

1. ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ ФРЕЙМВОРКА

1.1 Федеративные режимы

Фреймворк имеет несколько платформ федеративного обучения:

- **FedML Octopus** предназначена для кросс-сило
- **FedML Parrot** предназначена для симуляции федеративного обучения и проведении быстрых экспериментов
- **FedML Beehive** предназначена для кросс-девайс
- **FedML Spider** предназначена для обучения на приватных данных через браузер
- **FedML Cheetah** предназначена для обучения в облаке

1.2 Типы данных

Фреймворк работает с изображениями, текстом на естественном языке, табличными данными.

В FedML есть уже загруженные датасеты, по многим из них на github есть примеры обучения. На github перечислены следующие встроенные датасеты:

Компьютерное зрение (изображения)

- MNIST
- cifar10
- cifar100
- fed_cifar100
- fed_emnist
- cinic10
- ImageNet
- Landmarks

Обработка естественного языка

- shakespeare
- fed_shakespeare
- stackoverflow

Автономное вождение (распознавание объектов)

- KITTI
- CityScapes
- Waymo
- Lyft Level 5
- UCB BDD100K
- ArgoVerse
- nuScenes

Финансовые данные (табличная форма)

- lending_club_loan

Другие

- NUS_WIDE (изображения)
- UCI (изображения)
- Synthetic
- edge_case_examples

1.3 Модели анализа данных

Из классических моделей машинного обучения во фреймворке есть только линейная регрессия.

Что касается нейронных сетей, на гитхабе есть простые примеры с использованием CNN (сверточная нейронная сеть), RNN(рекуррентная), MobileNet, Resnet56, Resnet20, Generating adversarial networks, DART.

Так же на гитхабе есть примеры приложений , разработанных на платформе FedML.

Для компьютерного зрения есть модели:

1. Классификация изображений:
 - CNN
 - DenseNet
 - MobileNetv3
 - EfficientNet
2. Сегментация изображений
 - UNet
 - DeeplabV3

- TransUnet
3. Распознавание объектов
- YOLOv5

Для обработки естественного языка:

- Классификация текстов
- Распознавание команд (seq tagging, seq2seq)
- Извлечение фрагментов текста (span extraction)

Федеративное обучение для графовых нейронных сетей:

1. На уровне графов (Graph-Level)
 - i. Предсказание свойств молекул (MoleculeNet Property Prediction)
 - Классификация графов
 - Регрессия графов
 - ii. Социальные сети (Social Networks)
 - Классификация графов
2. На уровне подграфов (Subgraph-Level)
 - i. Рекомендательные системы (Recommendation Systems)
 - Предсказание связей (Link Prediction)
3. На уровне узлов (Node-Level)
 - i. Сети Ego(Citation & Coauthor Networks)
 - Предсказание связей
 - Классификация узлов

1.4 Функции агрегирования

Согласно константам на гитхабе, доступны следующие алгоритмы федеративного обучения: FedAvg, FedOpt, FedProx, classical_vertical, split nn, decentralized FL, FedGan, FedAvg robust, FedAvg seq, FedOpt seq, FedGKT, FedNAS, FedSeg, turbo_aggregate, FedNova, FedDyn, SCAFFOLD, Mime, Hierarchical FL, FedSGD, Fed Local SGD, Async FedAvg.

Больше всего примеров есть с FedAvg, остальные в основном использовались только в режиме симуляции.

1.5 Защита конфиденциальности

Есть дифференциальная приватность (механизм Гаусса и Лапласа). Так же есть информация о проведении тестов со следующими типами атак и механизмами защиты:

Attack

1. ByzantineAttack: (1) zero mode (2) random mode (3) flip mode
2. (NeurIPS 2019) DLGAttack: "Deep leakage from gradients" <https://proceedings.neurips.cc/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf>
3. (NeurIPS 2020) InvertAttack: "Inverting gradients-how easy is it to break privacy in federated learning?" <https://github.com/JonasGeiping/invertinggradients/>
4. LabelFlippingAttack: "Data Poisoning Attacks Against Federated Learning Systems" <https://arxiv.org/pdf/2007.08432>
5. (NeurIPS 2021) RevealingLabelsFromGradientsAttack: "Revealing and Protecting Labels in Distributed Training" <https://proceedings.neurips.cc/paper/2021/file/0d924f0e6b3fd0d91074c22727a53966-Paper.pdf>
6. (NeurIPS 2019) BackdoorAttack: "A Little Is Enough: Circumventing Defenses For Distributed Learning" <https://proceedings.neurips.cc/paper/2019/file/ec1c59141046cd1866bbcbdfb6ae31d4-Paper.pdf>
7. (NeurIPS 2020) EdgeCaseBackdoorAttack: "Attack of the Tails: Yes, You Really Can Backdoor Federated Learning" <https://proceedings.neurips.cc/paper/2020/file/b8ffa41d4e492f0fad2f13e29e1762eb-Paper.pdf>
8. (PMLR'20) ModelReplacementBackdoorAttack: "How To Backdoor Federated Learning" <http://proceedings.mlr.press/v108/bagdasaryan20a/bagdasaryan20a.pdf>

Рисунок 1 – Типы атак

Defense

1. (PMLR 2018) BulyanDefense: "The Hidden Vulnerability of Distributed Learning in Byzantium." <http://proceedings.mlr.press/v80/mhamdi18a/mhamdi18a.pdf>
2. CClipDefense: "Byzantine-Robust Learning on Heterogeneous Datasets via Bucketing" <https://arxiv.org/pdf/2006.09365.pdf>
3. GeometricMedianDefense: "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent." <https://dl.acm.org/doi/pdf/10.1145/3154503>
4. (NeurIPS 2017) KrumDefense: "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent" <https://papers.nips.cc/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf>
5. (ICLR 2021) MultiKrumDefense: "Distributed momentum for byzantine-resilient stochastic gradient descent" <https://infoscience.epfl.ch/record/287261>
6. NormDiffClippingDefense: "Can You Really Backdoor Federated Learning?" <https://arxiv.org/pdf/1911.07963.pdf>
7. (AAAI 2021) RobustLearningRateDefense: "Defending against backdoors in federated learning with robust learning rate." <https://github.com/TinfoilHat0/Defending-Against-Backdoors-with-Robust-Learning-Rate>
8. SoteriaDefense: "Provable defense against privacy leakage in federated learning from representation perspective." <https://arxiv.org/pdf/2012.06043>
9. SLSGDDefense: "SLSGD: Secure and efficient distributed on-device machine learning" <https://arxiv.org/pdf/1903.06996.pdf>
10. RFA_defense: "Robust Aggregation for Federated Learning" <https://arxiv.org/pdf/1912.13445>
11. (USENIX2020) FoolsGoldDefense: "The Limitations of Federated Learning in Sybil Settings" <https://www.usenix.org/system/files/raid20-fung.pdf>
12. (ICML 2018) CoordinateWiseMedianDefense: "Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates" <http://proceedings.mlr.press/v80/yin18a/yin18a.pdf>
13. (NeurIPS 2021) WbcDefense: "Enhancing Robustness against Model Poisoning Attacks in Federated Learning from a Client Perspective" <https://arxiv.org/abs/2110.13864>
14. (ICML 2021) CRFLDefense: "CRFL: Certifiably Robust Federated Learning against Backdoor Attacks" <http://proceedings.mlr.press/v139/xie21a/xie21a.pdf>

Рисунок 2 – Реализованные механизмы защиты

1.6 Распределения данных

Согласно константам на гитхабе, для кросс-сило поддерживается только горизонтальное и иерархическое распределение.

Для режима симуляции поддерживается так же централизованное, децентрализованное, вертикальное и сплит распределение.

FedML Parrot supports representative algorithms in different communication topologies (as the figure shown below), including Fedvrg, FedOpt (ICLR 2021), FedNova (NeurIPS 2020), FedGKT (NeurIPS 2020), Decentralized FL, Vertical FL, Hierarchical FL, FedNAS, and Split Learning.

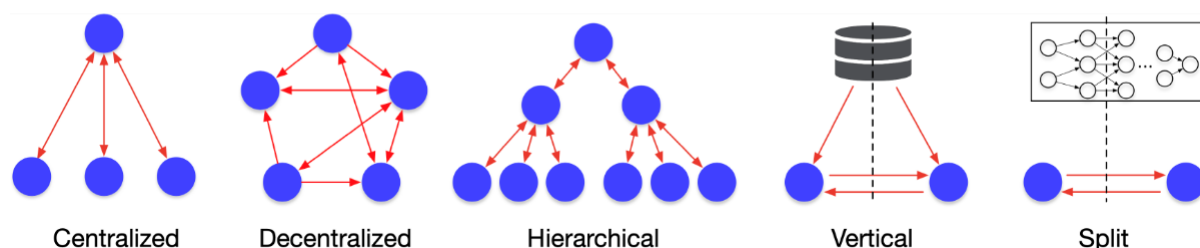


Рисунок 3 – Распределения для симуляционного режима

1.7 Метрики

Для простых примеров на встроенных датасетах рассчитывается только accuracy и loss, расчет метрик находится в файле [my_server_aggregator_prediction.py](#). Судя по объявлению словаря с метриками, планировалось так же реализовать расчет precision и recall, но пока их нет.

Для некоторых продвинутых приложений, примеры которых находятся в папке app, прописывается расчет других метрик. Например, precision, recall, confusion matrix. ([Пример расчета метрик для модели компьютерного зрения](#))

1.8 Простота настройки

Документация скудная. Нет описаний функций, только общие описания архитектуры, способов применения. Информацию о данных, моделях, безопасности, метриках приходится искать по разным папкам на гитхабе.

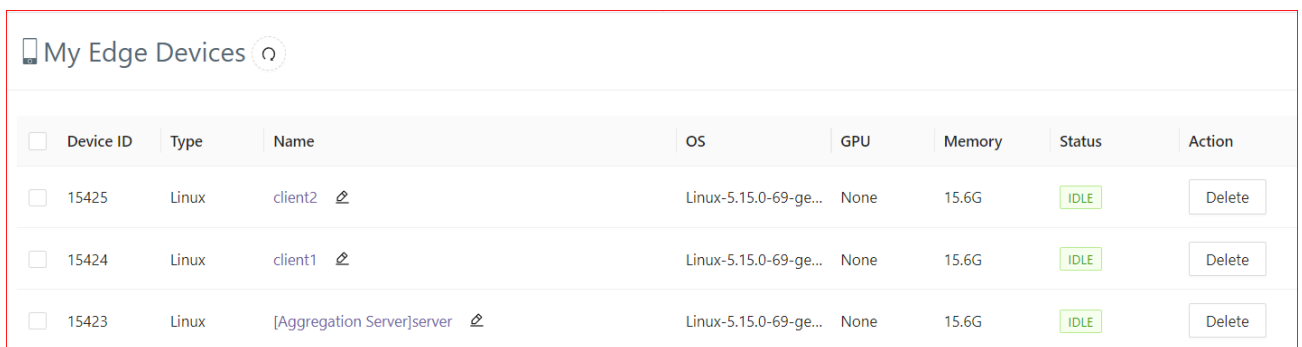
Настройка обучения сама по себе удобная и несложная, но найти информацию о настройке очень сложно. По возможным проблемам и ошибкам в процессе настройки так же нет никакой информации. Коммьюнити не активное.

2. ЭКСПЕРИМЕНТЫ

2.1 Запуск примера

Была проведена настройка платформы FedML для федеративного обучения и запущен пример, согласно следующим шагам:

1. На виртуальные машины необходимо установить FedML:
`$ pip install fedml`
2. Нужно зарегистрироваться на сайте FedML для доступа к MLOps
<https://open.fedml.ai/octopus/profile>
3. Прикрепить машины к MLOps с помощью команды (Account ID указан в профиле):
`$ fedml login < Account ID >`
4. Чтобы подключить машину в качестве сервера:
`$ fedml login < Account ID > -s`
5. Проверить подключение машин можно во вкладке Edge device. Тут же можно переименовать машины для удобства



<input type="checkbox"/>	Device ID	Type	Name	OS	GPU	Memory	Status	Action
<input type="checkbox"/>	15425	Linux	client2 🔗	Linux-5.15.0-69-ge...	None	15.6G	IDLE	Delete
<input type="checkbox"/>	15424	Linux	client1 🔗	Linux-5.15.0-69-ge...	None	15.6G	IDLE	Delete
<input type="checkbox"/>	15423	Linux	[Aggregation Server]server 🔗	Linux-5.15.0-69-ge...	None	15.6G	IDLE	Delete

Рисунок 4 – Подключенные устройства

6. На любой из машин создать следующую структуру проекта ([пример](#)):

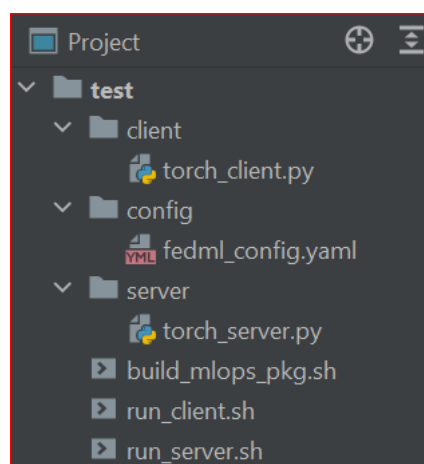


Рисунок 5 – Структура проекта

7. В файле указать необходимую конфигурацию: количество клиентов, датасет, распределение и т.д.

8. Запустить генерацию zip-файлов для MLOps:

```
$ bash build_mlops_pkg.sh
```

9. Скачать сгенерированные файлы на локальный компьютер. Скачать файлы с виртуальной машины на Windows можно следующей командой (виртуальная машина должна быть запущена):

```
pscp [имя пользователя]@[имя сервера/ip-адрес]:[путь к файлу в Linux]  
[путь к файлу в Windows]
```

```
C:\Users\User>pscp etu@158.160.44.128:/home/etu/test/mlops/dist-packages/*.zip .  
server-package.zip      | 2 kB | 2.3 kB/s | ETA: 00:00:00 | 100%  
client-package.zip      | 2 kB | 2.3 kB/s | ETA: 00:00:00 | 100%
```

Рисунок 6 – Скачивание файлов на Windows

10. Создать новый application в MLOps, прикрепив сгенерированные zip-файлы

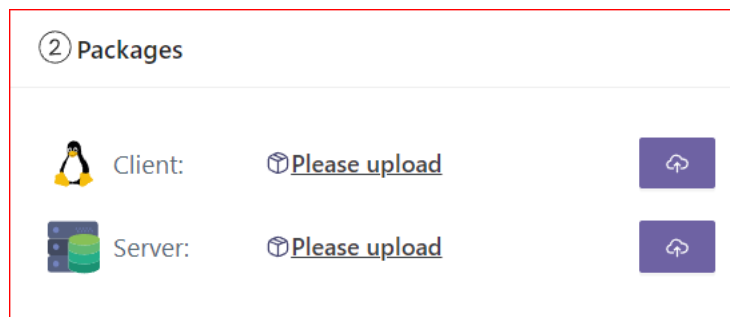


Рисунок 7 – Прикрепление zip-файлов

11. Создать новый проект

12. Зайти в проект и создать новый запуск, выбрав необходимые устройства и указав созданный ранее application. Так же здесь, если нужно, можно будет изменить конфигурацию

13. После этого начнется обучение, когда завершатся все раунды и устройства получат статус Finished, можно будет посмотреть результаты обучения в соответствующих вкладках.

14. Если возникает проблема, что статусы клиентов отображаются как Queued и обучение не начинается, можно попробовать удалить устройства на вкладке Edge device, присоединить их заново, а потом начать новый запуск

Run OverviewTraining StatusResultsSystemModelsLogs

Status

Training Progress: 10/10 rounds
Training Time: 0h 0m 46s

Server ID	Type	Name	OS	GPU	CPU	Memory	Status	Logs
15423	Linux	server	Linux-5.15.0-6...	None	x86_64	15.6G	FINISHED	Detail

1

Device ID	Type	Name	OS	GPU	CPU	Memory	Status	Logs
15424	Linux	client1	Linux-5.15.0-6...	None	x86_64	15.6G	FINISHED	Detail
15425	Linux	client2	Linux-5.15.0-6...	None	x86_64	15.6G	FINISHED	Detail

1

Рисунок 8 – Результаты обучения

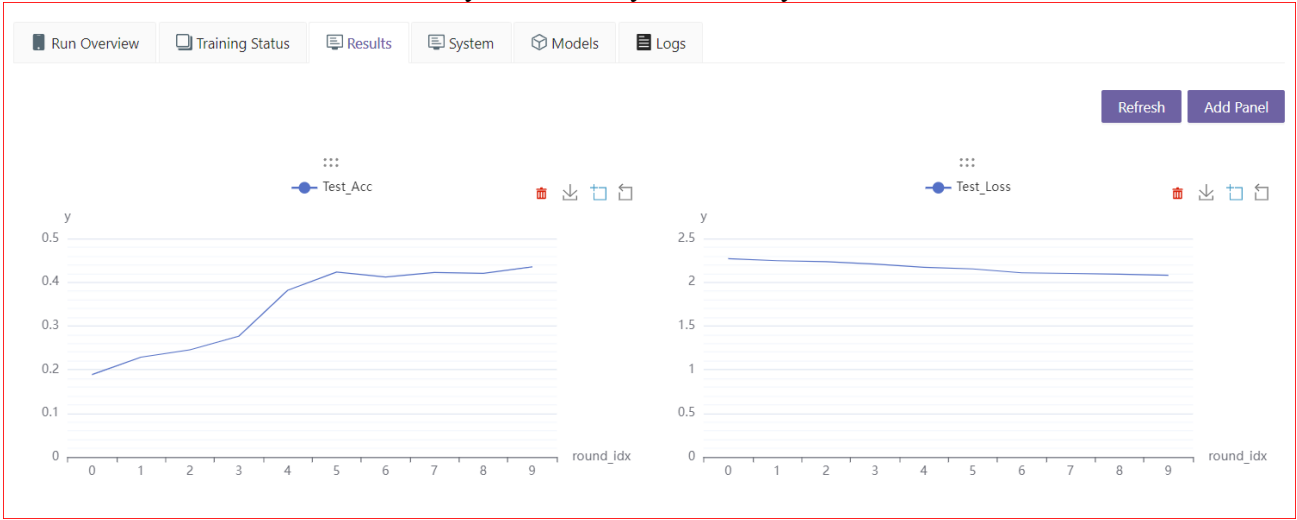


Рисунок 9 – Результаты обучения. Метрики

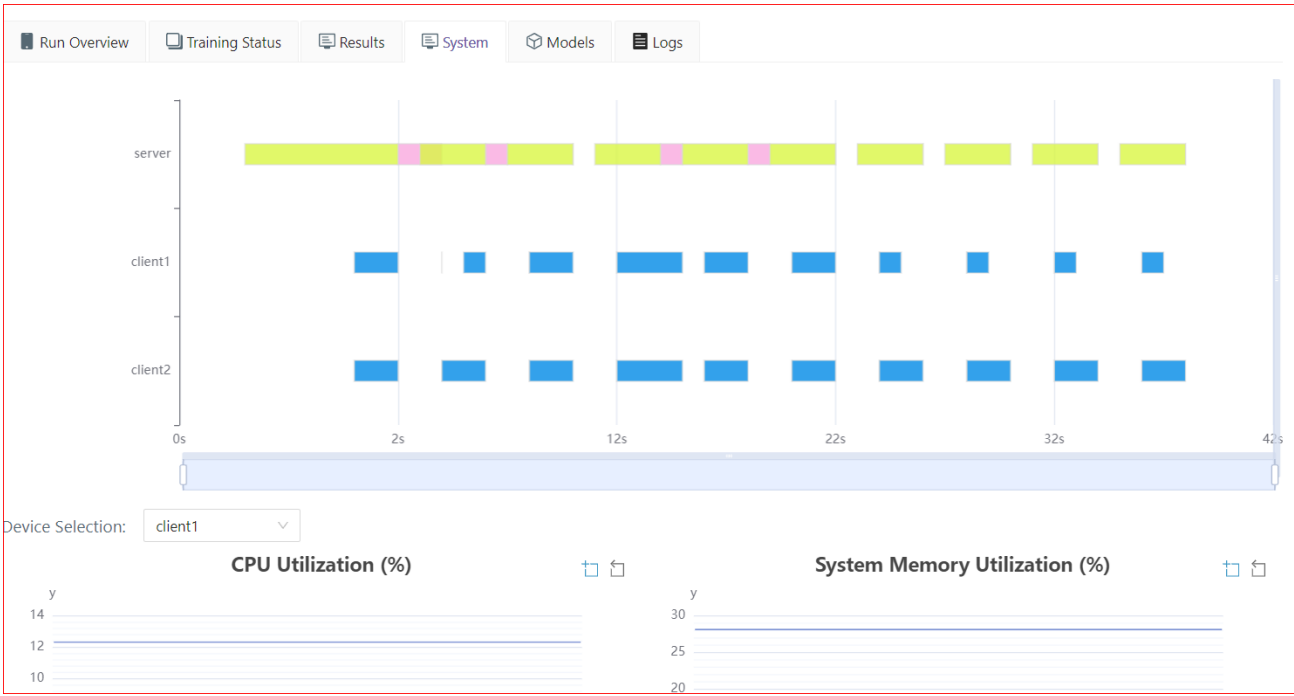


Рисунок 10 – Результаты обучения. Система

The screenshot displays the 'Models' tab in the FedML interface. It shows the 'Aggregated Model' section. Below the navigation bar, there are tabs for 'Aggregated Model' and 'Client Model'. The main area contains a table with the following data:

















RunId	Round_idx	CreateTime	Action
8567	10	05-03-2023	Download Create Model Card Update Model
8567	9	05-03-2023	Download Create Model Card Update Model
8567	8	05-03-2023	Download Create Model Card Update Model
8567	7	05-03-2023	Download Create Model Card Update Model
8567	6	05-03-2023	Download Create Model Card Update Model
8567	5	05-03-2023	Download Create Model Card Update Model
8567	4	05-03-2023	Download Create Model Card Update Model

Рисунок 11 – Результаты обучения. Модели


2.2 Создание новой модели

2.2.1 Структура проекта

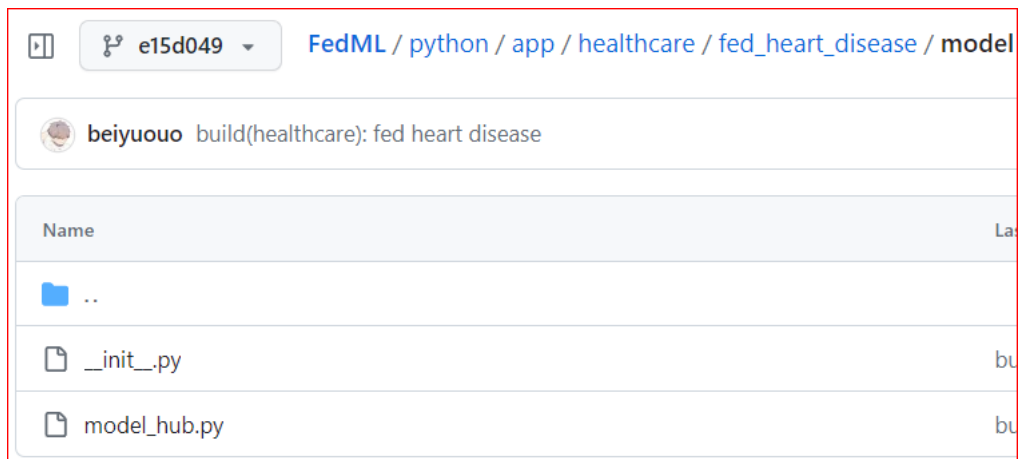
В папке [app](#) содержатся примеры более сложных моделей, разработанных на платформе FedML на новых датасетах. Все проекты имеют следующую структуру:

 e15d049 FedML / python / app / healthcare / fed_heart_disease /	
 chaoyanghe v0.7.361	
Name	
 ..	
 config	
 data	
 model	
 trainer	
 .gitignore	
 README.md	
 __init__.py	
 build_mlops_pkg.sh	
 main_fedml_heart_disease.py	
 requirements.txt	
 run_client.sh	
 run_server.sh	
 run_simulation.sh	

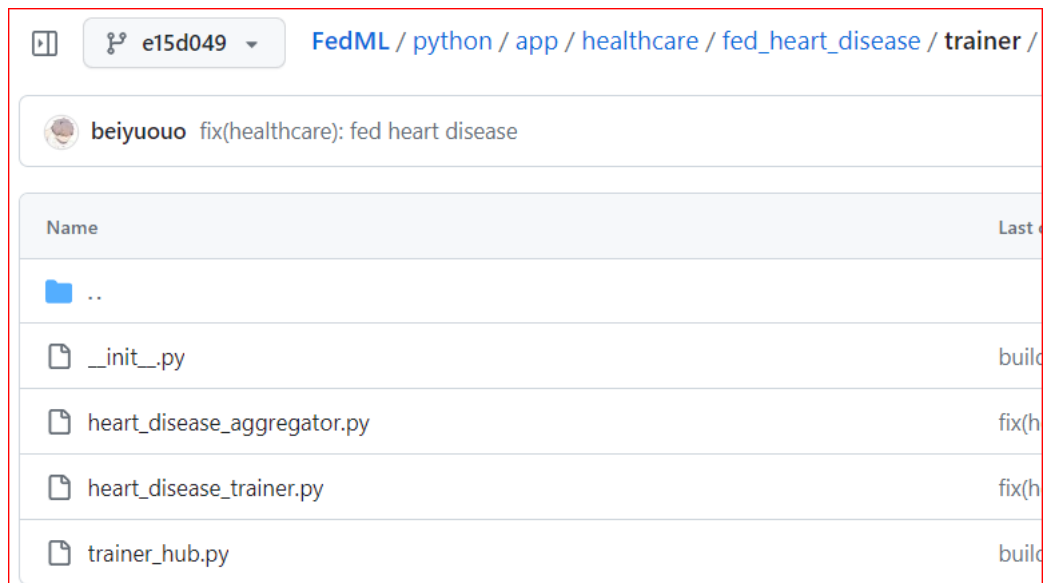
В папке data содержится загрузчик нового датасета

 e15d049 FedML / python / app / healthcare / fed_heart_disease / data /	
 beiyuouo build(healthcare): fed heart disease	
Name	La
 ..	
 __init__.py	be
 data_loader.py	be
 fed_heart_disease.py	be

В папке model содержится инициализация модели



В папке `trainer` содержится код запуска обучения и агрегатор



В папке `config` содержится стандартный файл с конфигурацией федеративного обучения, а так же, если необходима симуляция, дополнительная папка с конфигурацией для симуляции.



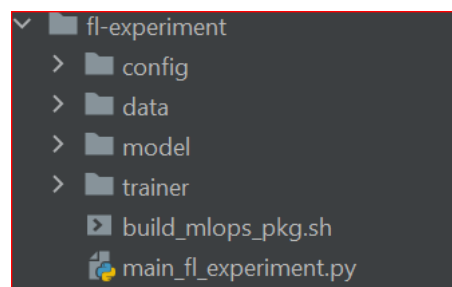
Кроме этого, проект содержит стандартный код и скрипты для запуска клиента и сервера, а также скрипт для генерации zip-файлов, необходимых для работы в MLOps.

2.2.2 Структура загрузчика данных

Код загрузчика данных создавался на основе этих примеров: [heart_desease](#) (загрузка нового датасета и построение модели на нем) и [imageNet](#) (загрузка встроенного датасета с изображениями).

Код загрузчика находится на [гитхабе](#), а также на виртуальной машине 158.160.44.128 в папке fl.

Проект имеет стандартную структуру для FedML:



Файл `main_fl_experiment.py` содержит стандартный код запуска федеративного обучения. С его помощью обучение можно будет запустить в консоли, при использовании FedML MLOps он не нужен.

`build_mlops_pkg.sh` – стандартный код генерации zip-файлов для MLOps.

Загрузчик данных содержится в папке `data`. При запуске обучения аргументы из файла конфигурации передаются функции `load_data` из файла `data_loader.py`.

2.2.3 data_downloader.py

Файл data_downloader.py: сначала происходит скачивание датасета из google drive. Файлы cookies.txt и confirm.txt нужны для подтверждения скачивания, напрямую по ссылке скачать датасет большого объема не получается.

```
file_id = "1qeQbob94err5Zd7DLoq3Q2RsAUbRH0cb"
url = f"https://docs.google.com/uc?export=download&id={file_id}"
session = requests.Session()
response = session.get(url)
confirm_token = None
if "download_warning" in response.content.decode():
    confirm_token = re.search(
        r"download_warning: ([^&]+)", response.content.decode()
    ).group(1)
if confirm_token:
    params = {"id": file_id, "confirm": confirm_token}
    response = session.get(url, params=params)
with open("Dataset.zip", "wb") as f:
    f.write(response.content)
os.remove("cookies.txt")
os.remove("confirm.txt")
```

Далее происходит распаковка архива. После чего датасет приводится к структуре необходимой для дальнейшей работы с FedML. Создаются директории для тренировочной и тестовой выборок. Здесь выборки разбиваются в соотношении 80/20, это можно изменить в соответствующей строке. В данном коде изменен был только Dataset1 и дальше работа происходит с ним, при надобности можно проделать то же самое с Dataset2.

```

# Set up directories
base_dir = data_cache_dir+"/Dataset/Dataset 1" # путь до корневой директории
#base_dir = data_cache_dir + "/dataset"
train_dir = os.path.join(base_dir, "train")
test_dir = os.path.join(base_dir, "test")
#classes = ["class1", "class2"]
classes = ["Bruce Force FTP-Patator", "Bruce Force SSH-Patator",
           "Dos GoldenEye", "Dos Hulk",
           "Dos Slowhttptest", "Dos Slowloris",
           "Heartbleed Port 444", "Normal"] #названия классов
for d in [train_dir, test_dir]:
    if not os.path.exists(d):
        os.makedirs(d)
# Move files to train/test directories by class
for c in classes:
    class_dir = os.path.join(base_dir, c)
    files = os.listdir(class_dir)
    n_train = int(len(files) * 0.8) #здесь можно настроить соотношение

    for f in files[:n_train]:
        src = os.path.join(class_dir, f)
        dst = os.path.join(train_dir, c, f)
        if not os.path.exists(os.path.join(train_dir, c)):
            os.makedirs(os.path.join(train_dir, c))
        shutil.move(src, dst)

    for f in files[n_train:]:
        src = os.path.join(class_dir, f)
        dst = os.path.join(test_dir, c, f)
        if not os.path.exists(os.path.join(test_dir, c)):
            os.makedirs(os.path.join(test_dir, c))
        shutil.move(src, dst)

    shutil.rmtree(class_dir)

```

2.2.4 fl_experiment.py

Файл fl_experiment.py. Здесь содержатся основные функции по загрузке данных. Загрузчик данных FedML для корректной работы фреймворка должен возвращать следующую информацию о датасете:

- train_data_num: Количество объектов во всем тренировочном датасете
- test_data_num: Количество объектов во всем тестовом датасете
- train_data_global: Весь тренировочный датасет

- `test_data_global`: Весь тестовый датасет
- `data_local_num_dict`: Словарь, который связывает каждого клиента с количеством локальных данных, доступных ему.
- `train_data_local_dict`: Словарь, который связывает каждого клиента с его локальным тренировочным датасетом
- `test_data_local_dict`: Словарь, который связывает каждого клиента с его локальным тестовым датасетом
- `class_num`: Количество классов в задаче классификации

Основная функция загрузки, выполняемая при запуске обучения на фреймворке, - **`load_partition_fed_experiment`**. Она получает аргументы из конфигурационного файла и возвращает всю необходимую информацию о датасете.

Если необходимо скачивание датасета, вызывается ранее рассмотренная функция из файла **`data_downloader.py`**.

Количество классов задается в файле конфигурации, но может быть задано здесь напрямую.

С помощью функции **`load_data_global`** происходит загрузка глобальных тренировочного и тестового датасетов. Здесь задается трансформация данных. После чего происходит загрузка датасетов и создание загрузчиков данных.

```

def load_data_global(data_dir, batch_size):
    train_dir = data_dir + '/train'
    test_dir = data_dir + '/test'

    # трансформация для изображений
    transform_train = transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
    ])

    transform_test = transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
    ])

    # Load dataset
    train_dataset = datasets.ImageFolder(train_dir, transform=transform_train)
    test_dataset = datasets.ImageFolder(test_dir, transform=transform_test)

    # Create DataLoader for train set
    train_loader = torch.utils.data.DataLoader(
        train_dataset,
        batch_size=batch_size,
        shuffle=True,
        num_workers=4,
        pin_memory=True
    )

    # Create DataLoader for test set
    test_loader = torch.utils.data.DataLoader(
        test_dataset,
        batch_size=batch_size,
        shuffle=False,
        num_workers=4,
        pin_memory=True
    )

    return train_loader, test_loader

```

После чего, рассчитывается количество объектов в соответствующих датасетах для train_data_num и test_data_num.

Далее происходит распределение данных между клиентами с помощью функции **distribute_data**. Она распределяет данные между клиентами поровну случайным образом. Объектам в датасете присваиваются индексы, и каждый клиент случайным образом получает список индексов, принадлежащих ему объектов. Функция возвращает `data_local_num_dict`, `train_data_local_dict`, `test_data_local_dict`.

Таким образом, вся необходимая о датасете информация получена, функция загрузки завершает работу.

2.2.5 Создание модели

Для запуска обучения на фреймворке необходимо также написать код инициализации модели в папке `model`, код запуска обучения в папке `trainer`, а также указать необходимые параметры в файле конфигурации `fedml_config.yaml`. После чего можно будет сгенерировать zip-файлы для запуска в MLOps или запустить фреймворк через консоль.

ЗАКЛЮЧЕНИЕ

Были изучены основные возможности фреймворка федеративного обучения FedML, такие как: запуск в федеративном режиме, типы данных, доступные модели анализа данных, функции агрегирования, дополнительные функции защиты конфиденциальности, доступные распределения данных, метрики. Был запущен пример федеративного обучения на виртуальных машинах в конфигурации из 1 сервера и 2 клиентов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. <https://github.com/FedML-AI/FedML>
2. <https://doc.fedml.ai>