모바일 애플리케이션 엣지환경 구축

지도교수 : 허의남

컴퓨터 공학과 2016110307 권동영

요 약

모바일 엣지 컴퓨팅은 모바일 클라우드 컴퓨팅이 중앙화된 클라우드 서버를 이용하면서 발생했던 high latency 문제를 사용자와 지리적으로 가까운 엣지 서버를 이용함으로써 해결한다. 엣지 서버는 클라우드 서버와는 달리 여러 장소에 분산되어 있기 때문에, 모바일 기기는 자신으로부터 가장 가까이 있는 엣지 서버를 통해 low latency 로 컴퓨팅 파워를 제공받을 수 있다. 본 문서는 iOS 애플리케이션과 엣지 서버 간의 통신을 하여 모바일 엣지컴퓨팅 환경을 제안 및 구현한다.

1. 서론

1.1. 연구배경

현재 정보통신 분야의 동향을 살펴보면 스마트폰과 태블릿 컴퓨터 같은 모바일 기기들 대부분의 서비스는 클라우드 컴퓨팅 인프라를 활용하여 운영되고 있다. 이에 따라 사림들이 모바일 기기에서 기대하는 성능도 같이 증가하고 있다. 하지만 이러한 성능 향상에도 불구하고 5G 시대에는 기기가 더 긴밀하게, 방대한 데이터를 주고받으면서, 중앙의 데이터센터에서 모든 데이터를 처리하기에 역부족인 상황이 우려된다. 특히 최근 주목받기 시작한 인공지능(Artificial Intelligence), 증강 현실(Augmented Reality), 클라우드 게이밍(Cloud Gaming) 같은 극단적으로 낮은 전송 지연과 많은 데이터를 요구하는 응용 프로그램을 모바일 기기에서 수행하기는 쉽지 않다.

이러한 문제점을 해결하기 위해 모바일 엣지 컴퓨팅(Mobile Edge Computing)이라는 개념이 등장했다. 모바일 엣지 컴퓨팅은 모바일 클라우드 컴퓨팅이 중앙화된 클라우드 서버를 이용하면서 발생했던 high latency 문제를 사용자와 지리적으로 가까운 엣지 서버를 이용함으로써 해결한다. 엣지 서버는 클라우드 서버와는 달리 여러 장소에 분산되어 있으므로 모바일 기기는 자신으로부터 가장 가까이 있는 엣지 서버를 통해 low latency 로 컴퓨팅 파워를 제공받을 수 있다. 즉, 클라우드 컴퓨팅의 로드를 개별 로컬 서버로 전환함으로써 모바일 네트워크의 혼잡을 줄이고 대기 시간을 줄임으로써 최종 사용자의 QoE (Quality of Experience)를 향상시킨다.

특히, 데이터 수집 및 실시간 계산에 중점을 둔 인공 지능/기계학습 (AI/ML) 태스크를 데이터 소스부터 더 가까운 곳에서 처리하고, 연산 결과만 중앙집중식 데이터센터로 전송하는 기술에 초점을 두어 본 연구를 진행한다.

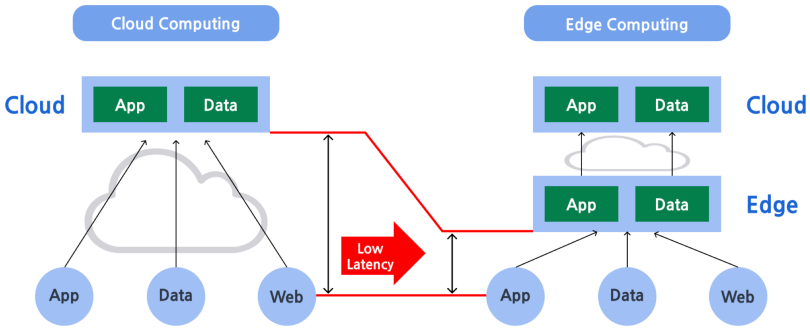

그림 1. 클라우드 컴퓨팅, 엣지 컴퓨팅

1.2. 연구목표

최근 몇 년 동안 거리에서 공유 자전거, 전동 킥보드로 대표되는 '퍼스널 모빌리티' 를 쉽게 마주할 수 있다. 일반적으로 정해진 승강장에서 여러 사람들이 함께 이용하는 여타 교통수단과 달리, '퍼스널모빌리티'는 원하는 목적지까지 부담 없이 이동할 수 있는 개인 맞춤형 모빌리티라는 점에서 도심 보행자들에게 인기를 끌고 있다. 이로 인해, 운전자들이 보도 위, 특히 시각장애인들을 위해 설치된 점자블록 위에 무단 주·정차해 시각장애인들의 통행을 방해하고 사고를 유발하는 경우가 빈번히 발생한다.

본 프로젝트는 공유 전동 킥보드를 반납하려는 사용자가 iOS 애플리케이션으로 촬영된 반납하려는 장소 이미지 기반으로 전동킥보드와 점자블록를 인식하여 점자블록 위에 무단 주·정차를 차단하는 기술을 구현하고자 한다.

이를 달성하기 위한 첫번째 목표로 전동 킥보드와 점자블록 이미지를 라벨링을 통해 데이터 셋을 제작한다. 두 번째 목표로 위에서 획득한 데이터 셋을 딥 러닝 모델에 학습시켜 빠른 시간 안에 전동 킥보드, 점자블록을 인식할 수 있도록 한다. 마지막으로 최종 사용자의 QoE 를 위해 처리 할 데이터를 원 서버로 오프로딩 하기 전에 연산을 위한 데이터를 미리 엣지 서버에 저장하여 학습 된 딥러닝 모델을 사용하여, 연산 결과만 중앙집중식 데이터센터로 전송할 수 있도록 구현한다.

2. 관련 연구

2.1. 객체 인식

이미지 또는 비디오 상의 객체를 식별하는 컴퓨터 비전 기술

2.1.1. 딥러닝 기반 객체 인식

객체 인식(Object Detection)은 컴퓨터 비전과 이미지 처리와 관련된 컴퓨터 기술로서, 디지털 이미지와 비디오로 특정한 계열의 시맨틱 객체 인스턴스(e.g. 사람, 자동 차, 건물)를 감지하는 일을 다룬다. 객체 인식은 크게 세 가지로 나눠진다. 영상 한 부분에 물체가 있다는 것을 인 식하는 Object Recognition, 그 물체가 무엇인지를 판단하는 Object Classification, 인식된 물체의 정확한 위치를 판단하는 Object Localization 이다. 순서도는 [그림 2]과 같다.

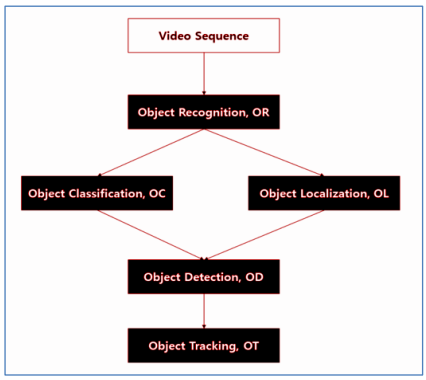

그림 2. 딥러닝 기반 객체 인식 순서도

2.1.2. 딥러닝 기반 객체 인식 모델

딥러닝 기반 객체 인식은 객체 검출 절차에 따라 두 가지로 구분한다. 먼저 등장한 방법은 Two-shot-detection 이다. Object Localization 과 Object Classification 이 각각 진행되며, CNN 에 Region Proposal 을 추가하여 물체가 있을 법한 곳을 제안하고, 그 구역에서 객체 인식을 하는 방법이다. Selective Search Algorithm 을 활용하여 높은 정확성으로 객체 인식을 하는 R-CNN(Regional Convolution Neural Network), Fast R-CNN 모델이 있으며, RPN(Region Proposal Network)을 활용하여 인식 속도를 개선한 Faster R-CNN, Mask R-CNN 모델이 있다. 하지만, Two-Shot-Detection 의 고질적 한계로 Real-time 객체 인식에는 속도가 아쉽다는 평이 있다. 이를 해결하고자, Object Localization 과 Object - Classification 이 동시에 진행되는 One-Shot-Detection 방식이 등장하였다. 영상을 한번 보는 것만으로 하나의 신경망을 통과하여 물체의 Bounding Box 와 Class 를 동시에 예측한다. 영상을 격자로 나누고 경계 상자를 예측하고 각 상자에 대한 신뢰도 점수를 예측한다. 영상 전체를 한 번에 처리하기 때문에 객체에 대한 맥락적 이해도가 높아서 배경에 대한 오류가 발생할 확률이 낮아지고 처리 과정이 단순하므로 매우 빠른 속도로 객체를 인식할 수 있다. 대표적 모델은 YOLO(You Only Look Once), SSD, RetinaNet 등이 있으며 그 중, YOLO 는 눈부신 발전을 통해 v5 까지 등장하였다.

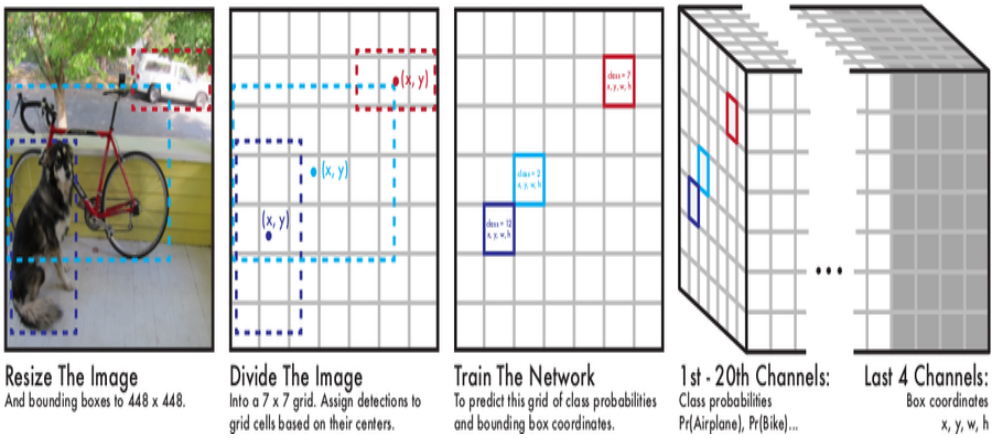

그림 3. YOLO 객체 인식 과정

YOLO v5 는 성능 및 fps 에 따라 YOLO v5s, YOLO v5m, YOLO v5l, YOLO v5x 로 나눠지며, 본 프로젝트는 제한된 환경인 엣지 서버에서 구현되어야 하기에 성능은 상대적으로 낮지만, fps 가 높은 YOLO v5s 모델을 사용하였다.

3. 프로젝트 내용

3.1. 시나리오

본 프로젝트의 전동 킥보드 반납 메커니즘은 다음과 같다:

1. 사용자가 전동 킥보드를 반납 시 주차 된 전동 킥보드 사진 촬영
2. 촬영된 이미지를 사용자와 지리적으로 가까운 엣지 서버로 전송
3. 학습 된 딥 러닝 모델을 통해 이미지에서 전동 킥보드와 점자블록을 인식
4. 인식 결과에 따라 다음과 같이 달라진다:
    A. 사용자에게 재촬영 요구
        i. 전동 킥보드가 이미지에 없는 경우
        ii. 전동 킥보드가 시각 장애인용 점자 블록 위에 주차된 경우
    B. 정상적 반납 완료
        i. 전동 킥보드가 이미지에 있으며, 점자블록 위에 주차된 경우가 아닌 경우
5. [4]에서 정상적 반납이 완료되었으면 최종 결과를 중앙집중식 데이터센터로 전송한다.

3.2. 요구사항

3.2.1. 딥러닝 모델에 대한 요구사항

객체 인식에 대한 최선의 결과를 도출하는 알고리즘 중 하나인 YOLO v5 는 좋은 선택지이다. 그러나 라즈베리파이와 같은 임베디드 장치에서 엣지서버로서 동작하기 위해서는 최대한 가벼운 모델이 필요하다. 이를 해결하고자 YOLO v5 모델들 중 가장 정확성은 낮지만, 인식 속도가 빠른 YOLO v5s 알고리즘을 선택하였다.

3.2.2. 엣지 서버에 대한 요구사항

모바일 기기에서 발생되는 데이터를 처리하기 위해 엣지 서버를 구축하여야 한다. 손쉽게 구할 수 있는 Raspberry Pi 를 통해 엣지 서버 구축을 목표로 한다.

3.2.3. 모바일 애플리케이션에 대한 요구사항

최종 사용자로부터 전동 킥보드에 이미지를 얻기 위해선 모바일 애플리케이션을 개발하여야 한다. Apple 사의 iPhone 위에서 작동하는 iOS 애플리케이션 개발을 목표로 한다.

3.3. 시스템 설계

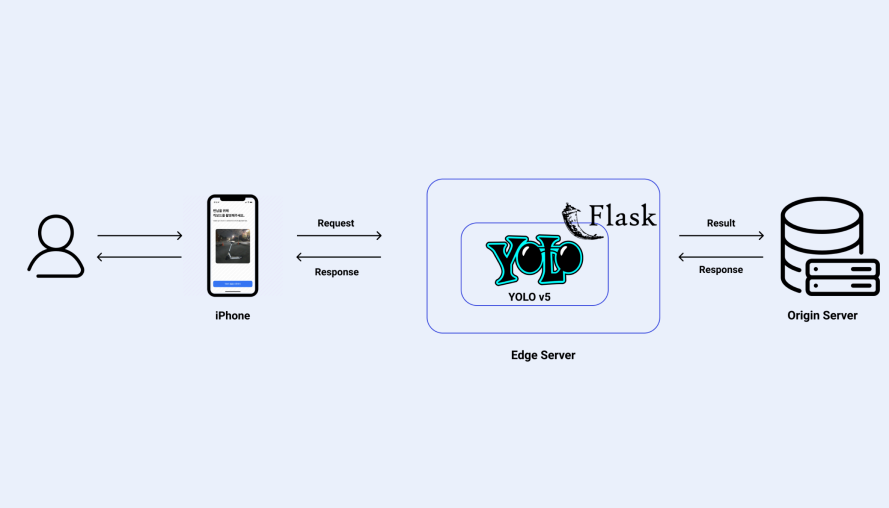

그림 4. 프로젝트 시스템 구조

본 프로젝트의 구조는 크게 클라이언트 사이드(Client-side), 엣지 서버 사이드(Edge-Server-side), 원장 서버 사이드 (Ledger-Server-side)로 나눌 수 있다. 클라이언트 사이드는 UIKit 프레임워크를 사용하여 아이폰 애플리케이션을 사용자에게 제공한다. 엣지 서버 사이드는 라즈베리파이 위에 구축된 Flask 프레임워크를 사용하여 YOLO v5 모델을 구동시킨다. Flask 프레임워크를 사용한 이유는 YOLO v5 모델 이랑 같은 Python 언어를 사용하기 때문이다. 마지막으로 원장 서버 사이드는 AWS DynamoDB 를 사용하여 엣지서버 사이드에서 처리한 데이터를 저장한다.

3.4. 구현

본 프로젝트의 Github 주소 및 데모 영상은 다음과 같습니다.

GitHub : https://github.com/Chuncheonian/mqtt-raspberrypi-ios
데모 영상 : https://drive.google.com/file/d/1wFqHKkJaNVUh7hnPfNkjdOBQlZ31h-OM/view?usp=sharing

4. 프로젝트 결과

4.1. iOS 애플리케이션 구현

iOS 어플리케이션은 UIKit 를 활용하여 제작하였으며, 총 4 가지의 화면을 가지고 있습니다. 사용자로부터 촬영된 전동 킥보드 이미지를 엣지 서버에 Request 하며 그에 대한 결과값을 Response 하여 재촬영을 요구할 지, 반납을 완료할 지를 사용자한테 알려줍니다.

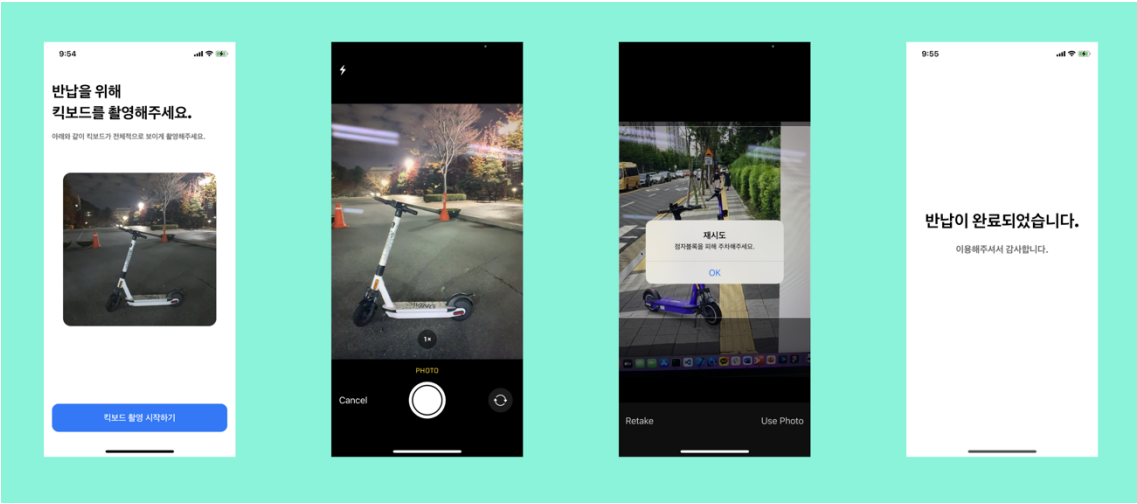

그림 5. iOS 애플리케이션 화면 구성

4.2. 전동 킥보드, 점자블록 객체 인식

본 프로젝트에서 제안하는 학습을 위해서는 이미지에 존재하는 물체의 BBOX (x, y, w, h)가 필요하다. 이를 위해 전동 킥보드, 점자블록 이미지를 RoboFlow 라벨링 툴을 사용하여 데이터셋을 수집하였다. 두 개의 클래스에 대한 데이터를 본 프로젝트에서 제안하는 모델의 학습 및 검증 데이터로

활용하였으며, 총 586 장의 데이터에서 409 장은 학습데이터로 128 장은 검증데이터로 49 장은 테스트 데이터로 사용하였다. 또한, 이미지 크기는 128*128, batch 는 3, epoch 는 60 으로 파라미터를 설정하였다.

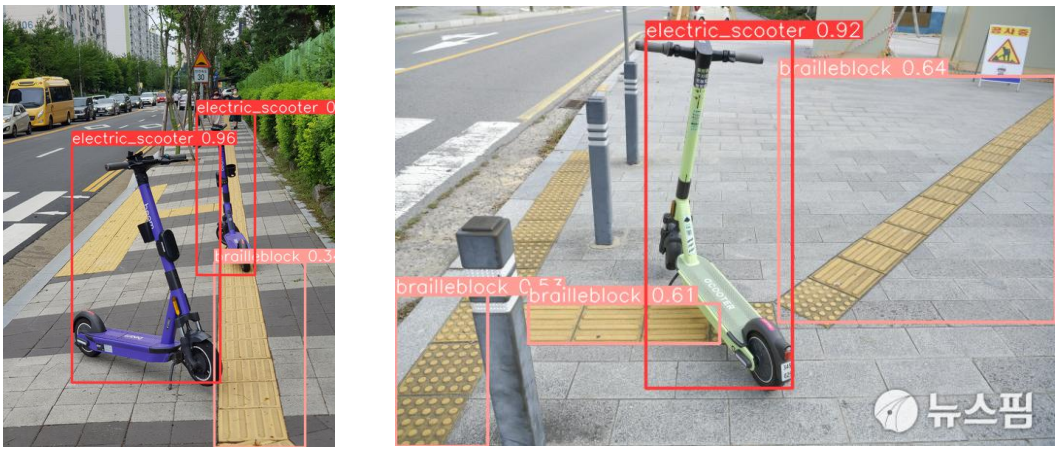

그림 6. 딥러닝 모델을 활용해 전동 킥보드, 점자블록 객체 인식 결과

4.3. 엣지 서버 구현

엣지 서버는 Flask 프레임워크를 활용하여 API 서버 구축을 하였으며 그 아래 YOLO v5 객체 인식기를 통해 iOS 애플리케이션로부터 전송 받은 이미지를 분석하고 그에 대한 결과값을 iOS 애플리케이션과 원장 서버로 전송한다.

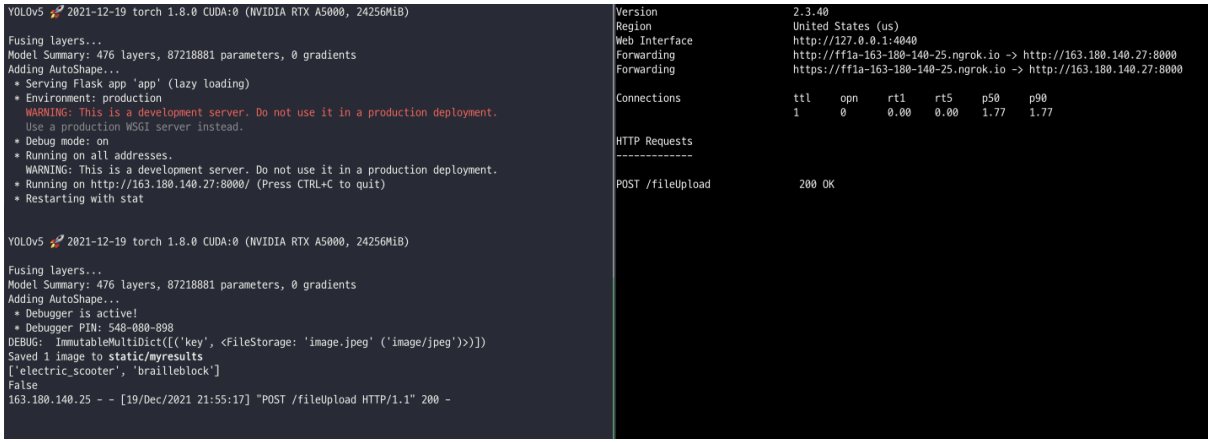

그림 7. 엣지 서버 구동 화면

4.4. 원장 서버 구현

원장 서버는 엣지 서버에서 처리한 연산 결과값만은 저장한 서버로서, AWS DynamoDB 로 구현하였다.

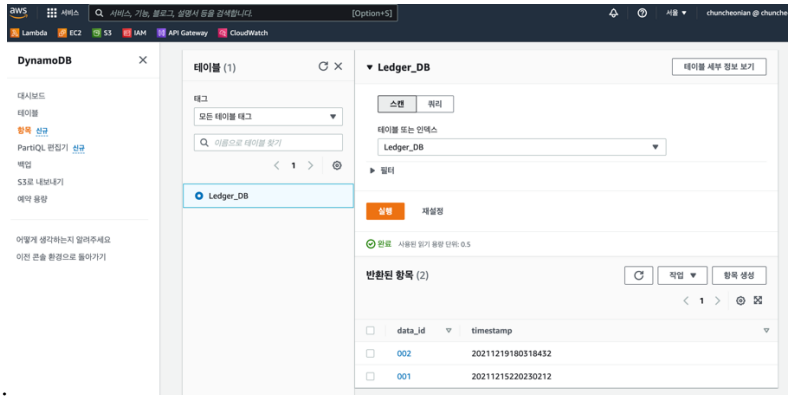

그림 8. 원장 서버

5. 결론 및 기대효과

다가오는 5G 시대에서 기기가 더 긴밀하게, 방대한 데이터를 주고받으면서, 중앙의 데이터센터에서 모든 데이터를 처리해야 한다는 문제점을 해결하기 위해 모바일 엣지 컴퓨팅이 등장하였다. 본 프로젝트에서는 모바일 엣지 컴퓨팅의 장점 중 하나인 데이터 수집 및 실시간 계산에 중점을 둔 인공지능/기계학습 (AI/ML) 태스크를 데이터 소스부터 더 가까운 곳에서 처리하고, 연산 결과만 중앙집중식 데이터센터로 전송하는 기술에 초점을 두어 진행하였다.

앞으로, 모바일 환경에서 인공지능(Artificial Intelligence), 증강 현실(Augmented Reality), 클라우드 게이밍(Cloud Gaming) 같은 극단적으로 낮은 전송 지연과 많은 데이터를 요구하는 응용 프로그램을 수행하는 것은 당연한 시대이다. 이번 프로젝트를 통해 모바일 엣지 컴퓨팅 환경에서 최종 사용자들이 얼마나 속도향상을 겪은 지를 확인하였으며 차후, 모바일 엣지 컴퓨팅 효과를 기대한다.

6. 참고 문헌

[1] Ultralytics, YOLO v5(2020), Retrieved June, 10, 2020, from https://github.com/ultralytics/yolov5