FL Open-Source Platform Overview

김진수

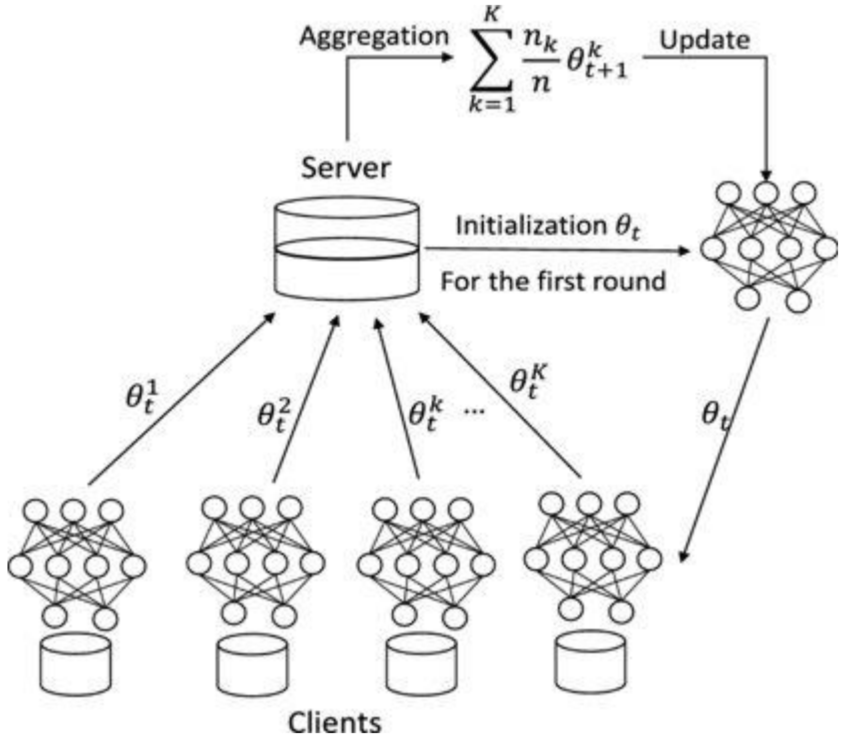
2022-09-15

강사 소개

- 김진수
- Tvstorm AI 개발팀 연구원
- 주요 연구 분야 : 연합학습, rPPG

- Email: wlstn25092303@gmail.com

Federated Learning



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 $t = 1, 2, \ldots$ **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**
      $w_{t+1}^k \leftarrow$ 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

Federated Learning: Collaborative Machine Learning without Centralized Training Data(Google AI Blog)

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

FL Open-Source Framework

- Flower
- FedScale
- PySyft
- FedML
- TFF

FL Open-Source Framework

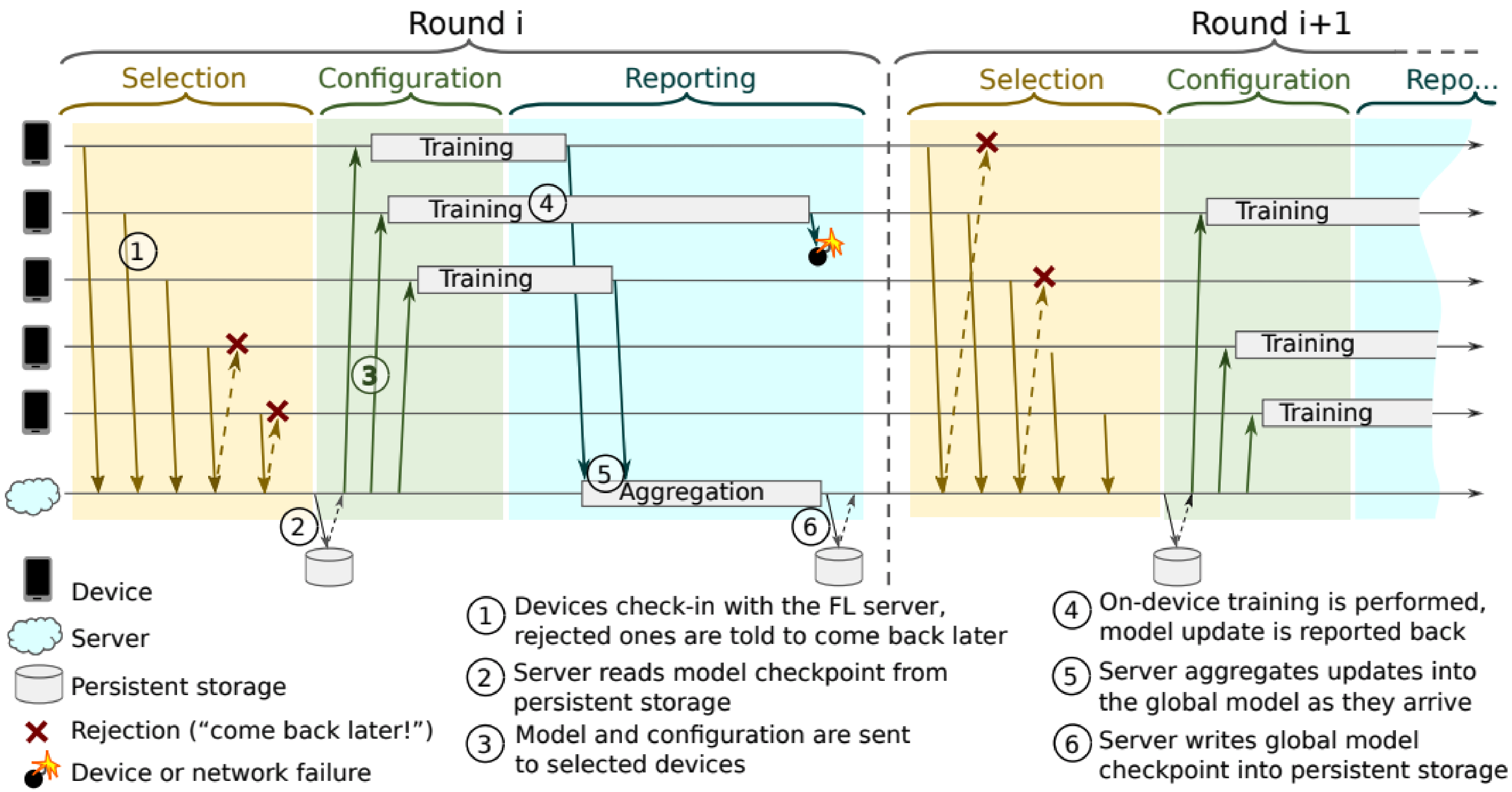


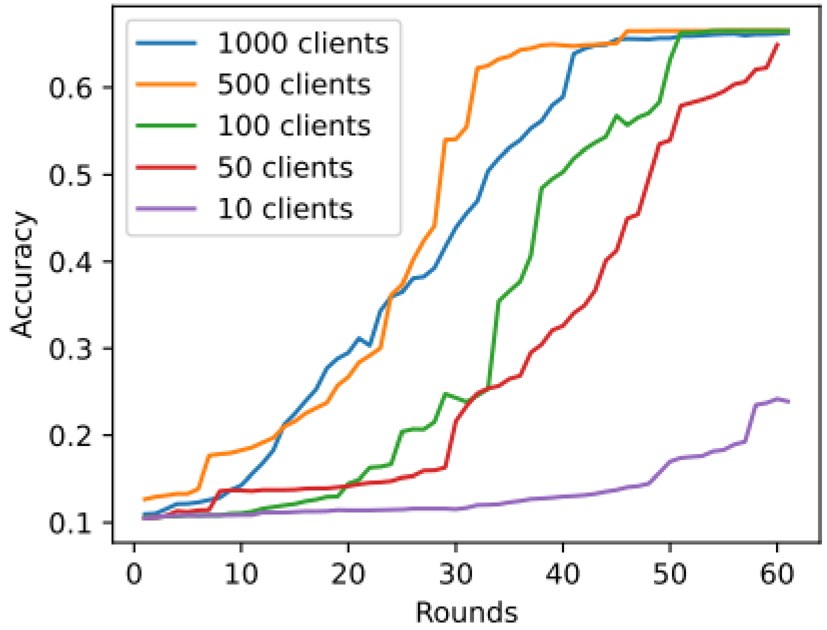
Figure 1: Federated Learning Protocol

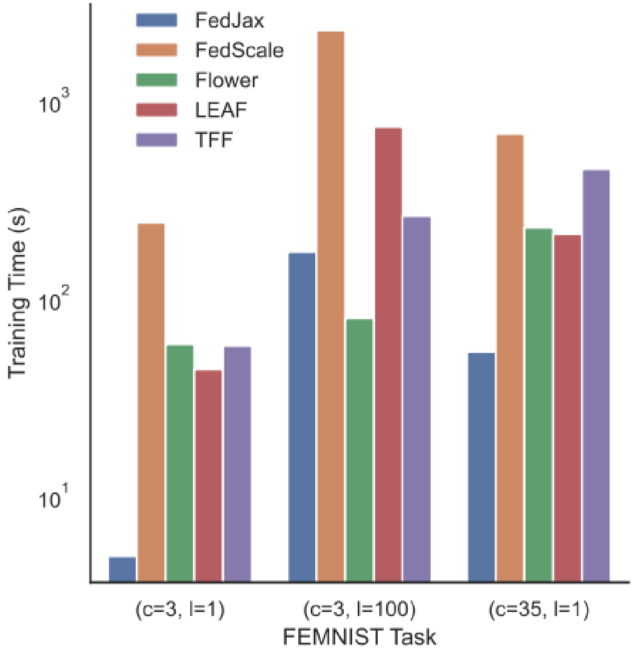
Flower

- 개발 언어나 ML 프레임워크에 구애 받지 않는다.
- 사용성(쉽게 사용 가능, 직관적)
- 확장성(모바일, 대규모 실험)

Table 1. Excerpt of built-in FL algorithms available in Flower. New algorithms can be implemented using the *Strategy* interface.

Strategy	Description
FedAvg	Vanilla Federated Averaging (McMahan et al., 2017)
Fault Tolerant FedAvg	A variant of FedAvg that can tolerate faulty client conditions such as client disconnections or laggards.
FedProx	Implementation of the algorithm proposed by Li et al. (2020) to extend FL to heterogenous network conditions.
QFedAvg	Implementation of the algorithm proposed by Li et al. (2019) to encourage fairness in FL.
FedOptim	A family of server-side optimizations that include FedAdagrad, FedYogi, and FedAdam as described in Reddi et al. (2021).





FedScale

- 포괄적이고 현실적인 데이터셋 제공
- 자동화된 평가 플랫폼 FedScale Runtime 제공
- Benchmarks에 집중

Category	Name	Data Type	#Clients	#Instances	Example Task
CV	*OpenImage*	Image	13,771	1.3M	Classification, Object detection
	Google Landmark	Image	43,484	3.6M	Classification
	Charades	Video	266	10K	Action recognition
	VLOG	Video	4,900	9.6K	Classification, Object detection
	Waymo Motion	Video	496,358	32.5M	Motion prediction
NLP	*Europarl*	Text	27,835	1.2M	Text translation
	Reddit	Text	1,660,820	351M	Word prediction
	LibriTTS	Text	2,456	37K	Text to speech
	Google Speech	Audio	2,618	105K	Speech recognition
	Common Voice	Audio	12,976	1.1M	Speech recognition
Misc ML	*Taobao*	Text	182,806	20.9M	Recommendation
	Puffer Streaming	Text	121,551	15.4M	Sequence prediction
	Fox Go	Text	150,333	4.9M	Reinforcement learning

```
import flwr as fl

def get_config_fn():
  # Implementation of randomly selection
  client_ids = random_selection()
  config = {"ids": client_ids}
  return config

# Customized Strategy
strategy = CustomizedStrategy(
  on_fit_config_fn=get_config_fn())

fl.server.start_server(
  config={"num_rounds":args.round},
  strategy=strategy)
```

```
import flwr as fl

class Customized_Client():
  def fit(self, config, net):
    # Customization of client data
    trainloader = select_dataset(
      config["ids"][args.partition])
    train(net, trainloader)
    compressed_result = self.get_parameters()
    # Implementation of compression
    compressed_result = compress_impl(
        training_result)
    return compressed_result

fl.client.start_numpy_client(
  args.address, client=CustomizedClient())
```

Comparison Table

	TFF	Syft	FedScale	LEAF	Flower
Single-node simulation	√	√	√	√	√
Multi-node execution	*	√	(√)***		√
Scalability	*		**		√
Heterogeneous clients		(√)***	**		√
ML framework-agnostic		****	****		√
Communication-agnostic					√
Language-agnostic					√
Baselines			√	√	*

Labels: * Planned / ** Only simulated
*** Only Python-based / **** Only PyTorch and/or TF/Keras

Beutel, Daniel J., et al. "Flower: A friendly federated learning research framework." *arXiv preprint arXiv:2007.14390* (2020).

Features	LEAF	TFF	FedML	Flower	**FedScale**
Heter. Client Dataset	○	✗	○	○	✔
Heter. System Speed	✗	✗	○	○	✔
Client Availability	✗	✗	✗	✗	✔
Scalable Platform	✗	✔	○	✔	✔
Real FL Runtime	✗	✗	✗	✗	✔
Flexible APIs	✗	✔	✔	✔	✔

Table 1. Comparing FedScale with existing FL benchmarks and libraries. ○ implies limited support.

Lai, Fan, et al. "FedScale: Benchmarking model and system performance of federated learning at scale." *International Conference on Machine Learning*. PMLR, 2022.

실습

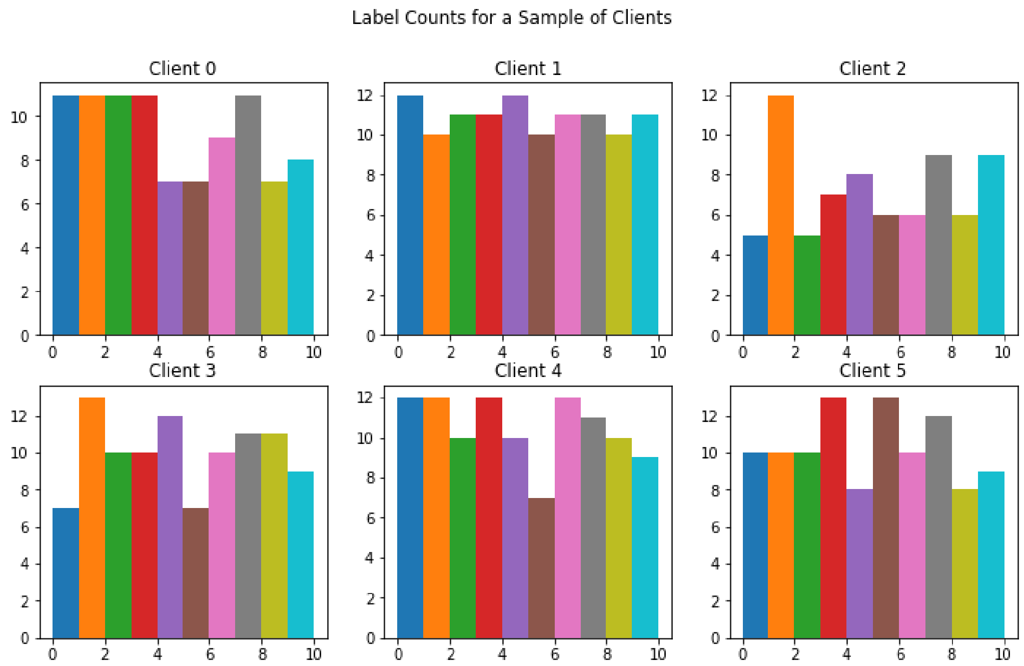
- Installing Flower
- Centralized to Federated
- Femnist

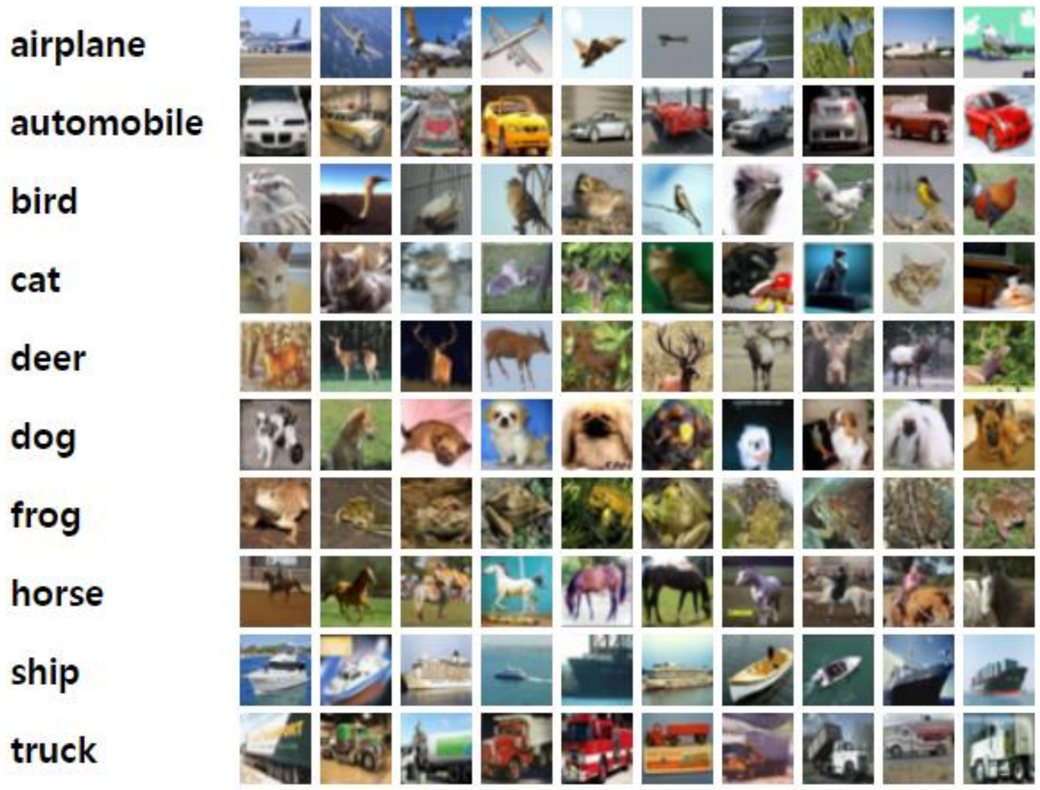
Installing Flower

- Pip
- Flower
- Pytorch
- Python(>=3.7)
- Code : https://github.com/jinsoogod/fed_flower

실습 데이터

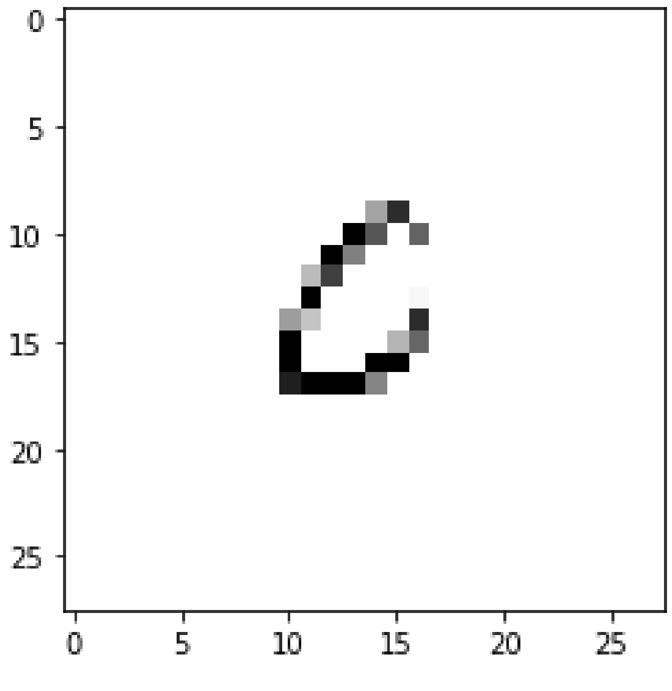
- Centralized to Federated
- Size 32x32
- 60000개 컬러 이미지 데이터(num_class =10)

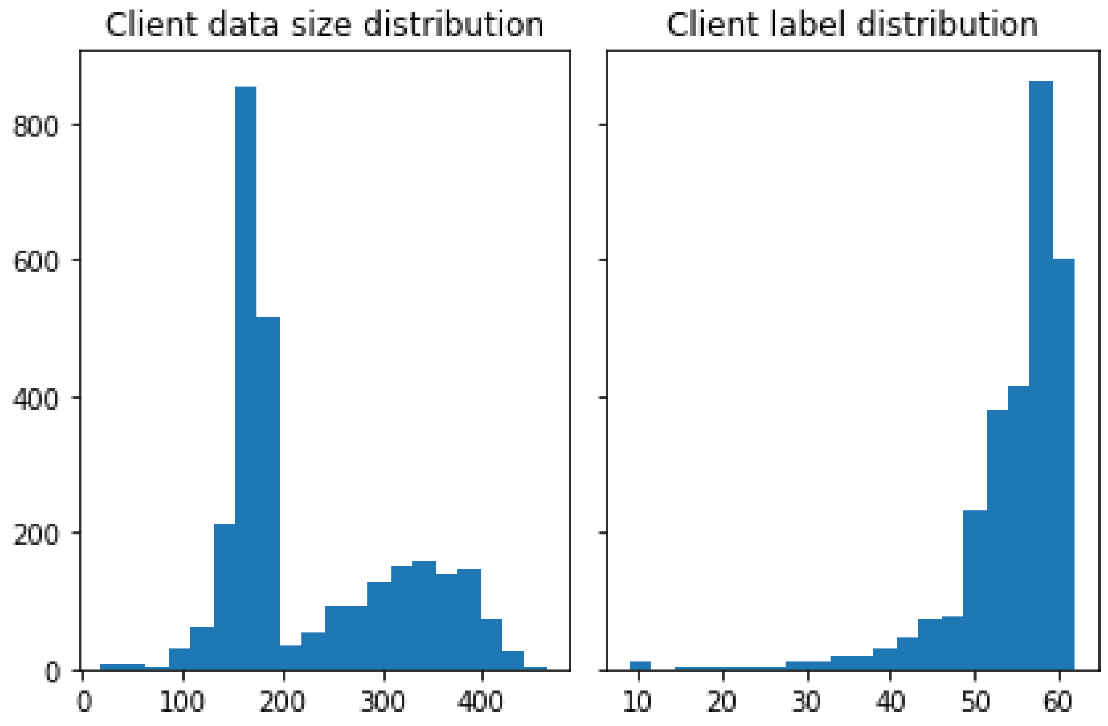




실습 데이터

- FEMNIST – 필기체 데이터(숫자, 소문자, 대문자)
- Size 28x28
- Total number of data samples: 637877(num_class = 62)





Centralized Training

- Model
- Load Data
- Train
- Test

Centralized Training

- Model

```python
class Net(nn.Module):

    def __init__(self) -> None:
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x: Tensor) -> Tensor:
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

Centralized Training

- Load Data

```
def load_data():
    """Load CIFAR-10 (training and test set)."""
    transform = transforms.Compose(
        [transforms.ToTensor(), transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]
    )
    trainset = CIFAR10("./data", train=True, download=True, transform=transform)
    trainloader = torch.utils.data.DataLoader(trainset, batch_size=32, shuffle=True)
    testset = CIFAR10("./data", train=False, download=True, transform=transform)
    testloader = torch.utils.data.DataLoader(testset, batch_size=32, shuffle=False)
    num_examples = {"trainset" : len(trainset), "testset" : len(testset)}
    return trainloader, testloader, num_examples
```

Centralized Training

- Train

```python
def train(net, trainloader, device, epochs):
    """Train the network."""
    # Define loss and optimizer
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.SGD(net.parameters(), lr=0.001, momentum=0.9)

    print(f"Training {epochs} epoch(s) w/ {len(trainloader)} batches each")

    # Train the network
    for epoch in range(epochs):  # loop over the dataset multiple times
        running_loss = 0.0
        for i, data in enumerate(trainloader, 0):
            images, labels = data[0].to(device), data[1].to(device)

            # zero the parameter gradients
            optimizer.zero_grad()

            # forward + backward + optimize
            outputs = net(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            # print statistics
            running_loss += loss.item()
            if i % 100 == 99:  # print every 100 mini-batches
                print("[%d, %5d] loss: %.3f" % (epoch + 1, i + 1, running_loss / 2000))
                running_loss = 0.0
```

Centralized Training

- Test

```
def test(net, testloader, device):
    """Validate the network on the entire test set."""
    criterion = nn.CrossEntropyLoss()
    correct = 0
    total = 0
    loss = 0.0
    with torch.no_grad():
        for data in testloader:
            images, labels = data[0].to(device), data[1].to(device)
            outputs = net(images)
            loss += criterion(outputs, labels).item()
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
    accuracy = correct / total
    return loss, accuracy
```

Federated Training

- Server

```
# Define strategy
strategy = fl.server.strategy.FedAvg(evaluate_metrics_aggregation_fn=weighted_average)

# Start Flower server
fl.server.start_server(
    server_address="0.0.0.0:8080",
    config=fl.server.ServerConfig(num_rounds=3),
    strategy=strategy,
)
```

Federated Training

- Client
  - Model
  - Load Data
  - Train
  - Test
  - FlowerClient
    - get_parameter
    - set_parameter
    - fit
    - evaluate

Federated Training

- FlowerClient
  - get_parameter

```python
def get_parameters(self, config):
    return [val.cpu().numpy() for _, val in net.state_dict().items()]
```

  - set_parameter

```python
def set_parameters(self, parameters):
    params_dict = zip(net.state_dict().keys(), parameters)
    state_dict = OrderedDict({k: torch.tensor(v) for k, v in params_dict})
    net.load_state_dict(state_dict, strict=True)
```

Federated Training

- FlowerClient
  - fit

```python
def fit(self, parameters, config):
    self.set_parameters(parameters)
    train(net, trainloader, epochs=1)
    return self.get_parameters(config={}), len(trainloader.dataset), {}
```

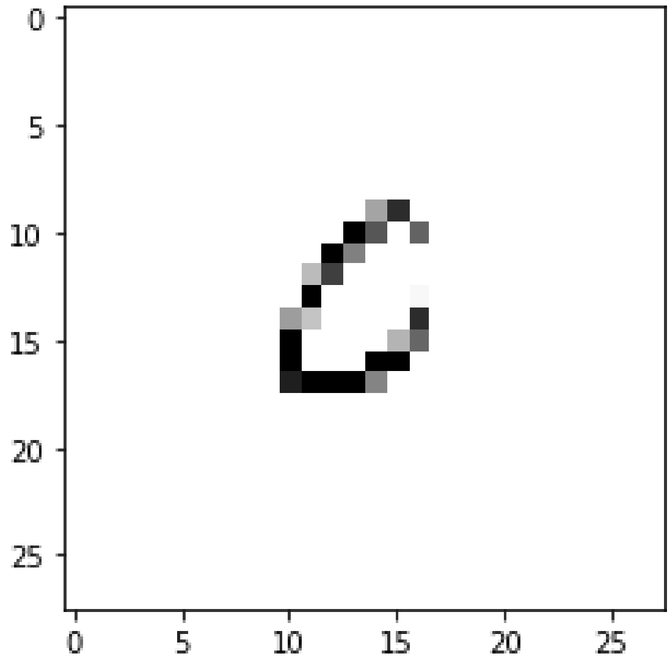
  - evaluate

```python
def evaluate(self, parameters, config):
    self.set_parameters(parameters)
    loss, accuracy = test(net, testloader)
    return loss, len(testloader.dataset), {"accuracy": accuracy}
```

FEMNIST

- Leaf: https://github.com/TalwalkarLab/leaf/tree/master/data/femnist
- Realistic heterogeneous 구성을 위한 데이터 분할

```
def load_data():
    """Load CIFAR-10 (training and test set)."""
    transform = transforms.Compose(
        [transforms.ToTensor()]
    )
    number = random.randint(0, 35)
    if number == 35:
        subject_number = random.randint(0, 96)
    else:
        subject_number = random.randint(0, 99)
    print('number : {}, subject number : {}'.format(number, subject_number))
    with open("./data/data/train/all_data_"+str(number)+"_niid_0_keep_0_train_9.json","r") as f:
        train_json = json.load(f)
    with open("./data/data/test/all_data_"+str(number)+"_niid_0_keep_0_test_9.json","r") as f:
        test_json = json.load(f)
    train_user = train_json['users'][subject_number]
    train_data = train_json['user_data'][train_user]
    test_user = test_json['users'][subject_number]
    test_data = test_json['user_data'][test_user]
    trainset = FemnistDataset(train_data, transform)
    testset = FemnistDataset(test_data, transform)
    trainloader = DataLoader(trainset, batch_size=64, shuffle=True)
    testloader = DataLoader(testset, batch_size=64)
    return trainloader, testloader
```



FEMNIST

- Leaf: https://github.com/TalwalkarLab/leaf/tree/master/data/femnist
- 데이터셋 링크 : data.zip