엣지 디바이스 기반 실시간 전기차 충전소 자리 인식 시스템

김서연, 김장환, 신은호, 이유진, 진경은, 전윤호*

국립한밭대학교

{seoyeon, kjh, eunho, yujin, jke}@edu.hanbat.ac.kr, *yhjeon@hanbat.ac.kr

Real-Time Electric Vehicle Charging Station Parking Slot Recognition System Based on Edge Devices

Seoyeon Kim, Janghwan Kim, Eunho Shin, Yujin Lee, Kyungeun Jin, *Yunho Jeon

Hanbat National University

요 약

기존의 전기차 충전소의 위치 정보 및 실시간 사용 현황을 보여주는 플랫폼이 존재하지만, 플랫폼 간의 정보 연동 문제로 인해 불편을 겪는 사용자들이 많다. 이러한 문제를 해결하고자 본 논문에서는 객체 인식 모델을 엣지 디바이스에 최적화하여 구현하였으며, 이를 통해 제한된 자원에서도 차량 충전소의 공간 점유 여부를 실시간으로 파악할 수 있는 알고리즘을 제안한다. 이 알고리즘을 통하여 차량의 진입 및 출차 여부에 따라 즉각적으로 상태가 업데이트되는 시스템을 구현하였다.

Ⅰ. 서론

최근 전기차의 보급이 급격히 증가함에 따라 전기차 충전 인프라의 중요성이 커지고 있다. 전기차 충전소의 위치 정보 및 사용 현황을 제공하는 다양한 플랫폼이 존재하며, 이를 통합한 정보 제공 플랫폼도 운영되고 있다. 그러나 플랫폼 간 정보 연동이 원활하지 않아 사용자들이 충전소의 정확한 이용 가능 여부를 실시간으로 확인하기 어렵다는 문제가 있다.

이러한 문제를 해결하기 위해, 본 논문에서는 엣지 디바이스 기반의 실시간 전기차 충전소 자리 인식 시스템을 제안한다. 해당 시스템은 라즈베리파이와 같은 엣지 디바이스가 직접 객체 인식을 수행하여 전기차 충전소의 주차 공간 및 주차 시간을 실시간으로 모니터링한다. 이를 통해 적은 자원으로 정보 연동을 진행하여 사용자에게 정확하고 즉각적인 정보를 제공하는 것을 목표로 한다.

Ⅱ. 본론

2.1 객체 탐지 모델

YOLO(You Only Look Once)는 실시간 객체 탐지 알고리즘으로, 정확성과 속도 면에서 높은 성능을 제공한다. 본 논문에서 사용한 YOLOv8 모델은 YOLO의 최신 모델이며, 정확도와 속도 간의 균형을 유지하는 데 중점을 두어, 다양한 응용 분야에서 실시간 객체 탐지 작업에 적합하다. 이 중에서도 논문에서 사용한 YOLOv8n 모델[1]은 YOLOv8 모델 중 가장 빠른 속도를 가지고 있어 엣지 디바이스 기반 작동을 위해 해당 버전을 선택하였다.

2.2 모델 활성화 함수

YOLOv8의 기본 활성화 함수는 SiLU이다. SiLU는 sigmoid 함수에 입력값을 곱하여 출력값을 도출하는 함수로, 그림 1의 그래프에서 볼 수 있듯이 음수 부분에서의 출력값을 0으로 만드는 ReLU와 달리 음수 부분에서 0이 아닌 출력값을 도출한다. 이로 인해 정보 손실을 줄여, 활성화 함수로 ReLU를 사용하였을 때보다 SiLU를 사용했을 때 더 좋은 성능을 낼 수 있다.

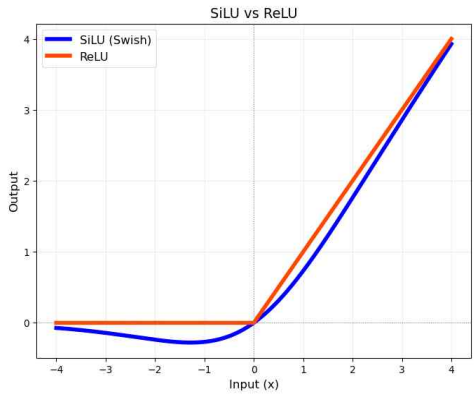

그림 1. 활성화 함수 비교

그러나 정보 손실을 줄인 만큼 연산량이 많아져 ReLU보다 정확성은 높지만 추론 시간이 오래 걸린다. 본 연구의 목적은 엣지 디바이스 기반의 알고리즘을 개발하는 것이기 때문에 제한된 자원 환경에서 모델의 추론 효율성을 향상시키기 위해 활성화 함수를 기존의 SiLU에서 연산량이 더 적은 ReLU로 변경하였다. 표 1에서 활성화 함수에 따른 추론 시간의 차이를 살펴볼 수 있다.

활성화 함수	추론 시간
SiLU	50.65ms
ReLU	48.58ms

표 1. 활성화 함수에 따른 추론 시간 비교

2.3 모델 Fine-Tuning

2.2절에서 서술한 활성화 함수 변경으로 인한 정확도 감소를 보완하기 위해 COCO 데이터셋으로 사전 학습된 모델을 Visdrone 데이터셋[2]을 활용하여 Fine-Tuning을 수행하였다.

Visdrone은 드론에 장착된 카메라로 촬영한 데이터셋으로, 보행자, 차량, 자전거 등 다양한 객체들이 포함되어 있다. 해당 데이터셋의 촬영 구도와 본 논문에서 사용할 알고리즘 개발에 사용된 데이터셋의 구도가 유사해 Visdrone2019-DET 데이터셋을 사용하여 Fine-Tuning을 진행했다. 그림 2에서 Fine-Tuning 이후 검출 성능이 향상된 것을 확인할 수 있다.

그림 2. 좌 - Fine-Tuning 전, 우 - Fine-Tuning 후

2.4 알고리즘 구현

본 논문에서는 YOLOv8n과 라즈베리파이 4를 사용하여 차량 주차 여부 및 시간 추적 알고리즘을 개발하였다. 주차 공간의 차량 점유 여부를 파악하기 위해 각 주차 공간을 다각형으로 정의하고, 인식할 객체(차량)의 중심이 주차 공간 내부에 포함되는지를 확인하였다.

Algorithm 1 주차 공간 및 시간 추적 알고리즘

입력:
current: 현재 상태 리스트 (1: 차량 존재, 0: 비어 있음)
previous: 이전 상태 리스트
entry: 차량 진입 시간 리스트
duration: 차량 머문 시간 리스트
miss_count: 차량 부재 카운트 리스트
vehicle_times: 전송할 머문 시간 리스트 (초 단위)

for $i \in \{1, 2, \ldots, n\}$ **do** ▷ n: 주차 공간 수
  if $previous[i] = 0$ **and** $current[i] = 1$ **then** ▷ 새로운 차량 진입
    $entry[i] \leftarrow current\_time()$
    $duration[i] \leftarrow 0$
    $miss\_count[i] \leftarrow 0$
  else if $previous[i] = 1$ **and** $current[i] = 0$ **then** ▷ 차량이 떠남
    $miss\_count[i] \leftarrow miss\_count[i] + 1$
    if $miss\_count[i] > limit\_counter$ **then** ▷ 차량 떠남 확정
      $entry[i] \leftarrow 0, duration[i] \leftarrow$ "", $miss\_count[i] \leftarrow 0$
    end if
  else if $current[i] = 1$ **and** $previous[i] = 1$ **then** ▷ 차량이 계속 머무름
    $duration[i] \leftarrow current\_time() - entry[i]$
    $miss\_count[i] \leftarrow 0$
  end if
  $vehicle\_times[i] \leftarrow duration[i]$ ▷ 머문 시간 저장
end for
$previous \leftarrow current$ ▷ 현재 상태를 이전 상태로 갱신
서버 전송: send_parking_data(vehicle_times)

그림 3. 주차 공간 및 시간 추적 알고리즘

차량의 진입 및 출차 여부에 대한 알고리즘은 그림 3과 같다. 알고리즘은 카메라를 이용하여 주차 공간을 지속적으로 모니터링하는 방식으로 동작한다. 카메라가 특정 주차 공간에서 차량을 인식했을 때, 해당 공간이 직전까지 비어 있었던 경우 새로운 차량이 주차 공간에 진입한 것으로 판단하고, 진입 당시의 시간을 저장한다. 반대로 차량이 인식되고 있었던 주차 공간에서 차량이 더 이상 감지되지 않을 경우 차량이 떠난 것으로 판단한다. 마지막으로, 특정 주차 공간에서 차량이 연속적으로 인식되는 경우 차량이 계속 주차하여 충전 중임을 의미한다. 이때 차량이 진입한 시간과 현재 시간의 차이를 계산하여 차량이 머문 시간을 저장하고, 저장된 정보는 서버로 전송한다.

2.5 웹 프론트엔드 구현

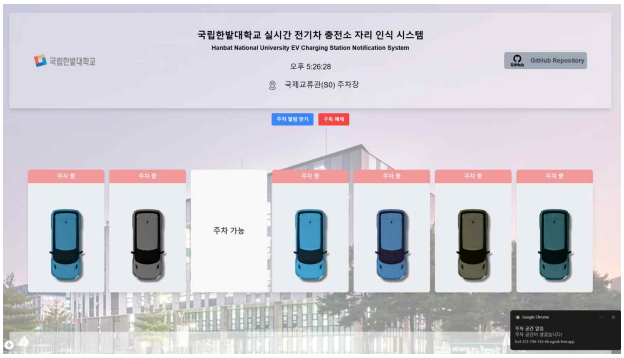

그림 4. 웹 페이지 작동 화면

Next.js[3]는 React를 기반으로 웹 애플리케이션을 개발하는 데 필요한 전체적인 틀을 제공하는 웹 개발 프레임워크로, 이를 사용하여 API 서버를 구현하였다. 해당 API 서버는 라즈베리파이에서 실시간으로 전송된 주차 공간 리스트를 수신하고, 이를 웹 페이지에 제공한다. 또한 모든 주차 자리가 채워져 있을 때 화면 중앙의 주차 알림 받기 버튼을 눌러 빈 주차 공간이 생겼을 때 사용자에게 알림을 발송하는 알림 시스템을 구현했다. 이 과정에서 사용자가 웹 페이지에 접근할 때마다 서버에서 최신 정보를 렌더링하는 서버 사이드 렌더링(SSR) 방식을 사용하여 사용자가 빠른 속도로 웹 페이지에 접근할 수 있도록 돕는다.

추가로 해당 애플리케이션의 배포 및 유지보수 작업을 위해 Vercel을 활용하였다. Vercel[4]은 Next.js 애플리케이션의 구축, 확장 및 보안을 위해 설계된 공식 플랫폼으로, 이를 사용하여 Next.js에 최적화된 방식으로 웹 페이지를 인터넷에 배포하였다. 또한 CI/CD 개발 방식을 사용하여 Github와 Next.js를 연동해 코드 수정 시 플랫폼에서 자동으로 테스트를 진행하고, 테스트를 통과하였을 때 수정된 웹 페이지가 자동으로 사용자에게 배포되도록 설정하여 개발 편의성을 높였다.

Ⅲ. 결론

본 논문에서는 기존 전기차 충전소 플랫폼의 정보 연동 문제로 인한 사용자의 불편을 해결하기 위해 엣지 디바이스 기반의 실시간 전기차 충전소 자리 인식 시스템을 구현하였다. YOLOv8n을 라즈베리파이 4에서 작동하도록 구현하여 엣지 디바이스에서 직접 객체 인식을 진행함으로써 낮은 사양에서 높은 정확도를 보였고, 다양한 오픈소스를 사용하여 최적화된 웹 어플리케이션을 구현했다. 이를 통해 사용자에게 신속하고 정확한 정보를 전달하며, 알림 시스템을 구현하여 보다 편리하게 전기차 충전소를 이용할 수 있도록 지원한다.

참 고 문 헌

[1] Ultralytics YOLOv8. (n.d.). Ultralytics. https://docs.ultralytics.com/ko/models/yolov8/

[2] D. Du et al., "VisDrone-DET2019: The Vision Meets Drone Object Detection in Image Challenge Results," 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pp. 213-226.

[3] Next.Js Routing Fundamentals. (n.d.). Next.Js. https://nextjs.org/docs/app/building-your-application/routing

[4] Vercel Documentation. (n.d.). Vercel. https://vercel.com/docs