 60000개의 data가 3명의 유저 Alice, John, Peter에게 각각 20000개씩 있다고 가정하고 학습을 진행하였다.

## 2) ResNet (Residual Neural Network)

프로젝트에서 Centralized Learning(CL), Federated learning(FL), individual learning(개별 데이터 학습)에 사용된 모델은 공통적으로 ResNet18이며 18개의 layers로 구성된 ResNet18의 architecture은 아래와 같다.

학습 시 hyper parameters는 공통적으로 batch size = 128, learning rate = 0.001, number of epochs = 10을 사용하였다.

| ResNet18 Architecture |               |  | [C, W, H]     | 비고              |
|-----------------------|---------------|--|---------------|-----------------|
| Input                 |               |  | [1, 28, 28]   |                 |
| 1                     | Conv1         | Conv2d (kernel: 3, padding:1)            | [64, 28, 28]  |                 |
| 2                     | Layer1 block0 | Conv2d (kernel: 3, padding:1)            | [64, 28, 28]  |                 |
| 3                     |               | Conv2d (kernel: 3, padding:1)            | [64, 28, 28]  |                 |
| 4                     | Layer1 block1 | Conv2d (kernel: 3, padding:1)            | [64, 28, 28]  |                 |
| 5                     |               | Conv2d (kernel: 3, padding:1)            | [64, 28, 28]  |                 |
| 6                     | Layer2 block0 | Conv2d (kernel: 3, stride: 2, padding:1) | [128, 14, 14] |                 |
| 7                     |               | Conv2d (kernel: 3, padding:1)            | [128, 14, 14] | down sample     |
| 8                     | Layer2 block1 | Conv2d (kernel: 3, padding:1)            | [128, 7, 7]   |                 |
| 9                     |               | Conv2d (kernel: 3, padding:1)            | [128, 7, 7]   |                 |
| 10                    | Layer3 block0 | Conv2d (kernel: 3, stride: 2, padding:1) | [256, 4, 4]   |                 |
| 11                    |               | Conv2d (kernel: 3, padding:1)            | [256, 4, 4]   | down sample     |
| 12                    | Layer3 block1 | Conv2d (kernel: 3, stride: 2, padding:1) | [256, 2, 2]   |                 |
| 13                    |               | Conv2d (kernel: 3, padding:1)            | [256, 2, 2]   |                 |
| 14                    | Layer4 block0 | Conv2d (kernel: 3, stride: 2, padding:1) | [512, 1, 1]   |                 |
| 15                    |               | Conv2d (kernel: 3, padding:1)            | [512, 1, 1]   |                 |
| 16                    | Layer4 block1 | Conv2d (kernel: 3, padding:1)            | [512, 1, 1]   |                 |
| 17                    |               | Conv2d (kernel: 3, padding:1)            | [512, 1, 1]   |                 |
| 18                    | Conv1         | Conv2d (kernel: 1, padding:1)            | [512, 1, 1]   | average pooling |
|                       | fc            | Dense                                    | [512, 10]     |                 |

### 3. 과제 수행 방법

#### (1) 주어진 데이터의 분포 확인하기

제출한 파일중에서 **Project2\_check\_IID.ipynb**에서 주어진 데이터셋이 IID인지 non-IID인지 구분하였다.

주어진 dataset이 IID(independently and identically distributed)의 조건을 만족하려면 dataset에서 모든 item(class)가 동일한 확률 분포를 가지고 dataset의 모든 item이 서로 독립적이어야 한다.

Dataset에서 10개의 class는 서로 독립적이기 때문에 independent의 조건은 만족된다. 따라서 **Project2\_check\_IID.ipynb**에서는 dataset1, dataset2의 class별 데이터의 분포를 확인해서 identically distributed의 조건이 만족하는지 확인해보았다.

```
csv_label_alice1 : [2002, 1947, 2026, 2048, 1999, 1965, 2051, 1980, 1927, 2055]
csv_label_john1 : [1952, 2053, 2046, 1983, 1988, 2003, 2017, 1966, 2046, 1946]
csv_label_peter1 : [2008, 2034, 1971, 1977, 2048, 2035, 1947, 2020, 1943, 2017]
csv_label_alice2 : [5978, 5985, 6023, 0, 0, 0, 0, 0, 0, 2014]
csv_label_john2 : [0, 0, 0, 0, 0, 0, 5963, 5961, 6065, 2011]
csv_label_peter2 : [0, 0, 0, 6067, 5979, 5983, 0, 0, 0, 1971]
```

결과적으로 3명의 유저 Alice, John, Peter에게 주어진 dataset1은 완벽하게 동일하지는 않지만 10개의 class별로 확률 분포가 거의 균일해서 dataset1은 IID dataset이라고 판별하였다.

반대로 3명의 유저 Alice, John, Peter에게 주어진 dataset2는 각각 10개의 class중 4개의 class의 data만 주어지고 나머지 6개의 class의 data는 주어지지 않아서 class별로 확률 분포가 일치하지 않는다. 따라서 dataset2는 non-IID dataset이라고 판별하였다.

Federated learning의 학습에 dataset2를 사용하면 client의 local update에 사용하는 mini-batch의 분포가 full training dataset의 분포와 다르기 때문에 stochastic gradient에 큰 bias가 생겨서 full gradient의 estimate으로 사용하기 힘들 것을 이라고 예상된다.

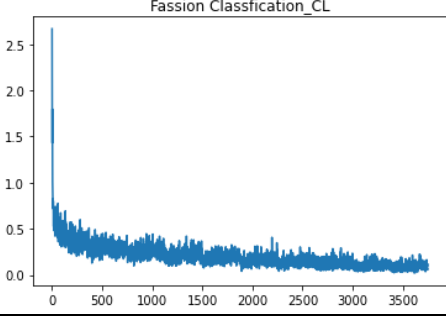
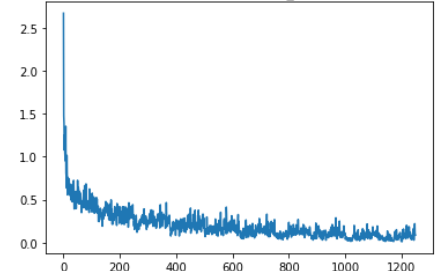
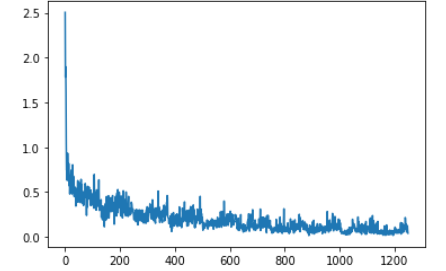
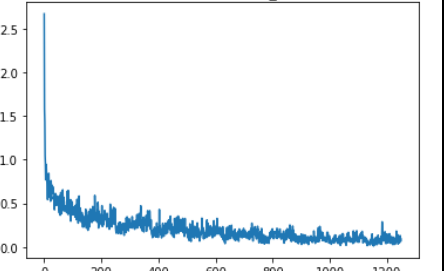
## (2) 3명의 유저와 2종류의 데이터셋을 이용하여 2개의 글로벌 모델 학습하기

제출한 파일중에서 **Project2\_dataset1\_train&test.ipynb**, **Project2\_dataset2\_train&test.ipynb**를 각각 데이터 셋 1,2에 대해서 CL, FL, 그리고 개별 데이터 학습 결과를 얻는데 사용하였다.

### - IID와 non-IID 일 때 학습 성능의 차이 비교 및 분석 필요

앞서 (1)에서 데이터 셋 1은 IID dataset, 데이터 셋 2는 non-IID dataset으로 분석하였기 때문에 아래에서 데이터 셋 1,2에 대하여 CL, FL 그리고 개별 데이터 학습 결과 비교 및 분석하는 과정에서 IID와 non-IID 일 때 학습 성능의 차이를 비교 및 분석하게 된다.

### - 데이터 셋 1에 대하여 CL, FL 그리고 개별 데이터 학습 결과 비교 및 분석

| CL  | FL   |   |
|---|--|---|
|   |  |   |
| Train accuracy : 98.44%   | Train accuracy : -   |   |
| Test accuracy : 90.07%  | Test accuracy : 92.89%   |   |
| 개별 데이터 학습 Alice   | 개별 데이터 학습 John   | 개별 데이터 학습 Peter   |
|  |  |  |
| Train accuracy : 97.66%   | Train accuracy : 99.22%  | Train accuracy : 98.44%   |
| Test accuracy : 86.73%  | Test accuracy : 85.80%   | Test accuracy : 86.82%  |

csv\_test\_label : [1038, 966, 957, 992, 965, 997, 985, 1034, 1084, 982]

주어진 test dataset의 class별 분포를 확인해보면 약 1000개씩 분포된 IID dataset임을 알 수 있다. 따라서 test dataset에 대해서 좋은 performance를 보여주기 위해서는 학습에 사용된 dataset 또한 IID dataset이어야한다.

(1)의 과정에서 dataset1을 IID dataset이라고 판단하였는데 같은 hyper parameter를 사용해서 CL, FL 그리고 개별 데이터 학습으로 학습된 model로 test accuracy를 측정해본 결과 모두 준수한 성능을 보여주어서 dataset1이 IID dataset이라는 추측을 뒷받침해준다.

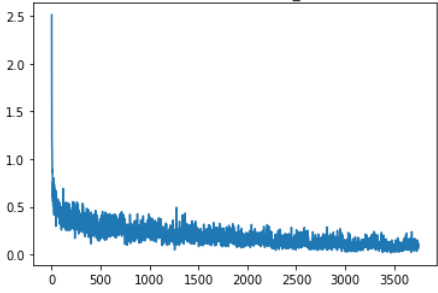
Centralized Learning의 경우 alice1, john1, peter1의 데이터셋을 하나로 합쳐서 학습에 사용하는데 각 class 별로 약 6000개의 data씩 거의 동일한 확률 분포로 주어지고 개별 데이터 학습은 alice1의 데이터만 사용하는 방식으로 학습하였는데 이 경우에도 (1)에서 분석하였듯이 class 별로 약 2000개의 data씩 거의 동일한 확률 분포로 주어지기 때문에 학습 과정에서 class imbalance로 인한 bias가 생기지 않아서 test dataset에 대해서 좋은 성능을 보여주었다.

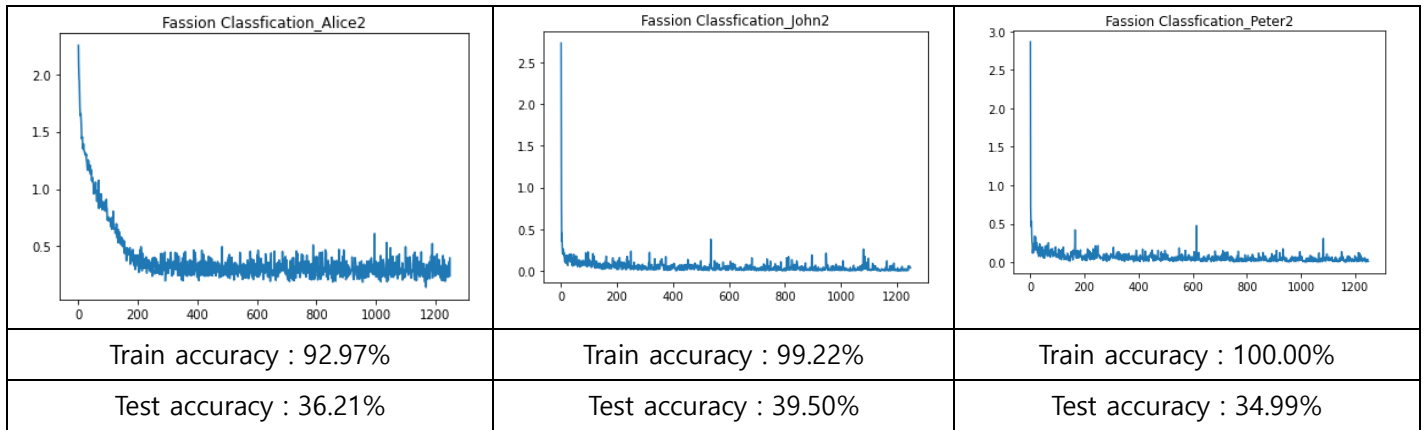
CL의 test accuracy는 약 90.07%인데 개별 데이터 학습의 test accuracy는 85~87%로 CL의 performance가 더 좋은 이유는 학습에 사용된 data sample이 더 많기 때문이다.

Federated learning의 경우 3명의 유저 alice1, john1, peter1의 dataset1이 IID dataset이기 때문에 client의 local update에 사용하는 mini-batch의 분포가 full training dataset의 분포와 동일하고 stochastic gradient에 큰 bias가 생기지 않아서 full gradient를 성공적으로 estimate할 수 있다.

따라서 FL의 test accuracy는 약 92.89%로 CL의 90.07%에 비교해서 부족하지 않은 performance를 보여주었다.

## - 데이터 셋 2에 대하여 CL, FL 그리고 개별 데이터 학습 결과 비교 및 분석

| CL  | FL                     |                 |
|---|------------------------|-----------------|
|  |                        |                 |
| Train accuracy : 96.88%   | Train accuracy : -     |                 |
| Test accuracy : 95.05%  | Test accuracy : 55.66% |                 |
| 개별 데이터 학습 Alice   | 개별 데이터 학습 John         | 개별 데이터 학습 Peter |



csv\_test\_label : [1038, 966, 957, 992, 965, 997, 985, 1034, 1084, 982]

주어진 test dataset의 class별 분포를 확인해보면 약 1000개씩 분포된 IID dataset임을 알 수 있다. 따라서 test dataset에 대해서 좋은 performance를 보여주기 위해서는 학습에 사용된 dataset 또한 IID dataset이어야한다.

(1)의 과정에서 dataset2를 non-IID dataset이라고 판단하였는데 같은 hyper parameter를 사용해서 CL, FL 그리고 개별 데이터 학습으로 학습된 model로 test accuracy를 측정해본 결과 CL은 준수한 성능을 보여주고 FL과 개별 데이터 학습의 경우 test accuracy가 약 56%, 35~40%으로 test dataset에 대한 prediction 결과가 좋지 않아서 dataset2이 non-IID dataset이라는 추측을 뒷받침 해준다.

반대로 3명의 유저 Alice, John, Peter에게 주어진 dataset2는 각각 10개의 class중 4개의 class의 data만 주어지고 나머지 6개의 class의 data는 주어지지 않아서 class별로 확률 분포가 일치하지 않는다. 따라서 dataset2는 non-IID dataset이라고 판별하였다.

Centralized Learning의 경우 3명의 유저들이 각각 10개의 class중 4개의 class의 data만 주어졌다 하더라도 alice2, john2, peter2의 데이터셋을 하나로 합치면 각 class 별로 약 6000개의 data씩 거의 동일한 확률 분포로 주어지기 때문에 CL은 학습 과정에서 class imbalance로 인한 bias가 생기지 않아서 test dataset에 대해서 약 95.05%의 test accuracy로 좋은 성능을 보여주었다.

Federated learning의 경우 3명의 유저 alice2, john2, peter2의 dataset2이 non-IID dataset이기 때문에 client의 local update에 사용하는 mini-batch의 분포가 full training dataset의 분포와 다르다. 따라서 stochastic gradient에 큰 bias가 생겨서 full gradient의 estimate으로 사용하기에는 poor estimate이기 때문에 test dataset에 대해서 약 55.66%의 test accuracy로 CL에 비해서 낮은 성능을 보여주었다.

개별 데이터 학습은 alice2의 데이터만 사용하는 방식으로 학습하였는데 이 경우에는 각 유저별로 10개의 class중 4개의 class의 data만 주어졌기 때문에 학습에 사용한 train dataset에 대해서는 약 98~100%의 train accuracy로 높은 performance를 보여주지만 10개의 class의 data를 모두 가지고

있는 test dataset에 대해서는 4개의 label로만 classify하기 때문에 모두 맞는다 하더라도 최대 40%의 정확도가 한계이다. 실제로도 약 35~40%의 test accuracy로 낮은 performance를 보여준다.

## **- IID와 non-IID 일 때 학습 성능의 차이 비교 및 분석 필요**

요약하자면 개별적으로 IID dataset인 데이터 셋 1에서 CL, FL, 개별 데이터 학습에 사용한 dataset은 모두 주어진 test dataset처럼 class 별로 data distribution이 고르기 때문에 학습된 모델은 좋은 performance를 보여주었다.

개별적으로 non-IID dataset인 데이터 셋 2의 경우에는 CL에 사용한 dataset은 IID dataset이기 때문에 학습된 모델은 좋은 performance를 보여주었지만 class 별로 data distribution이 고르지 않은 dataset을 사용한 FL과 개별 데이터 학습의 경우 낮은 performance를 보여주었다.

데이터 셋 2에서 FL의 test accuracy가 약 56%으로 개별 데이터 학습의 35~40%보다 높은 성능을 보여주는 이유는 epoch마다 FedAvg 방식으로 model parameter를 update하게 되면서 다른 client가 보유하고 있는 data도 간접적으로 사용해서 local model을 improve하는데 사용할 수 있었기 때문이다.

## **(3) FL을 실제 환경에 사용하려고 할 때 추가로 고려해야 하는 내용 서술**

실제 환경에서 FL을 사용하려고 할 때 사용자 device의 local data가 IID data일 것이라고 기대하기는 힘들다.

예를 들어서 사용자가 동물에 대해서 사진을 찍은 경우 일상 생활에서 흔히 볼 수 있는 개나 고양이의 사진은 수백 장의 서로 다른 사진을 찍을 수 있지만 사자나 코끼리에 대한 사진은 특수한 경우가 아닌 이상 찍을 수 없을 것이다.

이럴 경우 data의 identicalness에 violation이 일어나서 client의 local update 과정에서 bias가 발생하고 Federated learning의 학습 성능이 떨어지게 된다.

이번 프로젝트에서는 non-IID dataset2에 대해서 모든 사용자의 data를 합쳐서 Centralized learning을 수행하면 IID dataset이 되기 때문에 이를 해결할 수 있었다.

Federated learning에 이를 적용하기 위해서는 사용자끼리 data를 주고받으면 된다. 그러나 이는 client의 local data에 대해서 data privacy를 보장해준다는 Federated Learning의 강점과 전면 충돌한다.

실제 상황에서 이를 적용하기 위해서는 central server가 미리 가지고 있던 data를 모든 사용자에게 일부 배분하여서 data privacy를 침해하지 않으면서도 non-IID local data로 인한 문제를 완화할 수 있을 것이다.

```
csv_label_alice2 : [5978, 5985, 6023, 0, 0, 0, 0, 0, 0, 2014]
csv_label_john2  : [0, 0, 0, 0, 0, 0, 5963, 5961, 6065, 2011]
csv_label_peter2 : [0, 0, 0, 6067, 5979, 5983, 0, 0, 0, 1971]
```

예를 들어서 이번 프로젝트에서 dataset2의 alice2의 local data label 4-9에 해당하는 data를 가지고 있지 않기 때문에 central server에서 label 4-9에 해당하는 data를 Alice에게 보내주고 data augmentation 같은 방법으로 모든 class 별로 약 6000개의 data를 가지게 만들면 class imbalance로 인한 IID-data 문제를 해결할 수 있다.

그러나 이 경우에는 communication cost가 증가한다는 문제가 있다. 또한 central server에서 미리 환경에 걸맞는 dataset을 보유하고 있을 필요가 생기는데 data를 obtain하고 label하는 과정은 매우 expensive하기 때문에 새로운 비용이 발생한다.

또다른 방법은 Federated Learning의 학습 epochs를 늘리는 방법이다.

이번 프로젝트를 진행하면서 non-IID dataset2의 경우 Federated Learning으로 학습하는 경우 첫 epoch에서는 약 30%에 불과했던 test accuracy가 epoch가 늘어날수록 증가해서 10 epochs의 학습을 진행한 후에는 약 56%까지 증가하였다.

이는 epoch마다 FedAvg 방식으로 model parameter를 update하게 되면서 다른 client가 보유하고 있는 data도 간접적으로 사용해서 local model을 improve하는데 사용할 수 있게 되기 때문이다.

Epoch가 늘어나면서 발생하는 communication cost는 model-compression과 decentralized training으로 완화할 수 있다.

실제상황에서는 client마다 서로 다른 hardware를 사용하기 때문에 computing power와 network 환경의 차이로 인한 communication speed의 차이와 communication error rate로 인한 문제도 고려해야 한다.

Communication speed의 차이로 인한 문제를 해결하기 위해서 asynchronous communication 방식으로 일정하지 않은 interval로 update를 해서 느린 client의 기준에 맞춰 줄 수 있다. 또한 communication error가 발생한 경우 error가 발생한 device는 무시하고 model을 update하는 fault tolerance 방식으로 이를 해결할 수 있다.



#### 4. Reference

- 1) *Learning from Data: A Short Course*, Yaser S. Abu-Mostafa, AMLbook
- 2) *Federated Learning : Collaborative Machine Learning without Centralized Training Data*, Brendan McMahan and Daniel Ramage. 2017