J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr


# 연합학습시스템에서의 MLOps 구현 방안 연구☆

# The Study on the Implementation Approach of MLOps on Federated Learning System

홍 승 후[1] 이 강 윤[1*]
Seung-hoo Hong KangYoon Lee

**요 약**

연합학습은 학습데이터의 전송없이 모델의 학습을 수행할 수 있는 학습방법이다. IoT 혹은 헬스케어 분야는 사용자의 개인정보를 다루는 만큼 정보유출에 민감하여 시스템 디자인에 많은 주의를 기울여야 하지만 연합학습을 사용하는 경우 데이터가 수집되는 디바이스에서 데이터가 이동하지 않기 때문에 개인정보 유출에 자유로운 학습방법으로 각광받고 있다. 이에 따라 많은 연합학습 구현체가 개발되었으나 연합학습을 사용하는 시스템의 개발과 운영을 위한 시스템 설계에 관한 구체적인 연구가 부족하다. 본 연구에서는 연합학습을 실제 프로젝트에 적용하여 IoT 디바이스에 배포하고자 할 때 연합학습의 수명주기, 코드 버전 관리, model serving, 디바이스 모니터링에 대한 대책이 필요함을 보이고 이러한 점을 보완해주는 개발환경에 대한 설계를 제안하고자 한다. 본 논문에서 제안하는 시스템은 중단 없는 model-serving을 고려하였고 소스코드 및 모델 버전 관리와 디바이스 상태 모니터링, 서버-클라이언트 학습 스케줄 관리기능을 포함한다.

☞ 주제어 : 연합학습, 프레임워크, 개발 시스템 설계, 오픈소스, 개발환경, MLOps

**ABSTRACT**

Federated learning is a learning method capable of performing model learning without transmitting learning data. The IoT or healthcare field is sensitive to information leakage as it deals with users' personal information, so a lot of attention should be paid to system design, but when using federated-learning, data does not move from devices where data is collected. Accordingly, many federated-learning implementations have been developed, but detailed research on system design for the development and operation of systems using federated learning is insufficient. This study shows that measures for the life cycle, code version management, model serving, and device monitoring of federated learning are needed to be applied to actual projects and distributed to IoT devices, and we propose a design for a development environment that complements these points. The system proposed in this paper considered uninterrupted model-serving and includes source code and model version management, device state monitoring, and server-client learning schedule management.

☞ keyword : federated-learning, framework, open source, effective development environment, MLOps

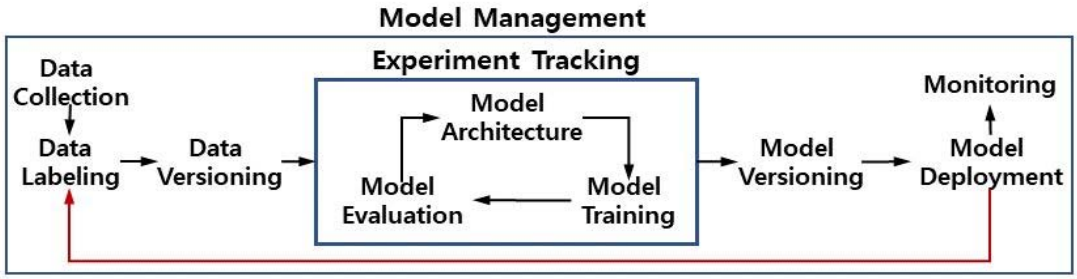
## 1. 서 론

### 1.1 연구배경

CNN(Convolutional Neural Network) 네트워크의 성공 이후 딥러닝(deep learning)은 이미지 인식(image recognition), 자연어 처리, 추천시스템과 같은 다양한 분야에서 사용되고 있다. 딥러닝 활용 분야가 증가하면 딥러닝 모델을 설계하고 이를 학습시키기 위한 작업의 복잡성이 증가한다. 이러한 증가된 복잡성은 딥러닝 모델 운용에 있어 기술부채로 작용하게 되며 전체 시스템을 운용함에 있어 잠재적인 문제가 된다.[1] 이 문제를 해결하기 위해 DevOps(Development Operations)를 바탕으로 모델 설계, 학습, 응용에 공통된 절차를 자동화하는 Machine Learning Operations(MLOps)가 제안되었다.[2]

DevOps는 애자일(Agile) 방법론을 기초로 하는 개발 및 운영에 대한 접근방식으로 2009년 Patrick Debois에 의해 처음 소개되었다[3]. DevOps의 가장 주요한 점은 지속적인 통합(Continuous Integration; CI)과 지속적인 배포

1 Dept. of Computer Engineering, Gachon University, Seongnam, 13120, Korea
* Corresponding author (keylee@gachon.ac.kr)
[Received 18 March 2022, Reviewed 24 March 2022, Accepted 30 March 2022]
☆ 이 논문은 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 대학ICT연구센터육성지원사업의 연구결과(IITP-2022-2017-0-01630)와 동적인 디바이스 환경에서 적응적 연합학습 기술 개발 연구결과(No. 2021-0-00900)로 수행되었음.




(그림 1) 기본적인 MLOps 구조
(Figure 1) Basic structure of MLOps
Source: https://neptune.ai/mlops-tool-stack

(Continuous Delivery; CD)이며 이를 통해 시스템 개발자와 운영자는 짧은 개발주기, 빠른 배포속도, 안정적인 출시의 이점을 얻을 수 있다.[4]

MLOps는 Machine Learning(ML) 모델 사용의 서비스를 제공하는 시스템 개발과 운영을 위한 DevOps 기반 접근방식으로 ML 모델을 사용하지 않는 시스템에 비해 복잡해진 시스템 개발 및 운영에 이점을 제공한다. MLOps는 DevOps와 달리 학습을 위한 데이터 처리 절차, 모델 개발과 테스트, 배포 절차, 모델의 모니터링 지원을 포함한다. 이를 통해 많은 ML시스템이 서비스되고 있다.[5]

아래의 그림 1은 MLOps의 가장 기본적인 흐름을 나타낸 것이다. 데이터를 수집하고 분석 처리하여 모델을 설계 학습시키고 검증하여 배포하고 모니터링 하는 기본 구조를 보인다.[4][5][6] 이러한 구성은 API를 통한 예측 서비스 제공 혹은 특정 디바이스에 배포되는 ML시스템에 적용된다.

한편 디지털 기술의 발전에 따라 사물인터넷과 같은 소비자의 생활과 밀접한 곳에서 작동되는 기기와 기관에서 많은 데이터들이 생산, 수집, 저장되는 데이터의 양은 늘어나고 있다.[7] 이러한 데이터를 사용하여 딥러닝 모델을 학습시키고자 한다면 분산되어있는 데이터들을 한곳으로 모아 전체 데이터 세트를 수집하고 학습시켜야 하지만 이는 참여자의 이해관계, 효율성, 분석시간, 비용 등 많은 제약이 따른다.

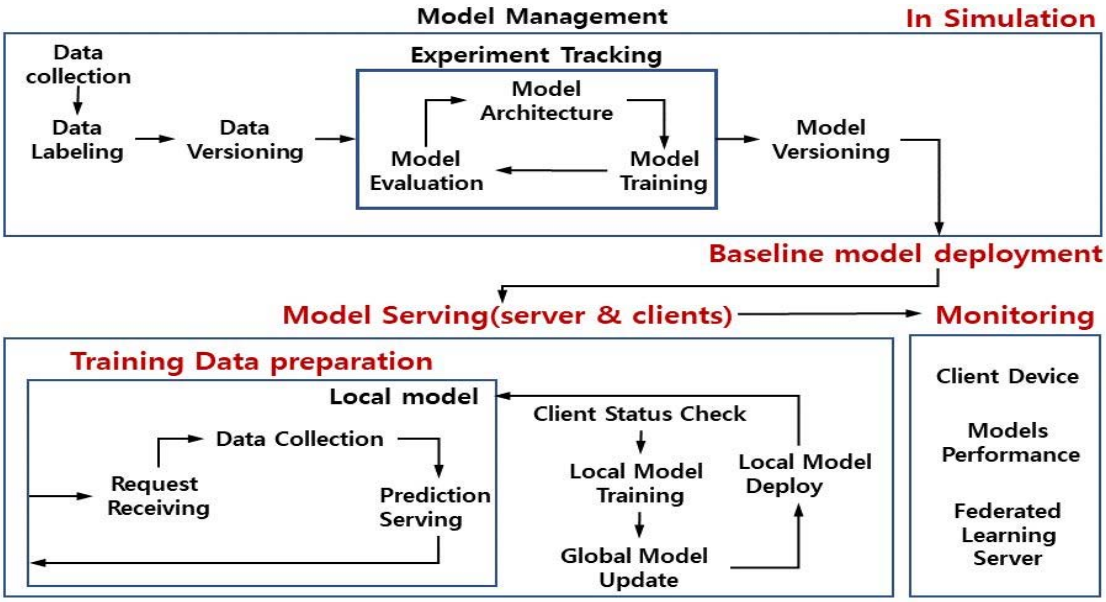
또한 의료정보, 금융데이터와 같은 개인정보들은 데이터의 관리에 매우 엄격한 규제를 따라야 한다. 유럽연합의 '일반 데이터 보호 규칙(GDPR)'과 대한민국의 '데이터 3법'이 그 예시다. 연합학습(Federated Learning)은 데이터를 외부로 전송하지 하지 않으면서 딥러닝 모델을 학습시키는 학습방법이다.[8] 연합학습은 로컬의 데이터로 학습된 각 모델의 가중치를 통합하고 재배포하는 방식으로 이뤄지며 학습에 사용되는 데이터가 외부로 노출되지 않기 때문에 프라이버시 문제에서 비교적 자유롭다. 이러한 특징으로 인해 의료데이터, 개인정보와 같은 높은 수준의 보안이 요구되는 데이터를 다루는 모델을 학습시키는 데 적합하다. 연합학습의 응용분야는 헬스케어나[9] IoT[10][11], 모바일[12][13]과 같은 일상에 가깝거나 민감한 정보를 포함할 수 있는 곳에서 활용되고 있다.

연합학습을 구현하기 위한 라이브러리들[14][15][16]은 존재하지만 대부분의 라이브러리들은 이를 이용해서 실험적인 프로젝트를 진행해보기에는 아래와 같은 몇 가지 현실적인 문제가 남아있다.

먼저 연합학습의 주기관리 클라이언트 시스템 모니터링 코드의 버전관리에 있어서 이 문제는 프로그램 외적으로 지원해주어야 하기 때문에 연합학습 라이브러리에서 지원하지 못한다. 클라이언트 측에서 학습이 이뤄지고 그 결과들을 필요로 하기 때문에 클라이언트는 모델 서빙과 학습 스케쥴관리가 필요[17]하지만 그런 기능을 지원하지는 않기 때문이다.

연합학습 응용에 있어서 단순한 시뮬레이션 이상의 작업을 위해서는 MLOps의 구성을 따르는 것이 유리하다. 이유는 연합학습의 경우 서버와 클라이언트의 구성을 가지며 다수의 클라이언트를 상정하므로 개발과정에 있어서 코드의 수정이 생기게 되면 각 클라이언트와 서버의 버전이 일치해야 하고 ML시스템이기 때문에 모델에 대한 버전과 모니터링 지원도 필요하기 때문이다.

또한 연합학습을 사용하는 시스템은 중앙화되어있는 머신러닝 시스템과 다르게 학습 데이터에 대한 수집과




(그림 2) MLOps에서의 연합학습
(Figure 2) federated learning in MLOps

분석이 불가능하고 계속적인 학습이 요구된다. 클라이언트는 연합학습에서 클라이언트의 역할을 하면서 외부에 모델을 서빙하는 서버의 역할도 겸하고 있기에 전체 시스템의 구성이 복잡하다. 따라서 기존의 MLOps를 연합학습에 적용하기에는 어려움이 따른다. 연합학습을 활용하기 위한 시스템 설계에 대한 연구들[15][17][18][19]이 있지만 추상적인 언어로 서술되어 있고 시뮬레이션을 넘어 실제 디바이스에 연합학습을 적용하는 프로젝트를 개발하기 위한 시스템 설계 연구가 부족하다.

위에서 언급한 문제로 인해 연합학습을 이용하여 실험적인 프로젝트를 진행하고자 할 때 개발자는 서버와 클라이언트의 구성을 정의하고 모니터링 시스템을 구축하는 등의 많은 사전 작업을 해야 하므로 개발자에게 다양한 사전지식을 요구하게 된다.

본 논문에서는 이러한 문제에 대한 해결책으로 다양한 클라이언트 디바이스의 상태 모니터링과 연합학습 주기관리, 클라이언트 서버 코드 버전관리를 지원하는 시스템 설계를 제안한다. 해당 시스템은 또 다른 연합 라이브러리가 아닌 외부 관리 시스템으로 사용자의 다양한 요구사항에 대응할 수 있도록 독립된 컴포넌트들로 이루어져 있다. 시스템의 유연성과 확장성을 위해 이미 개발된 MLOps 도구를 사용하여 구현할 수 있는 기능은 적극적으로 사용하고 쉽게 같은 기능의 다른 MLOps 도구를 시도해 볼 수 있게 디자인되어 있다. 또한 개발자의 실수를 감당할 수 있고 다양한 상황에서도 클라이언트에 대한 접근을 보장할 수 있도록 한다.

## 1.2 연구범위

본 연구는 연합학습을 응용하여 프로젝트를 진행하고자 할 때의 문제점을 분석하고 연합학습 시스템을 위한 MLOps를 제안한다.

본 연구에서는 MLOps에 연합학습을 적용하고자 할 때 필요한 복잡한 시스템 구성을 분석하고 이를 지원하는 단순하고 유연한 관리 시스템을 제안한다. 해당 시스템은 서버, 클라이언트에 위치하는 관리 시스템으로 개발의 용의성을 위해 단순한 API를 사용하며 연합학습의 라이브러리와는 독립적으로 작동한다, 이를 통해 개발자가 선호하는 연합학습 구현체를 사용할 수 있다.

본 연구가 제안하는 시스템은 실제 디바이스로의 원할한 확장과 안정적인 배포를 지원한다.

## 2. 관련연구

### 2.1 연합학습

연합학습은 다수의 클라이언트를 갖는 서버의 모델을 분산학습(distributed learning)하는 방법이다.[20] 연합학습은 클라이언트에 데이터가 분산되어 있는 상황을 전제로 하여 데이터를 한곳에 모으지 않고 각 클라이언트에서 학습한 뒤 이를 이용한다는 점에서 다른 분산학습과 차이가 있다. Federated Averaging(FedAvg)[21]이 표준적인 연합학습 알고리즘이며 전체 학습은 1) 임의의 클라이언트를 샘플링, 2) 서버가 글로벌 모델(Global Model)을 클라이언트에 전송, 3) 각 클라이언트에서 로컬 모델(Local Model) 학습 후 서버로 모델을 전송, 4)서버는 받은 로컬 모델들의 평균으로 글로벌 모델을 업데이트 하는 4단계를 거친다.

### 2.2 연합학습 플랫폼

연합학습을 위한 플랫폼은 여러 기관에서 개발되었다.[20][21] TensorFlow Federated[22], Flower[14]와 같은 라이브러리 수준의 플랫폼을 사용하는 경우는 개발자가 시스템 전반을 개발해야 한다는 단점이 있고 FATE (Federated AI Technology Enabler)[16]나 PaddleFL[23]과 같은 산업용 수준의 플랫폼은 배포, 모니터링, 작업속도(workflow)까지 지원하지만, 반드시 플랫폼에 속한 연합학습 라이브러리를 사용해야만 하고 제공되는 도구 외의 선택사항이 없다는 단점이 있다. 본 연구에서 제안하는 시스템은 연합학습 라이브러리에 독립적이며 다른 MLOps도구[24]와 호환이 되도록 한다.

### 2.3 MLOps

MLOps는 Machine Learning과 DevOps의 합성어로 데이터 수집, 처리에서부터 모델 개발, 평가, 배포, 서비스, 모니터링의 파이프라인을 자동화한다.[6] MLOps는 CD/CI를 ML모델에 적용하여 ML시스템을 계속 운용할 수 있도록 한다. MLOps의 기본적인 구조는 1) 데이터 수집 2) 데이터 처리 3) 모델 설계 4) 모델 학습 5) 모델 평가 6) 모델 배포 7) 모델 서비스 8) 모니터링이다.[25] 이러한 파이프라인 모델은 연합학습에는 적합하지 않다. MLOps는 클라이언트의 상태는 고려사항이 아니지만 연합학습은 클라이언트와 서버가 있고 학습된 클라이언트의 모델들이 서버에서 애그리게이션(aggregation)되어 다시 배포되는 형식이기 때문에 클라이언트에서의 관리도 필요하기 때문이다.

연합학습의 MLOps에서 초기 글로벌 모델이 학습되는 것은 시뮬레이션에서 일어난다. 개발자는 연합학습 라이브러리의 시뮬레이션을 이용하여 확보된 데이터를 바탕으로 여러가지 모델을 학습시키고 평가한다. 이렇게 정해진 초기모델을 배포하면 클라이언트는 배포 받은 모델을 가지고 인퍼런스(inference)를 제공한다. 인퍼런스 서비스를 제공하면서 클라이언트는 데이터들을 수집하고 저장한다. 한편 서버는 주기적으로 애그리게이션을 준비한다. 클라이언트는 서버가 애그리게이션 준비가 되어있고 클라이언트 디바이스가 학습을 진행할 조건이 만족되었는지 확인한 후에 연합학습에 참여한다(그림 2). 이 과정에서 클라이언트는 학습에 참여할지 판단하기 위해 서버의 상태를 주기적으로 파악하여야 하며 서버에서도 주기적으로 애그리게이션을 준비하고 서버의 상태를 클라이언트가 알 수 있도록 해야 한다. 이와 더불어 서버-클라이언트의 모델과 코드에 대한 CI/CD가 가능해야 하므로 기존의 MLOps로는 한계가 있다.

## 3. 연구방법론

### 3.1 연구모형

제안하는 시스템은 아래와 같은 요소를 만족한다.

모듈화: 시스템은 각각이 API로 통신하는 컴포넌트(component)로 이루어져 있다.

높은 오픈소스(open sauce) 활용성: 쉽고 단순한 API와 널리 사용되는 라이브러리를 사용하여 확장성을 높인다.

모델 개발과 환경 개발 분리: 애그리게이션 알고리즘(aggregation algorithm)과 응용 환경 분리 모델과 환경 개발을 통일된 방식으로 관리, 연합학습 시각화, 클라이언트 디바이스(Client device) 모니터링을 지원한다.

시스템의 구성은 그림 3과 같다. 연합학습은 연합학습 라이브러리를 통해 이뤄지며 클라이언트의 모델 서빙과 버전 관리, 트레이닝 트리거(training trigger)를 클라이언트 관리 컴포넌트(Client Management Component)가 관리한다. 그리고 서버의 관리 컴포넌트는 연합학습 라이브러리의 서버를 실행시켜 애그리게이션을 시작한다. 디바이스 모니터링은 클라이언트 혹은 서버의 리소스, 프로그램 로그, 현재 상태를 사용자에게 보이고 트레이닝 모니터링 서비스는 학습로그와 메트릭(metric) 그래프 등을 사용자에게 제공한다. 시스템은 서버-클라이언트 구성을

가지며 서버의 컴포넌트는 아래와 같다.

서버 관리 컴포넌트(Server Management Component): 주기적으로 연합학습 라이브러리의 서버를 실행시켜 애그리게이션을 시작시키는 역할과 서버의 상태를 클라이언트에게 알리는 역할을 맡는다.

디바이스 모니터링(Device Monitoring): 프로그램 에러 로그와 리소스 사용 상황, 서버-클라이언트 관리 컴포넌트의 상태를 모니터링한다. 응용프로그램 개발자 혹은 디바이스 관리자가 이를 보고 기능오류 혹은 프로그램 개선을 진행한다.

트레이닝 모니터링 서비스(Training Monitoring Service): 글로벌 모델의 학습 상황을 모니터링한다. 모델 개발자는 이를 보고 모델을 개선한다.

글로벌 모델(Global Model): 학습의 대상인 모델로 모든 클라이언트는 해당 모델을 공유한다.

연합학습 서버(Federated Learning Server): 연합학습에서 애그리게이션을 하는 역할로 정해진 횟수의 애그리게이션을 마치면 리소스를 반환하고 종료된다. 서버 관리 컴포넌트에 의해 생성되고 서버 관리 컴포넌트에 상태를 보고한다.

클라이언트는 연합학습 클라이언트(Federated Learning Client)와 클라이언트 관리(Client Management), 인퍼런스 서버(Inference Server), 데이터 파이프라인(Data Pipeline)으로 이루어져 있다. 각 컴포넌트는 서로에 대해 독립적이며 관리 컴포넌트(Management Component)의 요청에 응답만 가능하다면 개발자는 각 컴포넌트를 자유롭게 구성하는 것이 가능하다. 이에 대한 설명은 아래와 같다.

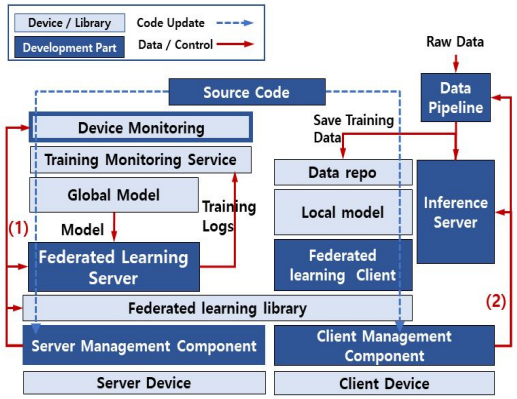
클라이언트 관리 컴포넌트(Client Management Component): 코드와 모델 버전과 인퍼런스 서버에 배포된 모델을 항상 최신 상태로 유지하고 클라이언트의 다른 컴포넌트와 통신으로 상태를 확인하여 항상 정해진 상태를 유지하도록 한다. 또한, 서버의 애그리게이션이 준비되었는지 확인하고 클라이언트가 학습에 참여할 수 있다면 연합학습 클라이언트에 트리거를 주어 학습을 시작한다.

연합학습 클라이언트(Federated Learning Client): 연합학습 라이브러리를 통해 연합학습을 수행하며 클라이언트 관리 컴포넌트의 신호 때문에 시작된다.

인퍼런스 서버(Inference Server): 해당 컴포넌트의 이름은 서버이지만 실제로 모델 서빙을 제공하는 컴포넌트다. 항상 동작 중이고 최신 글로벌 모델을 제공한다.

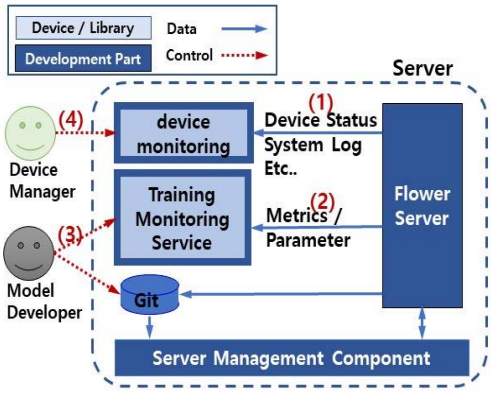
그림 3을 통해 제안하는 시스템 개념도를 알 수 있다.

본 연구에서 제안하는 시스템은 서버와 클라이언트의 관리 컴포넌트를 중심으로 이루어진다. 서버 관리 컴포넌트는 연합학습의 서버부와 모니터링 서비스를 관리한다. 클라이언트 관리 컴포넌트는 인퍼런스 서버와 연합학습 클라이언트를 관리하며 데이터 파이프라인의 관리도 맡는다.




(그림 3) 논문에서 제안하는 시스템의 개념도

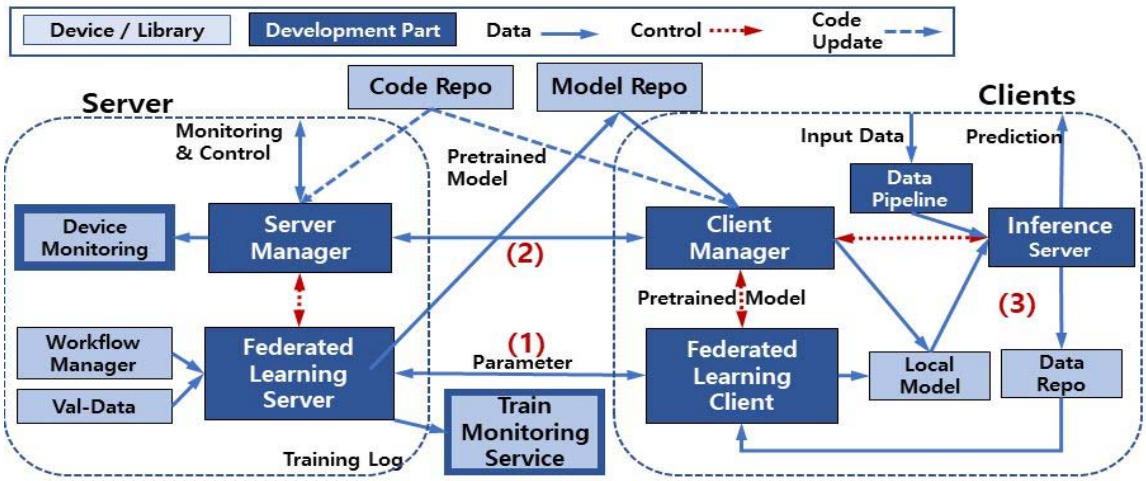
(Figure 3) Conceptual diagram of the proposed system




(그림 4) 제안한 시스템에서의 중앙관리

(Figure 4) Central management in the proposed system

본 연구에서 제안하는 시스템은 전체 시스템 개발을 담당하는 디바이스 매니저와 모델의 개발을 맡는 모델 개발자(model developer) 간의 역할 구분을 전제하고 있다. 1) 연합학습 서버에서의 디바이스 상태 로그(device status log) 및 시스템 로그(system log) 등을 디바이스 모




(그림 5) 제안한 시스템 구상
(Figure 5) Scheme diagram of the proposed system

니터링에서 보고, 2) 학습 상황이나 성능을 트레이닝 모니터링 서비스에서 본다. 3) 모델 개발자는 트레이닝 모니터링 서비스를 보고 모델을 조정하여 기트(Git)에 올려 재배포한다. 4) 디바이스 매니저는 디바이스 모니터링을 보고 조치를 취한다.

시스템의 컴포넌트들 간의 관계는 그림 5를 통해 확인할 수 있다. 연합학습 서버와 클라이언트의 파라미터(parameter) 교환은 연합학습과정에서의 모델 교환을 의미한다. 서버 관리 컴포넌트는 연합학습 서버를 주기적으로 실행시키며 초기모델 업로드와 같은 절차를 다루는 워크플로우 관리(workflow manager)와 그 외
서버의 상태를 클라이언트에게 알려주는 서버 매니저로 구성된다. 1) 연합학습 클라이언트와 서버에서는 gRPC (google's Remote Procedure Calls: 구글의 원격 프로시저 호출)를 통해 통신이 이루어진다. 2) 서버/클라이언트의 관리 컴포넌트 사이의 통신은 RESTful API를 사용하여 이루어진다. 3) 인퍼런스 서버는 클라이언트에서 외부의 데이터를 입력받아 예측값을 제공하고 해당 데이터와 정답을 저장한다.

서버 매니저(Server manager): 연합학습 서버를 제어한다. 클라이언트 매니저와 통신이 이루어지며 클라이언트의 상태와 서버의 상태를 교환한다. 코드 저장소(Code repository)에 변동사항이 있을 때 이를 적용한다.

워크플로우 관리기(Workflow manager): 주기적인 애그리게이션 수행과 초기모델의 배포와 같은 절차적인 것을 다루는 컴포넌트로 정해진 일정에 맞추어 연합학습 서버를 실행한다.

연합학습 서버(Federated learning server): 애그리게이션을 수행하며 검정 데이터(Validation Data)를 통해 학습결과를 확인한다. 학습과정 모니터링은 트레인 모니터링 서비스(Train monitoring services)를 이용한다. 또한 모델 저장소(Model repository)가 비어있거나 저장되어 있는 모델의 구조(Architecture)가 일치하지 않을 경우, 새로운 모델을 업로드한다.

코드 저장소(Code repository): 깃(Git)와 같은 코드 저장소로 코드버전관리와 단일소스저장소 역할을 한다.

모델 저장소(Model repository): 글로벌 모델이 저장되는 장소다.

클라이언트 매니저(Client manager): 코드 저장소의 코드와 버전 동기화, 인퍼런스 서버의 모델 업데이트, 연합학습 클라이언트 학습 시작 명령 전달한다. 처음에 글로벌 모델 다운로드를 한다.

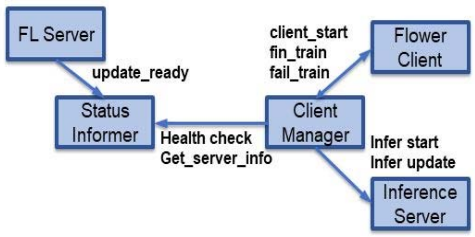
인퍼런스 서버(Inference server): 데이터를 입력받고 예측값을 반환한다. 입력 데이터는 학습을 위해 저장된다.

연합학습 클라이언트(Federated learning Client): 연합학습을 수행한다.

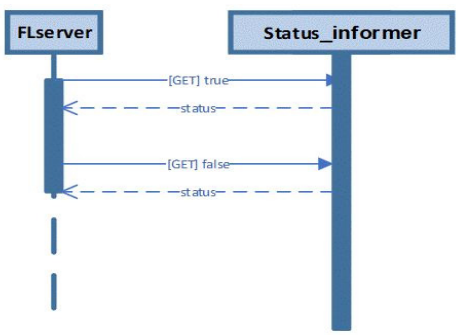
### 3.2 연구 방법 및 절차

본 논문에서 제안하는 시스템은 각 컴포넌트의 독립

성을 보장하고 높은 확장성을 위해 함수 호출 방식이 아닌 API 통신을 사용한다. 그림 6은 시스템에서 이루어지는 API 통신들을 보여준다.

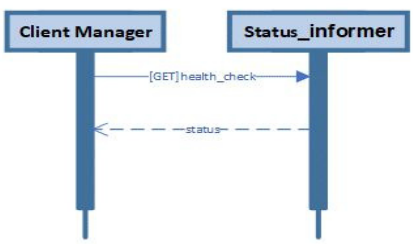



(그림 6) 시스템 컴포넌트 간의 API 통신들
(Figure 6) API communication between system components



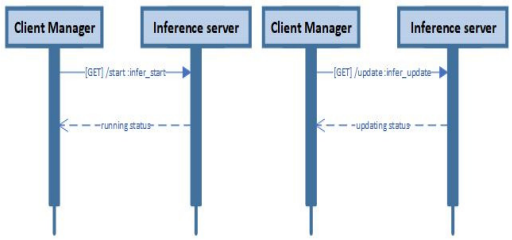

(그림 7) 헬스 체크
(Figure 7) Health check

Client manager ←→ Status informer
Health check → : Federated learning server가 준비되었는지 확인한다.
Get_server_info →: 처음 클라이언트가 서버에 접속하는 경우 모델 저장소에 대한 정보 요청.

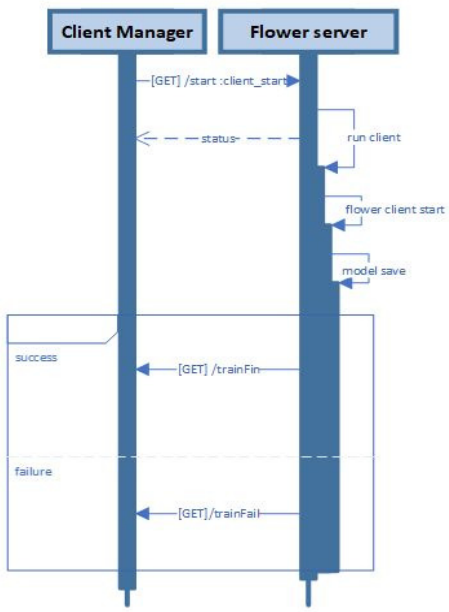



(그림 8) 업데이트 준비
(Figure 8) update_ready

Flserver → Status_informer (그림 8)
update_ready → 연합학습 서버가 준비됨.




(그림 9) 인퍼런트 스타트 / 업데이트
(Figure 9) infer_start / Infer_update




(그림 10) 클라이언트 스타트/ 학습 완료 / 학습실패
(Figure 10) Client start/ fin_train / fail_train

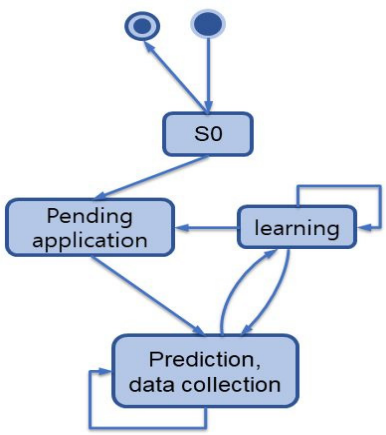
ClientManager ←→ flower_client (그림 10)
Cclient_start →: Client Manager의 연합학습 참여 명령
fin_train ← : 학습이 성공적으로 끝난 경우 반환
fail_train ←: 학습이 실패한 경우 반환

### 3.3 연구 모형설계

시스템의 컴포넌트들은 다양한 상태를 가진다.
서버는 3가지 상태를 가진다. 1) 초기 모델 배포, 2) 워

크플로우 관리기(workflow manager)의 트리거(trigger) 대기, 3) 클라이언트 접속 대기, 등이다.

1) 초기모델 배포: 모델 저장소(Model repository)가 비워 있거나 모델 저장소에 저장되어있는 모델의 학습체계(Architecture)가 일치하지 않을 경우 새로운 모델을 업로드한다.
2) 워크플로우 관리기(Workflow manager)의 트리거 대기: 대기 중인 상태의 서버에는 서버 매니저와 워크플로우 관리기만 있는 상태다. 주기적으로 워크플로우 관리기가 연합학습 서버를 배포하면 서버는 클라이언트 접속대기상태로 바뀐다.
3) 클라이언트 접속대기: 연합학습 서버가 배포되어 학습을 시작한다.



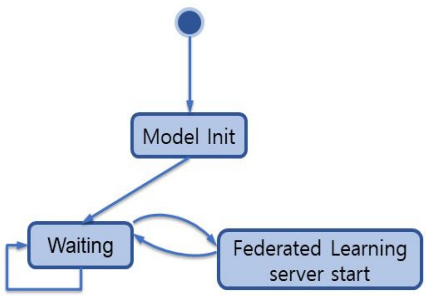

(그림 11) 클라이언트 상태 전이도
(Figure 11) client status diagram

그림 11에서처럼 클라이언트의 상태는 1) S0, 2) 모델적용 대기, 3) 예측 및 데이터 수집, 4) 연합학습 등 4가지가 있다. 이러한 상태의 전이를 클라이언트 매니저가 각 컴포넌트에 API를 통해 전달한다.

1) S0: 클라이언트가 처음 시작되었을 때의 상태로 서버 매니저에 접근하여 모델 저장소에 대한 정보와 글로벌 모델을 받아 다음 상태로 넘어가야 한다. 실패 시 클라이언트는 종료된다.

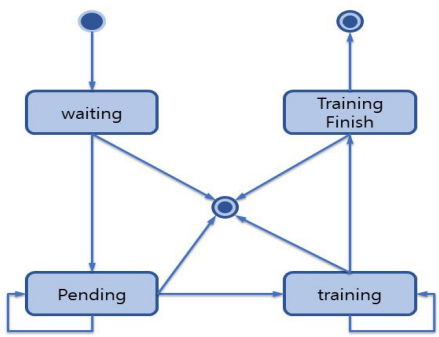
2) 모델적용 대기(Pending application): 클라이언트 내에 로컬 모델(local model)의 변경사항이 있어 인퍼런스 서버의 모델을 업데이트해야 하는 상태로 처음 글로벌 모델을 받아왔거나 연합학습 종료 후의 상태이다.

3) 예측 및 데이터 수집 중(Prediction, data collection): 로컬 모델을 모델 서빙(model serving)을 수행하면서 데이터를 수집한다. 서버가 클라이언트 접속대기 상태인 것이 확인되고 클라이언트의 상태가 학습에 참여 가능한 상태가 되면 연합학습 중 상태로 전이된다.
4) 연합학습 중(learning): 연합학습에 참여하여 학습이 진행 중인 상태로 학습이 정상적으로 종료되면 모델적용 대기 상태로 비정상적으로 종료되면 예측 및 데이터 수집 중으로 다시 돌아온다.




(그림 12) 서버 상태 전이도
(Figure 12) server status diagram

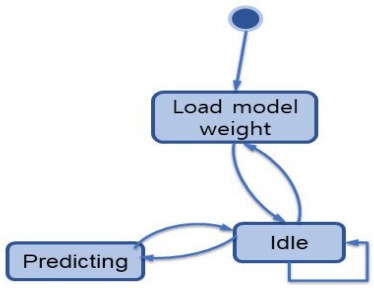
서버의 상태 전이도이다. 1) (Model Init): 모델 저장소에 저장된 모델이 없다면 초깃값을 저장한다. 2) (Waiting): 이후 대기 상태에서 로컬의 클라이언트가 전송하는 쿼리에 응답하며 대기한다. 3) (Federated Learning server start): 정해진 주기가 되었을 때 연합학습 서버를 배포함으로써 연합학습을 실행한다.



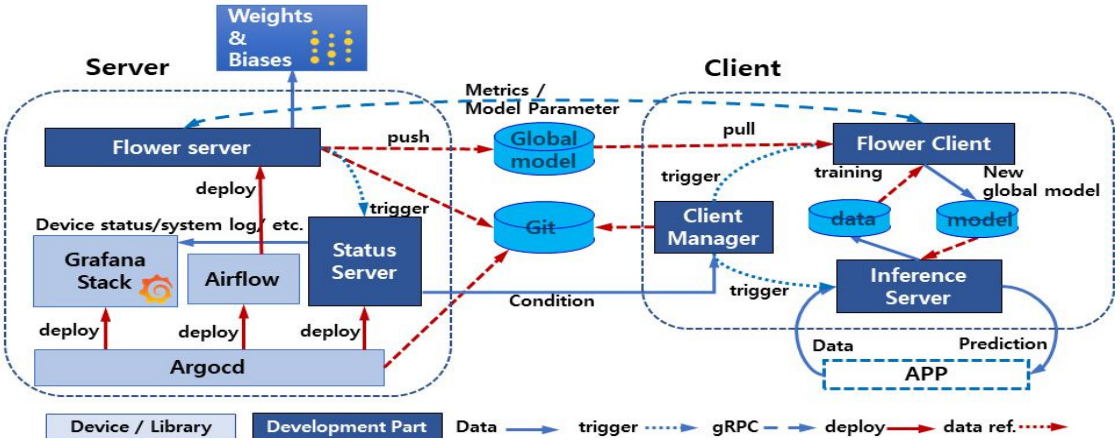

(그림 13) 연합학습 클라이언트의 상태 전이도
(Figure 13) Federated learning client status diagram

그림 13은 연합학습 클라이언트의 상태 전이도를 보여준다. 본 연구에서는 플라워(flower)[14]라는 연합학습 구현체 사용을 전제로 한다. 1) 준비 중(waiting) : 클라이언트 매니저의 트레인 스타트(train start) 명령을 대기하고 있다. 2) 대기 중(Pending): 서버에 접속하고 다른 클라이언트의 접속을 대기한다. 3) 학습 중(training): 학습을 진행한다. 4) 학습완료(Training Finish): 연합학습을 완료하고 모델을 저장한다.

우측의 그림 14는 인퍼런스 서버의 상태를 보여준다. 이 과정에서 외부의 데이터를 입력받아 예측값을 제공하고 해당 데이터와 정답을 저장한다.



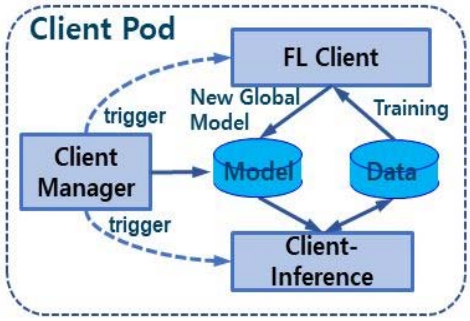

(그림 14) 인퍼런스 서버 상태 전이도
(Figure 14) Inference server status diagram




(그림 15) 구현 설계
(Figure 15) Implementation design

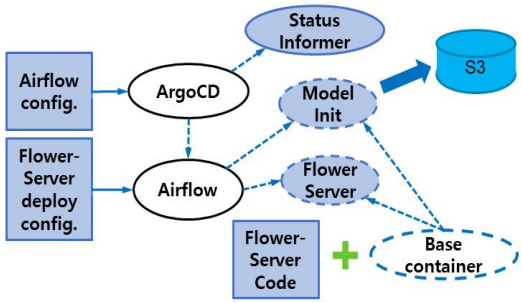
## 4. 실험 방법

구현을 위해 워크플로우 관리기(Workflow manager)는 airflow[26]로 연합학습 라이브러리는 FLOWER[4]로 디바이스 모니터링은 Grafana Stack[27], 트레인 모니터링 서비스는 Weights & Biases[28]로 하였고, 코드저장소는 Git-hub[27], 모델 저장소는 AWS[29]의 S3[30]를 사용한다. 그림 15가 위의 사항을 반영한 그림이다. 또한 인프라스트럭처(infrastructure)는 Kubernetes[31]위에 Argo-CD[32]를 이용하여 구성한다. 그림 15와 그림 16에서 Air-flow[26]를 통해 초기모델 업로드를 수행한다. 컨테이너[33]를 사용하는 경우 코드 변경시 이미지를 재빌드하는 번거로움이 있기에 런타임에 깃허브에서 소스를 받아오는 컨테이너 이미지를 베이스 컨테이너로 사용한다.




(그림 16) Kubernetes에서 구현된 클라이언트 컨테이너들은 공유 저장소가 있다 (emptydir)
(Figure 16) Client implemented on Kubernetes. Containers have shared storage (emptydir)

그림 16에서 클라이언트의 컴포넌트들 사이 상호작용을 보여준다. 클라이언트 매니저가 연합학습을 실제로 수행하는 fl-client와 외부의 데이터에 대해 예측값을 전달하는 클라이언트-인퍼런스 컴포넌트에 트리거 역할을 하며 fl-client와 client-inference는 로컬 모델 저장소와 로컬 데이터 저장소를 공유한다. 구현에서는 pod의 emptydir을 통해 구현한다.




(그림 17) Airflow[26]를 통해 초기 모델 업로드한다. 런타임에 깃허브에서 소스를 받아 컨테이너 이미지를 베이스 컨테이너로 사용한다.

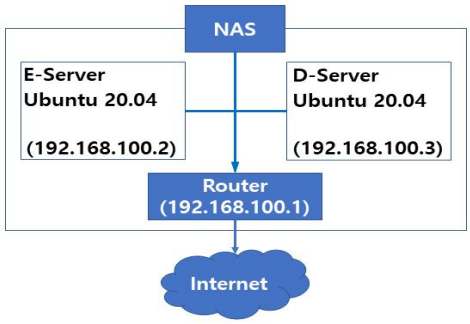
(Figure 17) The initial model is uploaded through Airflow[26]. Use the container image that pulls the source from gitHub as the base container for Runtime.

그림 17에서는 서버의 배포와 주기적 서버시작 방법을 보여준다. ArgoCD를 통해 깃허브에 저장된 내용을 바탕으로 서버시스템을 배포하며 airflow에서는 model init를 통해 모델의 초기값을 배포하며 플라워 서버(flower server)를 배포하여 학습을 시작한다. 각 컨테이너는 런타임 깃허브를 복제(clone)해 오도록 하여 재활용성을 극대화한다. 모델 저장소는 S3를 사용한다.

## 4.1 실험환경

구현은 그림 18과 같은 환경에서 이뤄진다. Kuber-netes[31]를 기반으로 한다. 그림 18과 같이 네트워크를 구성하고 Kubernetes를 배포한다. 서버 측에는 Argocd를 배포하고 argocd 위에서 airflow와 grafana stack을 배포한다.

클라이언트 측은 서버와 같은 환경을 사용하기 때문에 실험환경에서 추가적으로 취해야 할 것은 없으나 서버와 클라이언트가 다른 기기에서 작동할 때는 클라이언트에 kubernetes를 배포하고 그림 16의 client pod를 배포한다.




(그림 18) Kubernetes 구현 환경

(Figure 18) Kubernetes Environment

## 4.2 실험 분석

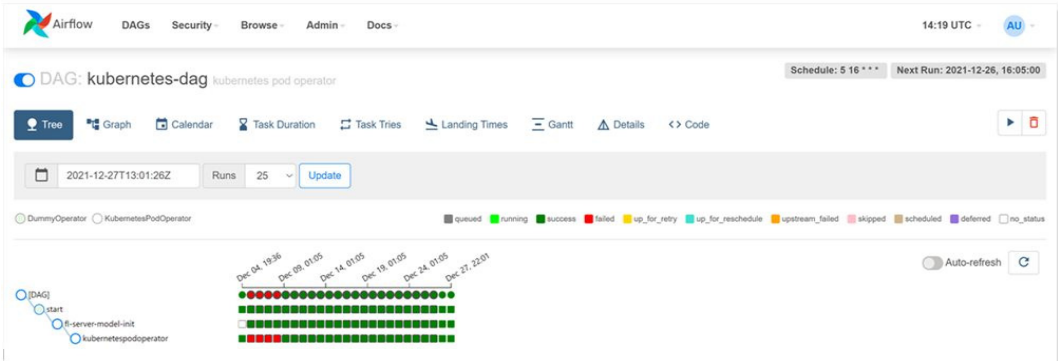
전체 실험 절차는 1) 코드 저장소 준비, 모델 저장소 준비 2) Workflow 관리 프로그램 배포, 3) 서버 배포 4) 클라이언트 배포 5) 학습진행 6) 학습결과 확인 등이다.

프로젝트 실험에서 사용되는 Jetson Nano나 Rasp-berryPi와 같은 임베디드 개발보드에서도 k3s와 같은 경량형 kubernetes를 지원하고 이러한 컨테이너 기반의 시스템은 하드웨어 의존도가 낮고 재현성이 높다. 따라서 서버와 클라이언트의 모든 구현은 모두 GitHub의 내용을 동기화되도록 하여 Kubernetes에서 pod형태로 구현된다. Kubernetes의 배포(deployment) 방식을 이용하여 서버를 배포하고 ReplicaSet을 이용하여 클라이언트들을 배포한다. Deployment는 kubernetes에서 업데이트와 롤백(rollback)을 지원하지만 ReplicaSet은 그렇지 않고 자유롭게 스케일을 늘릴 수 있기 때문에 클라이언트는 비교적 현실적이면서도 실험에 용이하다.

코드 저장소는 GitHub를 사용한다. 모델 저장소는 객체기억장소(object storage)인 AWS의 S3서비스를 사용한다. 업로드할 때에는 사용자인증이 필요하도록, 다운로드할 때에는 누구나 가능하도록 설정한다.

Workflow는 아파치(apache) Airflow를 사용한다. 시스템 배포는 gitHub 동기화 기능을 지원하는 Argo-CD를 사용한다. 상태 서버(status server)를 배포하고 클라이언트를 배포한다.

학습은 엠니스트(mnist) 데이터를 사용하며 3층의 완전접속층(fully connected layer)을 사용하여 32배치(batch) 사이즈로 2에포크(epoch)씩 4라운드로 이뤄지며 모델의 성능이 아닌 전체 시스템의 정상 작동을 확인하는 것을 목적으로 각 라운드의 로스(loss)값의 시작이 이전 라운드의 마지막 값과 일치하는지를 통해 글로벌 모델이 라운드를 거듭할수록 학습되는지 확인한다.

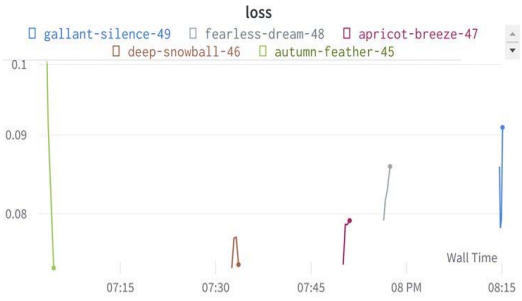



(그림 19) Airflow를 통해 학습이 정상적으로 수행되고 있는 것을 확인할 수 있다.
(Figure 19) It can be confirmed that learning is being carried out normally through Airflow.

## 4.3 실험 결과

실험환경에서 구현된 모습이다.

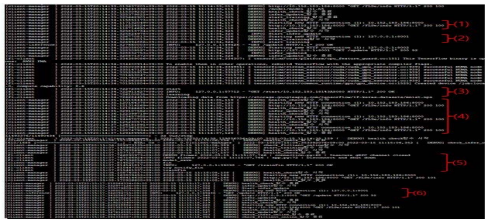
그림 19는 airflow dash board로써 좌측하단은 태스크 그래프(task graph), 우측은 실행 성공 내역을 실행시간별 태스크 당 도형의 색을 통해 표현한 것이다. 세팅 후 초반에는 코드 오류로 4번의 실패가 있었지만, 수정 후에는 정상적인 실행이 계속 이루어졌음을 확인할 수 있다.




(그림 20) 5 라운드 로스 그래프
(Figure 20) 5 round loss graph

그림 15의 Weights & Biases를 이용해 기록한 test loss 그래프로 y축은 loss를 x축은 학습이 진행된 실제 시간이다. Weights & Biases의 기본 설정에 따라 이름이 임의로 정해졌으며 'autumn-feather-45'에서 이미 학습이 수렴상태(saturation)가 되어 이후부터 과적합(overfitting) 되는 것을 볼 수 있다. 이 점은 weight가 학습을 시작할 때마다 초기화되지 않았다는 것을 의미하기 때문에 구현한 시스템이 continual learning을 정상적으로 수행하고 있음을 알 수 있다.

Wall time에 따라 끊어져 있지만, 다음 라운드의 시작이 이전 라운드의 마지막 결과에서 이어지는 것을 보아 글로벌 모델 저장소에 저장된 최신(lastest) 모델을 받아와 정상적으로 학습이 진행되었음을 알 수 있다.



(그림 21) 학습로그
(Figure 21) Training Logs

그림 21은 클라이언트에 기록된 학습 로그로 우측에 Pod 내의 컨테이너 네임, 로그 기록시간, 로그 타입, 로그 내용이다. 로그에 기록되는 컨테이너는 클라이언트에 Client-manager, fl-client, client-inference가 있다. 프로그램은 함수의 시작과 종료 시에 함수 이름을 로그에 남긴다. (1)행: Client-manager가 server status에게서 서버 상태를 받아온다. (2)행: 시작(start) 명령을 fl-client에 보내는 것을 확인할 수 있다. (3)행: Fl-client는 수신확인을 보내고 학습을 시작한다. (4)행: Fl-client에서 연합학습을 진행하는 동안 fl-manager는 각 컴포넌트의 상태를 확인한다. (5)행 fl-client에서 연합학습이 종료되었다는 것을 클라이언트 매니저에게 알린다. (6)행: client-inference에게 업데이트를 명령한다.

## 5. 결론 및 향후연구

### 5.1 결론

본 연구에서는 연합학습 시스템을 이용한 프로젝트 진행에 있어 만나게 되는 어려움을 소개하고 이를 해결할 수 있는 개발 시스템 설계를 기계학습 시스템 개발/운영에 대한 철학인 MLOps의 관점에서 제시하였다. 제안한 시스템은 연합학습 구현체와 독립적으로 설계되었고 다양한 하드웨어에서 원활히 실행 가능하도록 하기 위해 컨테이너 기반 구현을 전제로 하였다. 본 연구에서 제안하는 시스템은 서버와 클라이언트에 관리를 위한 컴포넌트를 추가하여 연합학습 응용프로그램 외적으로 필요한 서비스를 제공한다. 해당 시스템의 컴포넌트들은 서로 API를 통한 통신을 이용하고 마이크로 서비스로 분할하여 각 컴포넌트를 자유롭게 개발할 수 있도록 하였다. 추후 많은 개발자가 연합학습을 응용하여 프로젝트를 진행하고자 할 때 시스템 구성의 참조자료로써 사용할 수 있을 것으로 기대한다.

### 5.2 향후연구

본 연구에서는 Kubernetes를 기반으로 시스템을 구축하고 있으며 클라이언트도 Kubernetes를 기반으로 하고 있다. 향후 연구에서는 Kubernetes와 독립적으로 작동할 수 있는 환경을 구축할 필요가 있다. 또한 클라이언트 신뢰도와 애그리게이션 시 반영할 클라이언트 가중치(weight) 별 기여도를 런타임에서 모니터링 할 수 있도록 하고, 다양한 조합의 애그리게이션에 따른 성능 향상을 예측할 수 있도록 하여 연합학습을 위한 플랫폼의 완성도를 향상시키는 연구를 기대하고 있다.

## 참고문헌(Reference)


[1] Sculley, D. et al. (10 others), “Hidden technical debt in machine learning systems”, Advances in neural information processing systems 28, 2015. https://doi.org/10.48550/arXiv.2103.10510

[2] Seong-yoon, Byen, “Arrangement of MLOps in the Warring States period”, MLOps KR, Announcement of the 1st online event, 2021. https://speakerdeck.com/mlopskr/mlops-cuncu-jeongug-sidae-jeongri-byeonseongyun

[3] Patrick Debois, “Devops café Episode 12”, Devopsday, 2009. http://devopscafe.org/show/2010/9/15/episode-12.html

[4] Pandey, V. and Bengani S., “Operationalizing Machine Learning Pipelines: Building Reusable and Reproducible Machine Learning Pipelines Using MLOps”, Bpb Publications, (English Edition), 2022.

[5] Zhou, Y., Yu, Y., and Ding, B., “Towards MLOps: A Case Study of ML Pipeline Platform”, 2020. International Conference on Artificial Intelligence and Computer Engineering(ICAICE), 494-500, 2020. doi: 10.1109/ICAICE51518.2020.00102.

[6] Treveil, M. et al. (5 others), “Introducing MLOps”, O’Reilly Media, 2020. https://doi.org/10.48550/arXiv.1911.06270

[7] Seung-min, Lee, “Technological Trends and Industrial Implications of Federated Learning”, Technology Policy Trends, 2020. https://library.etri.re.kr /service/main/index.htm?eco=open

[8] Qinbin, L. et al. (7 others), “A survey on federated learning systems: vision, hype and reality for data privacy and protection”, IEEE Transactions on Knowledge and Data Engineering, 2021. https://doi.org/10.48550/arXiv.1907.09693

[9] Xu, J. et al. (5 others), “Federated Learning for Healthcare Informatics”, 1-19, 2021. https://doi.org/10.1007/s41666-020-00082-4

[10] Zhou, J. et al. (10 others), “A survey on federated learning and its applications for accelerating industrial internet of things”, arXiv preprint arXiv:2104.10501, 2021. https://doi.org/10.48550/arXiv.2104.10501

[11] Zhang, T. et al. (5 others), “Federated learning for internet of things: a federated learning framework for On-device anomaly data detection”, arXiv preprint arXiv: 2106.07976, 2021. https://arxiv.org/abs/2106.07976

[12] Yang, T. et al. (7 others), “Applied federated learning: Improving google keyboard query suggestions”, arXiv preprint arXiv:1812.02903, 2018. https://arxiv.org/pdf/1812.02903.pdf

[13] Hard, A. et al (8 others), “Federated learning for mobile keyboard prediction”, arXiv preprint arXiv: 1811.03604, 2018.

https://arxiv.org/pdf/1811.03604.pdf

[14] Beutel, D, J. et al. (10 others), “Flower: A friendly federated learning research framework”, arXiv preprint arXiv:2007.14390, 2020. https://arxiv.org/pdf/2007.14390.pdf

[15] Yang, J. et al. (5 others), “Prototyping federated learning on edge computing systems”, Frontiers Comput. Sci. vol. 14 no. 6: 146318, 2020. https://doi.org/10.1007/s11704-019-9237-3

[16] “An Industrial Grade Federated Learning Framework. Available online”, https://fate.fedai.org/ (accessed on 24 September 2021).

[17] Lo, Sin K. et al. (5 others), “Architectural patterns for the design of federated learning systems”, arXiv preprint arXiv:2101.02373, 2021. https://doi.org/10.48550/arXiv.2101.02373

[18] Bonawitz, K. et al (13 others), “Towards federated learning at scale: System design”, Proceedings of Machine Learning and Systems 1, 374-388. 2019. https://doi.org/10.48550/arXiv.1902.01046

[19] Damaskinos, G. et al. (5 others), “Fleet: Online federated learning via staleness awareness and performance prediction”, Proceedings of the 21st International Middleware Conference. 2020. https://doi.org/10.48550/arXiv.2006.07273

[20] Kairouz, Peter, et al. (58 others), “Advances and open problems in federated learning”, Foundations and Trends® in Machine Learning 14.1–2, 1-210, 2021. https://ml-ops.org/content/references.html

[21] McMahan, B. H., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A., “Communication-efficient learning of deep networks from decentralized data”, in Artificial Intelligence and Statistics. PMLR, 273–1282. https://doi.org/10.48550/arXiv.1602.05629

[22] Google. “Tensorflow federated: Machine learning on decentralized data”, https://www.tensorflow.org/federated, 2021. accessed 25-Dec-21.

[23] https://github.com/PaddlePaddle/PaddleFL

[24] “Three Levels of ML Software”, https://ml-ops.org/content/three-levels-of-ml-software

[25] Nasron Cheong “Design a federated learning system in seven steps”, https://towardsdatascience.com/design-a-federated-learning-system-in-seven-steps-d0be641949c6

[26] https://airflow.apache.org/

[27] Grafana Stack https://grafana.com/

[28] https://wandb.ai/site

[29] https://aws.amazon.com/ko/

[30] https://aws.amazon.com/ko/s3/

[31] https://kubernetes.io/ko/

[32] https://argo-cd.readthedocs.io/en/stable/

[33] https://www.redhat.com/ko/topics/containers/whats-a-linux-container

## ◐ 저 자 소 개 ◑

**홍 승 후(Seung-Hoo Hong)**
2019년～현재 가천대학교 컴퓨터공학과(공학사)
관심분야 : 인공지능, 딥 러닝, 머신러닝, etc.
E-mail : hoo0681@gachon.ac.kr

**이 강 윤(KangYoon Lee)**
1986년 연세대학교 전자공학과(공학사)
1996년 연세대학교 대학원 전자계산학과(공학석사)
2010년 숭실대학교 대학원 IT정책경영학과(공학박사)
1991년～2016년 한국 IBM 연구소(연구소장)
2016년～현재 가천대학교 컴퓨터공학과 교수
관심분야 : 인공지능, IoT, 빅데이터, 클라우드, 커그너티브컴퓨팅, etc.
E-mail : keylee@gachon.ac.kr