KIPS Trans. Comp. and Comm. Sys.
Vol.11, No.10 pp.323~332
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTCCS.2022.11.10.323

A Study on the System for AI Service Production

Yong-Geun Hong†

ABSTRACT

As various services using AI technology are being developed, much attention is being paid to AI service production. Recently, AI technology is acknowledged as one of ICT services, a lot of research is being conducted for general-purpose AI service production. In this paper, I describe the research results in terms of systems for AI service production, focusing on the distribution and production of machine learning models, which are the final steps of general machine learning development procedures. Three different Ubuntu systems were built, and experiments were conducted on the system, using data from 2017 validation COCO dataset in combination of different AI models (RFCN, SSD-Mobilenet) and different communication methods (gRPC, REST) to request and perform AI services through Tensorflow serving. Through various experiments, it was found that the type of AI model has a greater influence on AI service inference time than AI machine communication method, and in the case of object detection AI service, the number and complexity of objects in the image are more affected than the file size of the image to be detected. In addition, it was confirmed that if the AI service is performed remotely rather than locally, even if it is a machine with good performance, it takes more time to infer the AI service than if it is performed locally. Through the results of this study, it is expected that system design suitable for service goals, AI model development, and efficient AI service production will be possible.

Keywords : Artificial Intelligence, Object Detection, AI Inference, AI Production, Edge Computing

인공지능 서비스 운영을 위한 시스템 측면에서의 연구

홍 용 근†

요 약

AI 기술을 활용한 다양한 서비스가 개발되면서, AI 서비스 운영에 많은 관심이 집중되고 있다. 최근에는 AI 기술도 하나의 ICT 서비스를 보고, 범용적인 AI 서비스 운영을 위한 연구가 많이 진행되고 있다. 본 논문에서는 일반적인 기계학습 개발 절차의 마지막 단계인 기계학습 모델 배포 및 운영에 초점을 두고 AI 서비스 운영을 위한 시스템 측면에서의 연구 결과를 기술하였다. 3대의 서로 다른 Ubuntu 시스템을 구축하고, 이 시스템상에서 서로 다른 AI 모델(RFCN, SSD-Mobilenet)과 서로 다른 통신 방식(gRPC, REST)의 조합으로 2017 validation COCO dataset의 데이터를 이용하여 객체 검출 서비스를 Tensorflow serving을 통하여 AI 서비스를 요청하는 부분과 AI 서비스를 수행하는 부분으로 나누어 실험하였다. 다양한 실험을 통하여 AI 모델의 종류가 AI 머신의 통신 방식보다 AI 서비스 추론 시간에 더 큰 영향을 미치고, 객체 검출 AI 서비스의 경우 검출하려는 이미지의 파일 크기보다는 이미지 내의 객체 개수와 복잡도에 따라 AI 서비스 추론 시간이 더 큰 영향을 받는다는 것을 알 수 있었다. 그리고, AI 서비스를 로컬이 아닌 원격에서 수행하면 성능이 좋은 머신이라고 하더라도 로컬에서 수행하는 경우보다 AI 서비스 추론 시간이 더 걸린다는 것을 확인할 수 있었다. 본 연구 결과를 통하여 서비스 목표에 적합한 시스템 설계와 AI 모델 개발 및 효율적인 AI 서비스 운영이 가능해질 것으로 본다.

키워드 : 인공지능, 객체 검출, AI 추론, AI 운영, 에지 컴퓨팅

1. 서 론

인공지능(Artificial Intelligence: AI) 기술이 발전하고 다양한 분야에 적용이 되기 시작하면서, AI 기술을 활용한 다양한 서비스가 개발되고 있다. AI 기술의 핵심인 기계학습(Machine Learning: ML)은 지금까지 대부분 추론(inference)의 정확도(accuracy)를 높이는 방향으로 연구가 진행이 되었다. AI 모델의 성능을 높이기 위해서는 방대한 데이터와 파라미터를 요구하며, 높은 정확도를 가진 AI 모델은 매우 크고 무거운 특성이 있다. 현재 자연어처리 분야에서 가장 뛰어난 알고리즘으로 알려진 OpenAI의 GPT-3 모델에는 총 1,750억 개의 파라미터와 약 5,000억 개의 학습 데이터가 모델 개발에 사용되었다[1].

AI 기술은 실제 생활 현장에 적용하고 기업이 비즈니스 차원

※ 이 논문은 2021년도 정부(과학기술 정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(2021-0-00188, AI 기능 지원 프레임워크 기반의 이기종 IoT 플랫폼 연동 오픈소스 및 국제 표준 개발).
† 정 회 원 : 대전대학교 AI융합학과 교수
Manuscript Received : April 15, 2022
Accepted : May 24, 2022
* Corresponding Author : Yong-Geun Hong(yghong@dju.kr)

에서 AI 서비스를 제공하기 위해서, 개발한 AI 모델을 고객들에게 제공 가능한 수준으로 운영(production)이 되어야 한다. 아무리 성능이 좋은 AI 기술이라고 하더라도, 기업이 서비스나 상품으로 출시 가능한 수준의 가격 내에서 AI 기술을 제공할 수 있어야 하므로 Tensorflow serving[2], TorchServe[3], Nvidia Trion Server[4], Intel OpenVINO[5] 같이 학습이 아닌 추론에 초점을 맞추어 효율적으로 AI 서비스를 제공하려는 기술이 최근에 많이 연구되고 있다.

한편, AI 서비스는 적용하려는 서비스 도메인이 다르고 학습하는 데이터의 특성이 다르면 학습된(training) AI 모델도 달라진다. 또한, AI 기술을 활용하여 해결하려는 문제에 따라서 적용 가능한 AI 기술도 다르므로 범용적으로 AI 기술을 적용하기가 쉽지 않았다. 하지만, 최근에는 AI 서비스를 하나의 ICT 서비스로 보고 AI 서비스를 범용적으로 제공하려는 연구가 많이 진행되었다. 국제표준화 기구인 ITU-T에서 개발된 Y.3531 (Cloud computing - Functional requirements for machine learning as a service, 클라우드 컴퓨팅-서비스형 기계학습 기능 요구사항) 표준에서는 서비스형 기계학습(MLaas, Machine Learning as a Service) 기술에 대하여 정의하고 있다[6]. 일반적으로 기계학습 모델을 개발하기 위해서는 대용량의 학습데이터, 학습을 처리하기 위한 대용량의 리소스, 복잡한 학습 모델 등이 필요하여 온프레미스(on-premise) 환경에서 구축하기에 어려움이 있으므로, 이 표준에서는 클라우드 컴퓨팅 환경에서 간편하고 효과적으로 제공하기 위한 서비스인 서비스형 기계학습에 대하여 정의하였다[7]. ITU-T Y.3531 표준에서는 Fig. 1과 같이 기계학습의 프로세서를 데이터 수집(Data collection)과 데이터 레이블링(Data labelling)을 포함한 기계학습 데이터 수집(ML data acquisition) 단계, 데이터 전처리(Data pre-processing)와 특성 공학(Feature engineering)을 포함하는 기계학습 데이터 처리(ML data processing) 단계, 모델 학습(Model training), 모델 검증(Model validation), 하이퍼 파라미터 튜닝(Hyperparameter turning), 모델 테스트(Model testing)를 포함하는 기계학습 모델 개발(ML model development) 단계, 모델 배포(Model deployment)와 모델 관리(Manage model)를 포함하는 기계학습 모델 배포(ML model deployment) 단계에 관해서 기술하고 있다.

본 논문에서는 Fig. 1에서 기술된 일반적인 기계학습 개발 절차의 마지막 단계인 기계학습 모델 배포 단계에 초점을 두고 인공지능 서비스 운영을 위한 시스템 측면에서 기술하고자 한다. 지금까지의 대부분의 기계학습을 포함한 AI 기술은 Fig. 1의 기계학습 데이터 수집 단계, 기계학습 데이터 처리 단계와 기계학습 모델 개발 단계 위주로 연구가 진행이 되었다. 이것은 ICT 기술의 개발 단계와 비슷하게 AI 기술의 핵심인 추론 정확도가 높은 AI 모델 개발을 위한 연구가 제일 먼저 진행되고, AI 모델 개발이 어느 정도 완성이 되면 이제 개발된 AI 모델을 어떻게 배포하고 운영을 할 것인가가 다음 연구 단계이다. Google Tensorflow에서는 이러한 AI 모델 배포 및 운영 등을 포함한 AI 서비스 파이프라인을 Tensorflow Extended(TFX)로 정의하였다. TFX에서는 운영 기계학습 파이프라인을 배포하기 위하여 end-to -end 플랫폼을 위한 여러 기술을 소개하고 있는데, 본 논문에서는 AI 모델의 배포 및 운영과 가장 밀접한 Tensorflow serving 시스템을 활용하여 인공지능 서비스 운영을 위한 시스템 측면에서의 고려사항을 다양한 실험 결과를 통하여 연구 결과를 소개하고자 한다. 2장에서는 AI 서비스 운영 환경, 에지 컴퓨팅 환경에서의 AI 서비스와 객체 검출(Object Detection) 서비스 등의 관련 연구에 관해서 기술하고, 3장에서는 AI 서비스 중의 하나로 객체 검출 서비스를 선택하고 2017 validation COCO dataset의 데이터를 이용하여 지금까지 주로 연구되었던 AI 모델의 추론 정확도에서 벗어나서 전체 시스템 관점에서 AI 서비스 지연 시간(delay)을 어떻게 줄일 것인가에 대한 연구 결과를 기술한다. 시스템 측면에서의 다양한 현상을 연구하기 위하여, RFCN(Region-based Fully Convolutional Networks), SSD-Mobilenet(Single-Shot MultiBox Detector MobileNet), REST(Representational state transfer), gRPC (Google Remote Procedure Call) 조합으로 클라이언트 머신, 클라우드 서버, 에지 디바이스 등에서 수행한 실험 환경을 기술한다. 4장에서는 실험 결과와 그러한 결과에 대해서 고찰하고, 마지막 5장에서는 결론 및 향후 연구 방향에 관해서 기술한다.

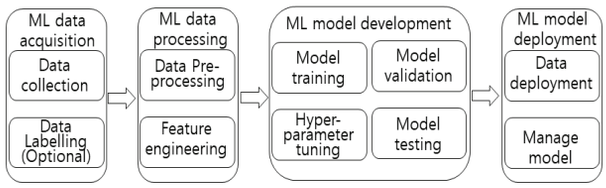

Fig. 1. General Procedure of AI Service Development[6]

2. 관련 연구

2.1 AI 서비스 운영 환경

AI 서비스를 제공하고 운영하는 환경도 AI 데이터 수집 단계, AI 데이터 처리 단계와 AI 모델 개발 단계의 환경과 비슷하게 서비스 도메인과 데이터의 특성, 해결하려는 문제의 특성, 고려하는 시스템 환경에 따라 매우 다양할 수 있다. 하지만, 학습된 AI 모델을 타켓 머신에 올려서 AI 추론 서비스를 제공한다는 측면에서 보면, AI 서비스 운영은 크게 다음과 같은 2가지 경우로 나눌 수 있다.

1) 일반적인 Web 서비스 방식

인터넷을 통하여 다양한 서비스가 제공되면서 여러 형태의 서비스 제공 방식이 논의되고 있지만, 가장 범용적으로 많이 사용되는 방식이 Web 서비스 방식이다. Web 서비스가 전통적인 Web 페이지의 내용을 보여 주는 것에서 벗어나 다양한

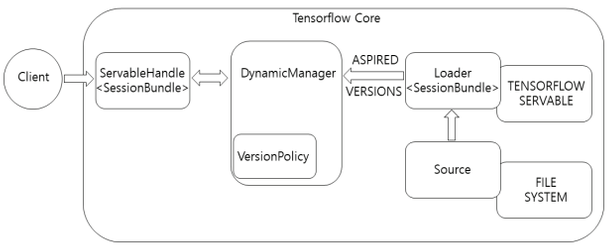

Fig. 2. Architecture of Tensorflow Serving[2]

인터넷 서비스를 Web 서비스를 통하여 제공할 수 있다. AI 서비스 추론 서비스도 이러한 Web 서비스의 형식을 활용하여 제공할 수 있으며, 대표적인 기술로는 Flask[8], Django [9], FastAPI[10] 등이 있다.

2) AI 서비스 추론에 최적화된 방식

AI 추론 서비스 제공을 위하여 일반적인 Web 서비스 방식을 사용하면 범용성과 다양한 환경에서 적용이 가능하다는 장점이 있지만, AI 서비스 추론에는 최적화되어 있지 않기 때문에 추론 시간 등의 성능 면에서는 떨어진다. 그래서, 아예 AI 서비스 추론에 최적화된 방식으로 Tensorflow serving[2], TorchServe[3], Nvidia Trion Server[4], Intel OpenVINO[5] 등과 같이 AI 서비스 운영을 위해서 전문화된 기술이 많이 등장하였다. 본 논문에서는 이 중에서 Tensorflow serving 시스템을 활용하였다.

Tensorflow serving을 포함한 AI 모델 서빙(serving) 시스템은 사용자에게 AI 모델 추론/예측 결과를 효율적으로 전달하기 위하여 개발한 시스템이다. Tensorflow serving은 AI 서비스 운영 환경을 위해 설계되었으며 AI 모델을 고성능으로 적용하는 유연한 시스템을 표방하고 있다. Tensorflow serving을 사용하면 동일한 AI 학습 서버 아키텍처와 API를 유지하면서 새로운 AI 알고리즘과 실험을 쉽게 배포할 수 있다. 또한 Tensorflow serving은 Tensorflow 모델과의 기본적인 통합을 제공하면서도 다른 유형의 AI 모델 및 데이터를 제공하도록 쉽게 확장할 수 있다. Fig. 2는 Tensorflow serving의 구조를 나타낸 것인데, 그림에서 Client는 Tensorflow serving과 동일한 머신에서 실행시켜도 되고, IP주소와 포트번호를 활용하여 원격에서 실행할 수 있다.

2.2 에지 컴퓨팅 환경에서의 AI 서비스

AI 기술은 방대한 양의 데이터를 다루고 복잡한 계산을 수행하는 등의 강력한 컴퓨팅 자원이 필요하므로 지금까지는 클라우드 컴퓨팅같이 인프라를 갖춘 백엔드(Back-end) 환경에서 주로 발전해 왔다. IoT 기술이 AI 기술과 결합하던 초기에도 데이터 대부분은 클라우드 서버에 전달이 되고, 클라우드 서버에서 AI 기술이 수행되어 그 결과를 IoT 디바이스가 전달받아 보여 주는 형태로 진행이 되었다.

에지 컴퓨팅 기술은 다양한 단말들이 생성하는 데이터를 클라우드와 같은 중앙 집중식 데이터 센터로 보내지 않고 단말 또는 사용자 근처에서 실시간으로 처리하는 방식으로 불필요한 데이터 전송을 최소화하고 데이터 흐름 가속화를 지원하는 컴퓨팅 방식이다. 최근 반도체 공정과 메모리 성능 향상, AI 하드웨어의 발전은 에지 컴퓨팅 기기의 연산 성능 향상과 가격 하락을 촉진하였고, 에지 컴퓨팅 기술을 확산시키는 요소가 되었다[11].

에지 컴퓨팅 기술이 발전하고 에지 기기의 하드웨어와 소프트웨어 수준이 에지 기기에서 지원할 만큼 강력해짐에 따라 클라우드 서버로 데이터를 전달할 필요 없이 에지 기기에서 AI 서비스 수행이 가능해졌다. 클라우드 서버가 아니라 에지 디바이스와 같은 작은 디바이스 상에서 사전에 훈련된 AI 딥러닝(Deep Learning) 모델을 이용하여 AI 추론 서비스를 제공하는 기술을 온디바이스(On-device) 딥러닝이라고 한다. 이러한 온디바이스 딥러닝은 전통적인 클라우드 컴퓨팅 기반 딥러닝에 비하여 다음과 같은 이점이 존재한다.

첫째, 실시간성이 중요한 AI 서비스에 적용할 수 있다. 자율주행차나 드론같이 클라우드 서버와 통신하며 AI 서비스를 수행하기에는 지연 시간이 매우 중요한 응용에서는 클라우드 서버와 통신하지 않고 에지 디바이스에서 AI 서비스 수행이 가능하므로 실시간 성능이 우수하다[12]. 또한 에지 디바이스에서 AI 서비스를 수행하면 네트워크 대역폭을 효율적으로 활용할 수 있으며, 기존 네트워크의 혼잡 정도를 낮추고 시스템 용량을 줄일 수 있다[13].

둘째, 민감하거나 보안이 필요한 AI 서비스의 경우 데이터 정보의 보안성을 유지할 수 있다. 만약 클라우드 서버에서 AI 서비스를 수행하면 클라우드 서버로 데이터를 전송해야 하므로 정보 유출의 가능성이 존재한다. 반면에 에지 디바이스에서 AI 서비스를 수행할 때 네트워크 상에서 정보의 유출 없이 AI 서비스 제공이 가능하다.

셋째, 다양한 서비스 도메인에 낮은 가격의 하드웨어로 AI 서비스 제공이 가능하다. 현재 에지 디바이스에 특화된 AI 모델을 만들기 위하여 기존의 높은 성능의 서버를 위해 개발된 딥러닝 모델을 에지 디바이스에 특화된 모델로 경량화하여 에지 디바이스에서 AI 서비스 제공이 가능하다[14].

에지 컴퓨팅 환경에서 AI 서비스를 제공하면 네트워크 트래픽과 서비스 지연 시간을 줄이는 효과는 있지만, 에지 디바이스에서 사용 가능한 컴퓨팅 자원은 클라우드 컴퓨팅과 비교했을 때 현저히 낮다. 온디바이스 딥러닝에 대한 산업계의 수요가 증가하고 있지만, 에지 디바이스에서 컴퓨팅 자원 제약으로 기존의 고성능의 서버를 위해 개발된 딥러닝 모델의 활용이 불가능해 낮은 성능을 가진 에지 컴퓨팅 환경에서 AI 서비스 제공을 가능하게 하는 경량 AI 딥러닝 모델 개발이 활발하게 진행되고 있다[15-17].

2.3 객체 검출 서비스

본 논문에서는 AI 서비스 운영을 위한 시스템 측면에서의 연구를 위하여 AI 서비스의 하나의 예로써 객체 검출 서비스

를 이용하였다. 객체 검출(Object detection) 서비스는 어떤 물체인지 분류하는 분류(Classification) 문제와 물체의 위치 정보를 로컬리제이션(Localization) 문제를 해결하는 분야이다. 딥러닝 기반 객체 검출 방식은 크게 영역제안 기반의 two-stage 기법과 회귀 기반의 one-stage 기법이 있다[18, 19].

Two-stage 기법의 대표적인 기법은 Region-based Convolution Neural Network(R-CNN)[20], Fast R-CNN[21], Faster R-CNN[22], Region-based Fully Convolution Neural Network(R-FCN)[23], light head R-CNN 방법 및 컨볼류션 신경망에 기반한 기타 개선 기법[24, 25] 등이 존재한다.

One-stage 기법의 대표적인 기법은 YoLo(You Only Look Once) 시리즈 방법[26-28], SSD(Single Shot multibox Detector)[29], Local Loss[30], RefineDet[31] 등이 존재한다.

객체 검출 서비스는 물체의 위치를 찾는 문제와 찾은 물체를 식별하는 문제를 모두 해결해야 한다. One-stage 기법은 이 두 문제를 동시에 해결하는 방법이고, two-stage 기법은 두 문제를 순차적으로 해결하는 방법이다. 따라서 two-stage 기법은 one-stage 기법보다 정확도는 높지만 검출 속도에서는 one-stage보다 느리다[32, 33]. 따라서, 만약 높은 정확도를 요구하는 객체 검출 서비스에는 two-stage 기법을 선택하고, 높은 정확도 보다는 빠른 검출 속도를 요구하는 객체 검출 서비스에서는 one-stage 기법을 선택하여 객체 검출 서비스를 설계하는 것이 바람직하다.

본 논문에서는 객체 검출 서비스를 이용하여 AI 서비스 시스템을 구축할 시 다양한 측면을 연구하기 위하여 two-stage 기법의 대표적인 기법인 RFCN과 one-stage 기법의 대표적인 SSD-Mobilenet[34] 기법을 선택하여 연구에 활용하였다. RFCN 모델은 FCN을 backbone으로 사용하지만, Position-sensitive Score Map을 사용하고 Region Proposal Network 이후의 단계를 수정하여 전체 속도를 향상시킨 two-stage 기법이다. SSD-Mobilenet 모델은 주어진 이미지에서 객체의 경계 상자를 설정하고 객체의 분류 문제를 한번에 수행하는 Single Shot Detector이며, Mobilenet을 backbone으로 사용하는 one-stage 기법이다.

3. 실험 환경

3.1 전체 시스템 구성

본 논문에서는 AI 서비스의 하나의 예로서 객체 검출 서비스를 이용하여 AI 기술을 활용한 시스템을 구축하고, 이 시스템상에서 AI 서비스를 운영할 시 고려해야 할 사항을 연구하고자 한다. 특히 본 논문에서는 지금까지 AI 기술 연구에서 주로 연구되었던 모델의 성능을 추론의 정확도에서 벗어나서 전체 시스템 관점에서 AI 서비스 지연 시간을 어떻게 줄일 것인가에 초점을 두고 연구를 진행하였다. 그래서, 기본적으로 본 논문에서 구축한 AI 시스템의 구성은 다음과 같다.

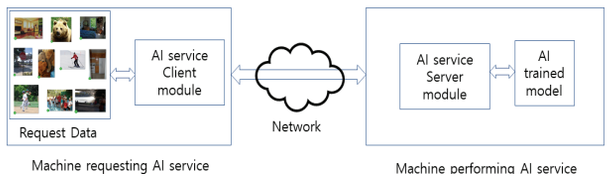

Fig. 3. System for AI Service Production

Fig. 3에서 보는 바와 같이, AI 서비스를 요청하는 머신과 AI 서비스를 수행하는 머신은 논리적으로 구분하였다. 이 구분은 논리적인 것으로써, AI 서비스를 요청하는 머신과 AI 서비스를 수행하는 머신이 같은 머신일 수도 있고, 서로 다른 머신일 수 있다(이것은 AI 서비스를 요청하는 머신에서 AI 서비스를 수행하는 머신의 IP 주소를 어떻게 설정하느냐에 따라 쉽게 설정할 수 있다). AI 서비스를 요청하는 머신은 실제로 AI 서비스를 요청하는 클라이언트와 요청 데이터로 구성된다. AI 서비스를 수행하는 머신은 실제로 AI 서비스를 수행하는 서버와 AI 서비스를 수행하기 위한 AI 학습 모델로 구성된다. 본 논문에서는 객체 검출 서비스와 2017 validation COCO dataset을 이용하였기 때문에, 별도의 데이터 전처리 모듈은 필요하지 않았다. 만약 다른 AI 서비스와 데이터를 사용한다면 AI 서비스를 요청하는 머신이나 AI 서비스를 수행하는 머신에서 AI 서비스 추론 서비스를 수행하기 위해 별도의 전처리 모듈이 필요할 수 있다.

본 논문에서는 다음과 같은 3개의 머신을 사용하였고, 모두 Ubuntu 20.0.4를 설치하여 실험에 활용하였다. 머신-1은 Intel Core i7-10510U CPU를 사용하는 머신(노트북)으로 Geekbench 수행 결과에서 Single-core 점수는 1,175, Multi-core 점수는 3,589점이 나왔다. 머신-2는 Intel Core i7-10700K CPU를 사용하는 머신(일반 PC)으로 Geekbench 수행 결과에서 Single-core 점수는 1,465, Multi-core 점수는 8,078점이 나왔다. 머신-3은 Intel Core i7-1165G7 CPU를 사용하는 머신(노트북)으로 Geekbench 수행 결과에서 Single-core 점수는 1,660, Multi-core 점수는 5,617점이 나왔다. Geekbench 수행 결과에서 나온 시스템 성능을 보면, 3개의 머신 중 머신-2가 가장 높은 성능을 보이므로, 이 머신은 클라우드 서버로 활용하였고, 가장 낮은 성능을 보이는 머신-1은 에지 디바이스로 활용하였다. 그리고 중간 성능을 보이는 머신-3은 AI 서비스를 요청하는 클라이언트 머신으로 활용하였다.

3.2 객체 검출을 위한 AI 학습 모델

객체 검출 서비스 제공을 위하여 핵심인 AI 모델은 Intel AI github에서 제공하는 다음의 2가지 모델을 사용하였다[35]. 이 논문에서는 학습된 AI 모델을 가지고 AI 서비스 운영과 관련된 연구를 진행하는 것이므로, 별도의 학습 과정은 진행하지 않았고 Intel AI github에서 제공되는 미리 학습된 AI 모델을 다운 받아서 사용하였다.

```
docker run \
    --name=tfserving \
    -d \
    -p 8500:8500 \
    -p 8501:8501 \
    -v "$(pwd)/obj_detection:/models/$model_name" \
    -e MODEL_NAME=$model_name \
    -e OMP_NUM_THREADS=$num_physical_cores \
    -e TENSORFLOW_INTER_OP_PARALLELISM=2 \
    -e TENSORFLOW_INTRA_OP_PARALLELISM=$num_physical_cores \
    intel/intel-optimized-tensorflow-serving:latest
```

Fig. 4. Code for Execution of Tensorflow Serving

- RFCN (Region-based Fully Convolutional Networks)
- SSD-Mobilenet (Single-Shot MultiBox Detector MobileNet)

실험에서 사용하는 머신은 모두 Intel CPU를 사용함으로, Intel CPU 측면에서 최적화된 인공지능 모델을 사용하고자 하였다. 또한 Intel AI에서 제공하는 다음과 같은 파라미터 설정을 통하여 최적의 인공지능 서비스를 제공하고자 하였다.

- OMP_NUM_THREADS : 가능한 CPU의 쓰레드 개수
- TENSORFLOW_INTER_OP_PARALLELISM : 2
- TENSORFLOW_INTRA_OP_PARALLELISM : 가능한 CPU의 쓰레드 개수

3.3 AI 서비스 요청 및 수행을 위한 모듈

AI 서비스는 Flask, Django 같은 범용적인 Web 서버 서비스를 이용하여 제공할 수 있지만, Google Tensorflow serving과 같이 AI 서비스 운영 환경을 위해 머신러닝 모델을 고성능으로 유연하게 적용이 가능한 시스템인 Tensorflow serving을 이용하였다.

Tensorflow serving은 AI 학습 모델을 가지고 요청을 받으면 실제로 AI 추론 서비스를 수행하는 서버 모듈과 데이터를 가지고 AI 추론 서비스를 요청하는 클라이언트 모듈로 나눌 수 있다. AI 추론 서비스를 수행하는 서버 모듈에서는 Tensorflow serving을 Docker container를 이용하여 설치하였고, 다음 Fig. 4와 같이 커맨드를 사용하여 Tensorflow serving을 실행시켰다.

AI 추론 서비스를 요청하는 클라이언트 모듈에서는 Tensorflow Object Detection API를 설치하였고, Intel AI github에서 제공하는 'object_detection_benchmark. py' 모듈을 수정하여 실험에 활용하였다.

3.4 AI 서비스 요청에 사용한 데이터

AI 서비스 시스템의 성능을 측정하기 위하여 사용한 데이터는 AI 학습 모델을 만들 때 사용한 데이터를 그대로 사용하였고, 그 데이터는 780MB 크기의 2017 validation COCO dataset이다. 이 데이터셋 중에서 10개의 이미지를 선정하여

Fig. 5. Original Images for AI Service Request

객체 검출 서비스를 각각 RFCN과 SSD-Mobilenet 모델을 활용했을 때 추론 성능을 비교하였다(Fig. 5).

4. 실험 결과 및 고찰

4.1 실험에 사용한 데이터와 AI 모델에 대한 성능 분석

이번 장에서는 앞 장에서 설명한 2017 validation COCO dataset의 10개의 이미지에 대하여 로컬 머신에서 RFCN 모델과 SSD-Mobilenet 모델을 각각 AI 추론 서비스의 AI 모델로 활용했을 때 어떤 차이가 있는지 실험 결과를 설명한다. RFCN 모델은 two-stage 객체 검출 서비스의 기법으로 추론 정확도는 높지만, 검출 속도는 느리다고 알려져 있다. 반면 SSD-Mobilenet의 경우 one-stage 객체 검출 서비스의 기법으로 검출 속도는 빠르지만, 추론 정확도는 낮은 것으로 알려져 있다. 다음 그림 각각은 동일한 이미지 데이터 10개를 각각 RFCN 모델과 SSD-Mobilenet 모델을 활용하여 객체 검출 서비스를 수행하였을 때, 검출 결과를 나타낸 것이다.

Fig. 6과 Fig. 7의 결과를 비교해 보면, RFCN 모델로 객체 검출 서비스를 수행하면 이미지상에 있는 대부분의 객체

Fig. 6. Result of Object Detection with RFCN

Fig. 7. Result of Object Detection with SSD-Mobilenet

를 높은 정확도(95% 이상)로 검출한 것을 알 수 있지만, SSD-Mobilenet 모델로 객체 검출 서비스를 수행하면 이미지상에 있는 주요 객체만 검출하고 작은 객체는 검출하지 못하며, 주요 객체의 검출 정확도도 RFCN 모델보다는 떨어지는 것을 알 수 있다. 예를 들어 첫 번째 이미지의 경우 RFCN 모델의 경우 TV를 95%의 정확도와 의자를 99%의 정확도로 검출하지만, SSD-Mobilenet 모델의 경우 TV를 84%의 정확도와 의자를 65%의 정확도로 검출하였다.

4.2 각각의 머신에서 수행한 AI 서비스 추론 시간 비교

이번에는 같은 10개의 이미지 데이터를 가지고 3개의 서로 다른 머신(클라이언트 머신, 클라우드 서버, 에지 디바이스) 로컬에서 객체 검출 서비스를 수행했을 경우 각각의 추론 시간을 비교하였다. 본 논문에서는 AI 서비스 클라이언트 모듈에서 AI 서비스 서버 모듈로 객체 검출 서비스를 요청하면 통신 방법으로 REST 기법과 gRPC 기법을 선택하여 사용하였다. REST 기법은 일반적인 Web 기반 환경에서 분산 하이퍼미디어 시스템을 위한 소프트웨어 아키텍쳐의 한 형식이며, gRPC 기법은 구글에서 대용량의 데이터 전송에 적합한 기법으로 알려져 있다. 반면 SSD-Mobilenet의 경우 one-stage 객체 검출 서비스의 기법으로 검출 속도는 빠르지만, 추론 정확도는 낮은 것으로 알려져 있다. 다음 그림 각각은 동일한 이미지 데이터 10개를 각각 RFCN 모델과 SSD-Mobilenet 모델을 활용하여 객체 검출 서비스를 수행하였을 때, 검출 결과를 나타낸 것이다.

Fig. 8, Fig. 9, Fig. 10에서 보면 클라이언트 머신, 클라우드 서버, 에지 디바이스 각각 로컬에서 10개의 이미지별 객체 검출 서비스 추론 시간을 비교하여 다음과 같은 평균값을 도출하였다.

Table 1을 보면, 가장 높은 CPU 성능을 보이는 클라우드 서버에서 객체 검출 평균 시간이 제일 짧았고, 그다음으로 중간 정도 CPU 성능을 보이는 클라이언트 머신, 그리고 가장 낮은 CPU 성능을 보이는 에지 디바이스에서 객체 검출

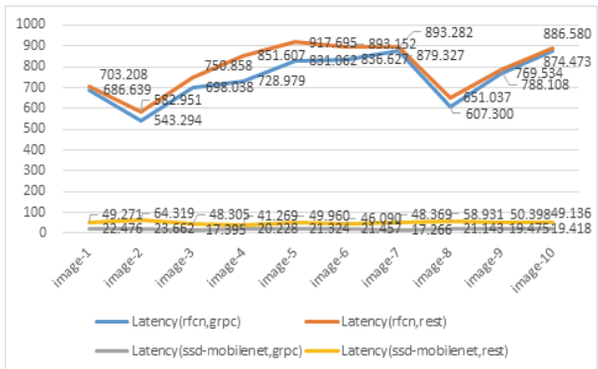

Fig. 8. Latency of Object Detection in Client Machine

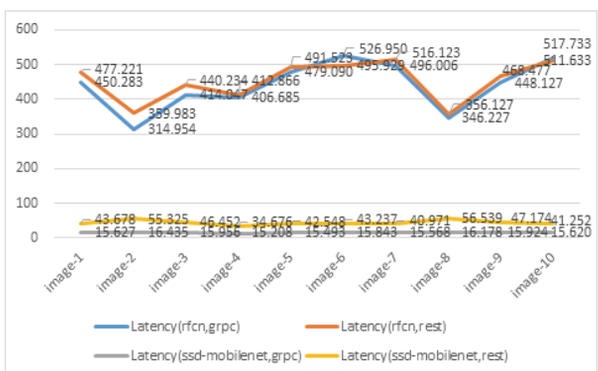

Fig. 9. Latency of Object Detection in Cloud Server

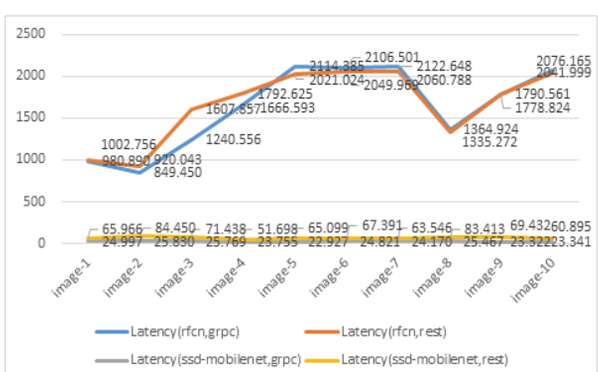

Fig. 10. Latency of Object Detection in Edge Device

Table 1. Average Latency Time of Each Machine

	Client machine	Cloud server	Edge device
Latency(ms) (rfcn, grpc)	745.5273	440.0102	1631.2673
Latency(ms) (rfcn, rest)	791.8478	453.0116	1661.1157
Latency(ms) (ssd-mobilnet, grpc)	20.3844	15.7852	24.4399
Latency(ms) (ssd-mobilnet, rest)	50.6048	45.1852	68.3328

평균 시간이 제일 오래 걸렸다. 또한, 클라이언트 머신, 클라우드 서버, 에지 디바이스 각각에서 RFCN, SSD-Mobilenet, gRPC, REST 4가지 조합을 비교해 보면 모든 머신에서 (SSD-Mobilenet, gRPC) 조합이 객체 검출 평균 시간이 제일 짧게 걸렸고, 그다음으로 (SSD-Mobilenet, REST) 조합, (RFCN, gRPC) 조합, (RFCN, REST) 조합 순으로 객체 평균 시간이 짧게 걸렸다.

Table 1의 결과를 분석해 보면 다음과 같은 결과를 도출할 수 있었다.

- 객체 검출 서비스에서 사용하는 AI 모델의 종류(RFCN, SSD-Mobilenet)가 객체 검출 시간에 미치는 영향이 AI 서비스 클라이언트와 AI 서비스 서버간의 통신 방식의 종류(gRPC, REST) 보다 더 크다는 것을 확인할 수 있었다. 이것은 일반적으로 알려진 객체 검출 서비스의 two-stage 기법과 one-stage 기법의 특성과 유사한 결과를 보인다.
- 같은 AI 모델을 사용할 경우, REST 통신 방식이 gRPC 통신 방식보다 객체 검출 시간이 더 많이 걸리는 것을 확인할 수 있었다. 이것도 일반적으로 알려진 REST 통신 방식과 gRPC 통신 기법의 특성과 유사한 결과를 보인다.

4.3 로컬과 원격 머신에서 수행한 AI 서비스 추론 시간 비교

이번에는 평균적으로 객체 검출 시간이 가장 적게 걸리는 Image-2 이미지와 객체 검출 시간이 가장 많이 걸리는 Image-7 이미지에 대하여 객체 검출 서비스를 요청하는 머신은 클라이언트로 설정하고, 객체 검출 서비스를 수행하는 머신은 클라이언트 머신, 클라우드 서버, 에지 디바이스로 설정했을 때의 각각 객체 검출 시간을 비교하였다. 다음 그림과 같은 구성으로 실험을 진행하였다(Fig. 11, Fig. 12, Fig. 13). 즉 AI 서비스를 요청하는 'AI 서비스 클라이언트 모듈'은 클라이언트 머신에서 동일하게 수행하고, 객체 검출 서비스를 직접 수행하는 'AI 서비스 서버 모듈'은 각각 클라이언트 머신, 클라우드 서버, 에지 디바이스에서 동작하도록 하고, 객체 검출 시간을 측정하였다.

위에서 설명한 3개의 환경에서 Image-2 이미지와 Image-7 이미지 각각의 객체 검출 실험 결과는 다음 그림과 같다.

Fig. 14와 Fig. 15에서 나온 결과를 바탕으로 클라이언트 머신, 클라우드 서버, 에지 디바이스 각각에서 원격으로 수행한 2개의 이미지별 객체 검출 시간을 비교하여 다음과 같은 평균값을 도출하였다.

Table 2와 Table 3을 보면, Table 1과 다르게 동일한 Image-2와 Image-7 이미지에 대하여 클라이언트 머신에서 객체 검출 평균 시간이 제일 짧았고, 그 다음으로 클라우드 서버, 그리고 에지 디바이스 순으로 객체 검출 평균 시간이 증가하였다. 이것은 가장 높은 CPU 성능을 보이는 클라우드 서버라도 로컬이 아닌 원격으로 AI 서비스를 수행하고 그 결과를 받는 식으로 진행이 되기 때문에, 로컬에서 AI 서비스를

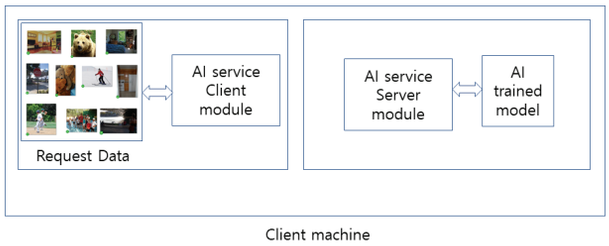

Fig. 11. AI Service Execution in Client Machine

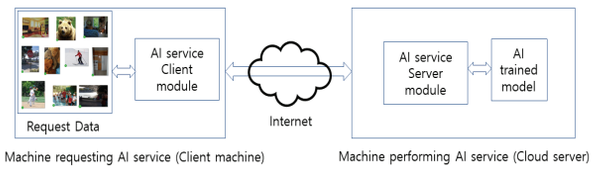

Fig. 12. AI Service Execution in Cloud Server

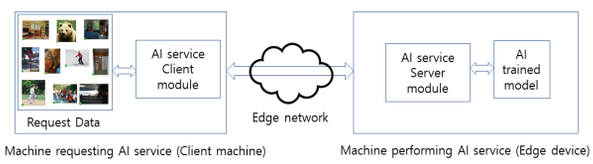

Fig. 13. AI Service Execution in Edge Device

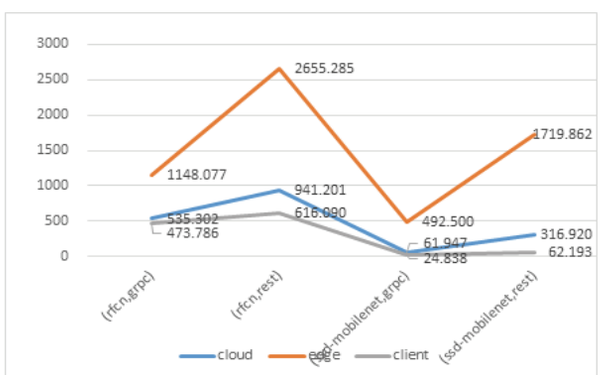

Fig. 14. Latency of Object Detection in Client Machine, Cloud Server, Edge Device of Image-2

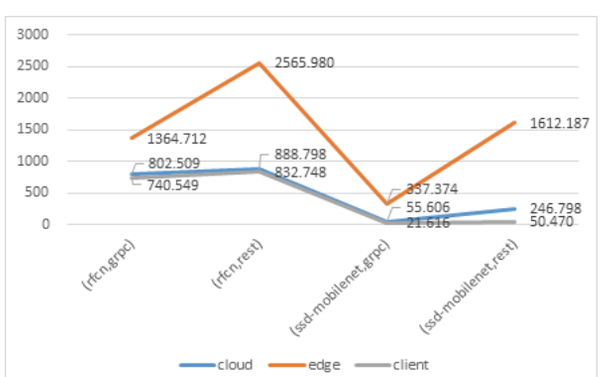

Fig. 15. Latency of Object Detection in Client Machine, Cloud Server, Edge Device of Image-7

Table 2. Average Latency Time of Each Machine of Image-2 in Remote

	Client machine	Cloud server	Edge device
Latency(ms) (rfcn, grpc)	473.786	535.302	1148.077
Latency(ms) (rfcn, rest)	616.09	941.201	2655.285
Average Latency(ms) (rfcn)	544.938	738.252	1901.681
Latency(ms) (ssd-mobilnet, grpc)	24.838	61.947	492.5
Latency(ms) (ssd-mobilnet, rest)	62.193	24.838	1719.862
Average Latency(ms) (ssd-mobilenet)	43.516	43.393	1106.181
Average Latency (ms)	294.227	390.822	1503.931

Table 3. Average Latency Time of Each Machine of Image-7 in Remote

	Client machine	Cloud server	Edge device
Latency(ms) (rfcn, grpc)	740.549	802.509	1364.712
Latency(ms) (rfcn, rest)	832.748	888.798	2565.98
Average Latency(ms) (rfcn)	786.649	845.654	1965.346
Latency(ms) (ssd-mobilnet, grpc)	21.616	55.606	337.374
Latency(ms) (ssd-mobilnet, rest)	50.47	246.798	1612.187
Average Latency(ms) (ssd-mobilenet)	36.043	151.202	974.781
Average Latency (ms)	411.346	498.428	1470.063

직접 수행하는 클라이언트 머신과 비교하여 객체 검출 시간이 더 걸리게 되는 것이다. Table 2와 Table 3에서 보면 클라이언트 머신, 클라우드 서버, 에지 디바이스에서의 RFCN, SSD-Mobilenet, gRPC, REST 4가지 조합을 비교해 보면 모든 머신에서 (SSD-Mobilenet, gRPC) 조합이 객체 검출 평균 시간이 제일 짧게 걸렸고, 그다음으로 (SSD-Mobilenet, REST) 조합, (RFCN, gRPC) 조합, (RFCN, REST) 조합 순으로 평균 시간이 짧게 걸렸다.

또, Table 2와 Table 3을 보면 Image-2 이미지는 평균적으로 객체 검출 시간이 제일 적게 걸렸고, Image-7 이미지는 평균적으로 객체 검출 시간이 제일 많이 걸렸다. Image-2 이미지의 파일 크기는 335,851 Byte이고, Image-7 이미지의 파일 크기는 62,406 Byte이다. 객체 검출 시간은 Image-2 이미지가 4가지 조합 평균적으로 클라이언트 머신의 경우 294,227ms, 클라우드 서버의 경우 390.822ms, 에지 디바이스 경우의 경우 1503.931ms가 걸렸다. Image-7 이미지의 객체 검출 시간은 4가지 조합 평균적으로 클라이언트 머신의 경우 411,346ms, 클라우드 서버의 경우 498.428ms, 에지 디바이스의 경우 1470.063ms가 걸렸다. Image-2와 Image-7 2개의 이미지를 비교하면, 파일 크기는 Image-2 이미지가 Image-7 이미지보다 대략 5.4배 더 크지만, 오히려 객체 검출을 위한 추론 시간은 Image-2가 훨씬 적게 걸리는 것을 확인할 수 있었다. 이것은 객체 검출을 위한 AI 모델의 특성에 따른 것으로 본 논문에서는 자세히 다루지 않는다. 이러한 현상은 AI 모델에 따라 달라질 수 있다. 본 실험에서 한 가지 흥미로운 현상은 4가지 조합의 Image-7 이미지의 평균 추론 시간은 Image-2 이미지의 평균 추론 시간보다 크고 이러한 현상은 클라이언트 머신과 클라우드 서버에서는 동일하게 나타났지만, 에지 디바이스의 경우는 미세하지만 오히려 반대 결과가 나왔다.

Table 2와 Table 3의 결과에서 다음과 같은 결과를 도출할 수 있었다.

- 객체 검출 서비스에서 AI 서비스를 수행하는 위치가 로컬이 아닌 원격이 되면, 성능이 좋은 CPU를 사용하는 머신이라도 로컬에서 수행하는 경우보다 추론 시간이 더 걸린다.
- 본 실험에서 사용한 RFCN 모델과 SSD-Mobilenet의 AI 모델의 경우, 추론에 걸리는 시간은 이미지 파일의 크기보다는 검출해야 할 객체의 개수와 복잡도에 따라 결정되는 것으로 추정된다. 또한 이러한 현상은 3개의 머신 중 가장 높은 성능을 보이는 클라우드 서버와 클라이언트 머신에서는 동일한 결과를 보이지만, 가장 낮은 성능을 보이는 에지 디바이스에서는 다른 결과를 보여 주었다.
- 객체 검출 서비스에서 사용하는 AI 모델의 종류(RFCN, SSD-Mobilenet)가 객체 검출 시간에 미치는 영향이 AI 서비스 클라이언트와 AI 서비스 서버간의 통신 방식의 종류(gRPC, REST) 보다 더 크다는 것을 확인할 수 있었다. 이것은 AI 서비스 수행을 로컬에서 하거나 원격에서 하거나 동일한 결과를 보였다.
- 같은 AI 모델을 사용할 경우, REST 통신 방식이 gRPC 통신 방식보다 객체 검출 시간이 더 많이 걸리는 것을 확인할 수 있었다. 이것도 AI 서비스 수행을 로컬에서 하거나 원격에서 하거나 동일한 결과를 보였다.

5. 결론 및 향후 연구

본 논문에서는 일반적인 기계학습 개발 절차의 마지막 단계인 기계학습 모델 배포 및 운영에 초점을 두고 AI 서비스 운영을 위한 시스템 측면에서의 연구 결과를 기술하였다. 3대의 서로 다른 Ubuntu 시스템을 구축하고, 이 시스템상에서 서로 다른 AI 모델(RFCN, SSD-Mobilenet)과 서로 다른 통신 방식(gRPC, REST)의 조합으로 2017 validation

COCO dataset의 데이터를 이용하여 객체 검출 서비스를 Tensorflow serving을 통하여 AI 서비스를 요청하는 부분과 AI 서비스를 수행하는 부분으로 나누어 실험을 진행하였다.

다양한 실험을 통하여 AI 모델의 종류가 AI 머신의 통신 방식보다 AI 서비스 추론 시간에 더 큰 영향을 미치고, 객체 검출 AI 서비스의 경우 검출하려는 이미지의 파일 크기보다는 이미지 내 객체 개수와 복잡도에 따라 AI 서비스 추론 시간이 더 큰 영향을 받는다는 것을 알 수 있었다. 그리고, AI 서비스를 로컬이 아닌 원격에서 수행하면 성능이 좋은 머신이라고 하더라도 로컬에서 수행하는 경우보다 AI 서비스 추론 시간이 더 걸린다는 것을 확인할 수 있었다.

본 논문에서는 일반적으로 알려진 객체 검출 서비스의 two-stage 기법과 one-stage 기법의 특성 및 gRPC와 REST 기법의 특성이 그대로 나타나는 것을 확인할 수 있었다. 하지만, 같은 AI 모델을 사용하는 경우에 통신 방식에 따른 결과의 차이와 같은 통신 방식을 사용하는 경우 AI 모델에 따른 결과의 차이는 다루지 못했다. 이 부분은 향후 연구에서 밝히고자 한다. 또한 논문에서 기술한 바와 같이 효율적인 AI 추론 서비스를 구현하기 위해서는 Tensorflow serving 같은 AI 서비스 추론에 최적화된 방식과 함께 Flask, Django, FastAPI 등과 같은 범용적인 Web 서비스 방식을 사용할 수 있는데, 향후 연구에서는 각각의 방식대로 구현하고 장단점을 밝히고자 한다. 마지막으로 본 논문에서는 클라우드 서버와 에지 디바이스의 경우 CPU 및 RAM 성능 차이에 따른 AI 서비스 추론 시간 결과는 확인할 수 있었지만, 일반적으로 에지 디바이스보다 먼 거리에 있는 클라우드 서버와의 통신 지연 시간은 고려하지 못하였다. 향후 연구에서는 클라우드 서버와 에지 디바이스 간의 통신 거리에 따른 AI 서비스 추론 시간의 영향을 살펴보고자 한다.

References

[1] T. Brown et al., "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, Vol.33, pp.1877-1901, 2020.

[2] Tensorflow serving [Internet], https://www.tensorflow.org/tfx/guide/serving.

[3] TorchServe [Internet], https://pytorch.org/serve/.

[4] Nvidia Trion Server [Internet], https://developer.nvidia.com/nvidia-triton-inference-server.

[5] Intel OpenVINO [Internet], https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html.

[6] ITU-T Y.3531, "Cloud computing - Functional requirements for machine learning as a service," 2020.

[7] Sungpil Shin, "MLaas(Machine Learning as a Service) Market Trend and Standards for functional requirement," TTA ICT Standard Weekly 1065, 2022.

[8] Flask [Internet], https://flask.palletsprojects.com/en/2.0.x.

[9] Django [Internet], https://www.djangoproject.com/.

[10] FastAPI [Internet], https://fastapi.tiangolo.com/.

[11] H. M. Park and T. H. Hwang, "Changes and trends of Edge computing technology," *KICS Information and Communication Magazine*, Vol.36, No.2, pp.41-47, 2019.

[12] W. Yu, F. Liang, X. He, W. Grant Hatcher, C. Lu, J. Lin and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, Vol.6, pp.6900-6919, 2017.

[13] S. Maheshwari, D. Raychaudhuri, I. Seskar, and F. Bronzino, "Scalability and performance evaluation of edge cloud systems for latency constrained applications," In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 286-299. IEEE, 2018.

[14] K. H. Kim, Y. G. Hong, and C. S. Pyo, "Standard technology and Trend of Edge computing for IoT and AI," *KICS Information and Communication Magazine*.

[15] E. H. Kim, K. Ha Lee, and W. Kyung Sung, "Technology trends of deep-learning model lightweight," *Communication of KIISE*, Vol.38, No.8, pp.18-29, 2020.

[16] F. Wang, M. Zhang, X. Wang, X. Ma, and J. Liu, "Deep learning for edge computing applications: A state-of-the-art survey," *IEEE Access*, Vol.8, pp.58322-58336, 2020.

[17] Y. Jun Choi and H. S. Eom, "Deep learning model compression for embedded system," *KIISE KCC 2019*, pp.1044-1046, 2019.

[18] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[19] M. Algabri, H. Mathkour, M. Abdelkader Bencherif, M. Alsulaiman, and M. Amine Mekhtiche, "Towards deep object detection techniques for phoneme recognition," *IEEE Access*, Vol.8, pp.54663-54680,2020.

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.580-587, 2014.

[21] R. Girshick, "Fast r-cnn," In *Proceedings of the IEEE International Conference on Computer Vision*, pp.1440-1448. 2015.

[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, Vol.28, 2015.

[23] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," *Advances in Neural Information Processing Systems*, Vol.29, 2016.

[24] S. H. Park, H. S. Yoon, and K. R. Park, "Faster R-CNN and geometric transformation-based detection of driver's eyes using multiple near-infrared camera sensors," *Sensors*, Vol.19, No.1, pp.197, 2019.

[25] K. Surya Vara Prasad, K. B. D'souza, and V. K. Bhargava, "A downscaled faster-RCNN framework for signal detection and time-frequency localization in wideband RF systems," *IEEE Transactions on Wireless Communications*, Vol.19, No.7, pp.4847-4862,2020.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.779-788, 2016.

[27] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.7263-7271, 2017.

[28] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C-Y Fu, and A. C. Berg, "Ssd: Single shot multibox detector," In *European Conference on Computer Vision*, pp.21-37. Springer, Cham, 2016.

[30] T-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," In *Proceedings of the IEEE International Conference on Computer Vision*, pp.2980-2988, 2017.

[31] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-shot refinement neural network for object detection," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.4203-4212, 2018.

[32] L. Zhou, W. Min, D. Lin, Q. Han, and R. Liu, "Detecting motion blurred vehicle logo in IoV using filter-DeblurGAN and VL-YOLO," *IEEE Transactions on Vehicular Technology*, Vol.69, No.4, pp.3604-3614, 2020.

[33] H. Zhang, L. Qin, J. Li, Y. Guo, Y. Zhou, J. Zhang, and Z. Xu, "Real-time detection method for small traffic signs based on Yolov3," *IEEE Access,* Vol.8, pp.64145-64156, 2020.

[34] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[35] Intel AI Object Detection [Internet], https://github.com/IntelAI/models/blob/master/docs/object_ detection/tensorflow_serving/Tutorial.md.

홍 용 근

https://orcid.org/0000-0003-2974-3820
e-mail : yghong@dju.kr
1997년 경북대학교 컴퓨터공학과(학사)
1999년 경북대학교 컴퓨터공학과(석사)
2013년 경북대학교 컴퓨터공학과(박사)
2001년 ~ 2020년 한국전자통신연구원 실장
2021년 ~ 현 재 대전대학교 AI융합학과 교수
관심분야 : IoT, 지능형 에지 컴퓨팅, 추론 시스템