https://doi.org/10.7840/kics.2023.48.7.842

# 운전자 행동 분석을 위한 차량 동작 상태 인식 온-디바이스 AI의 설계 및 구현

김 태 구•, 조 용 훈*, 백 윤 주°

# Design and Implementation of Vehicle Operating Status Recognition On-Device AI for Driver Behavior Analysis

Taegu Kim•, Yonghun Cho*, Yunju Baek°

요 약

최근 차량용 센서 및 인공지능의 기술 개발에 따라 전 세계적으로 자율 주행과 같은 운전자의 운전 편의에 대한 기술 개발이 활발하다. 하지만 시스템 안전성의 검증 때문에 개발 상황과 비교하면 상용화 비율이 낮다. 따라서 본 논문에서는 운전자가 주행하는 차량에서 사용 가능한 온-디바이스 AI를 구현하여 운전자 행동 분석 및 위험운전 인식에 사용할 수 있도록 차량 동작 상태를 분류하는 연구를 수행하였다. 차량의 동작 상태를 8가지로 정의하였고 추출한 차량 내부 정보를 사용하여 차량 동작 상태를 추론하는 딥러닝 모델을 설계하였다. 마지막으로 온-디바이스 AI에서 실시간으로 차량 동작 상태를 추론하기 위해 딥러닝 모델을 디바이스에 탑재한다. 디바이스에 딥러닝 모델을 탑재하기 위해 딥러닝 모델 구조를 변경하고 양자화를 통해 딥러닝을 경량화 하였다. 최종적으로 구현한 온-디바이스 AI를 실제 차량에서 주행하면서 실시간 차량 동작 상태 추론을 수행하여 성능을 평가한다. 차량 동작 상태 인식 정확도는 91.66% 성능을 나타내었고 추론 시간은 19.72ms로 구현되어 실시간 추론을 수행하였다.

**Key Words :** On-device AI, Vehicle operating status, Driver behavior analysis, Model lightweight

ABSTRACT

With the recent development of technologies for vehicle sensors and artificial intelligence, technologies for driver convenience such as autonomous driving are actively developed around the world. However, due to the verification of the safety of the system, the commercialization rate is lower than the development situation. Therefore, in this paper, a study was conducted to classify the vehicle operating status so that it can be used to analyze driver behavior and recognize dangerous driving by implementing on-device AI available in the vehicle driven by the driver. Deep learning model was designed to infer the vehicle's operating status using the extracted vehicle interior information. In order to mount a deep learning model on a device, the structure of the deep learning model was changed and lightened through quantization. The performance is evaluated by performing real-time vehicle operation status inference while driving the finally implemented on-device AI in the real vehicle. The vehicle operating status recognition accuracy showed 91.66% performance and the inference time was 19.72 ms

※ 본 연구는 한국산업기술평가관리원(KEIT) 연구비 지원에 의하여 수행되었습니다.(20015052)
• First Author : Pusan National University Department of Computer Science and Engineering, tbg8577@pusan.ac.kr, 학생회원
° Corresponding Author : Pusan National University Department of Computer Science and Engineering, yunju@pusan.ac.kr, 종신회원
* Pusan National University Department of Information Convergence Engineering, kchoyh95@pusan.ac.kr 학생회원
논문번호 : 202212-295-C-RU, Received December 8, 2022; Revised March 13, 2023; Accepted April 17, 2023

# Ⅰ. 서 론

## 1.1 연구 배경

최근 차량용 센서 및 인공지능의 기술 개발에 따라 전 세계적으로 자율 주행 기술 개발이 활발하다. 자율주행 기술은 운전자의 과실로 발생하는 교통사고를 줄여 운전자와 보행자의 안전을 보장하고 고령자, 장애인 및 임산부와 같은 운전이 불가한 사람의 교통편의 증진 등 다양한 기대가 높아지고 있다.

하지만 자율주행 자동차와 운전자의 주행 보조 시스템(ADAS: Advanced Driver Assistance System)의 활발한 개발 상황에 비해 상용화 전망이 낮고 운전자의 수용도도 낮게 평가되고 있어 시스템의 안정성을 검증하기 위해 시스템의 위험운전을 판단할 필요가 있다. 현재 위험운전을 판단하는 방법은 운행기록장치를 통해 운전자의 위험 운전 행동 분석에 사용한다. 교통사고에 직결되는 과속, 급가속, 장기 과속, 급출발, 급감속, 급정지, 급진로변경, 급앞지르기, 급회전, 급유턴, 급가속 등으로 11개의 위험운전을 정의하고 운전자의 위험운전을 판단한다. 하지만 단순히 차량의 상태만으로 위험운전으로 판단하여 활용하기는 한계가 있다. 교통안전을 위해서는 운전자의 차량 조작 행동과 차량의 현재 상태를 고려하여 위험 운전을 정확하게 추정하는 것은 중요하다.

## 1.2 관련 연구

차량에 센서를 부착하여 데이터를 수집하거나 차량 시뮬레이터를 사용하여 교통안전을 위해 운전자의 운전 스타일을 식별하는 연구가 있다.[1-3] 운전자의 스타일을 인식하여 분석하는 것으로 서로 다른 운전자를 구별하여 도움이 되는 변화를 포착한다. 이 작업은 사용자 개인화된 보험, 운전자 행동 예측, 자율 주행 설계에도 활용할 수 있다. 하지만 운전 스타일 인식 연구는 위험운전의 데이터를 의도적으로 정의하여 수집해서 사용하기 때문에 실제 사고에 관련된 위험운전을 인식하는 연구에 적용하는 것은 한계가 있을 수 있다. 또한 운전자 개개인의 운전 특징을 학습하여 운전자를 구분하는 연구가 있다.[4,5] 다양한 운전자를 통해 수집한 데이터를 사용하여 딥러닝을 설계하여 운전자 개개인을 인식하여 운전자의 행동 분석에 활용할 수 있다. 하지만 학습된 데이터가 모든 운전자를 대표하기 힘들기 때문에 학습 데이터에 과도하게 적용되어 새로운 데이터에서 성능이 감소할 수 있어 한계가 있다. 미국에서의 자동차 충돌보고서에 따르면 2015년 451,000건의 자동차충돌 사고가 차선변경 도중에 발생한다고 한다.[6] 따라서 운전자의 조작에 따라 발생하는 차량의 운전 동작을 인식하는 연구가 있다.[7,8] 차선 변경 중의 데이터를 수집하여 딥러닝으로 구현한다. 하지만 대부분의 차선 변경 인식 연구의 경우 제한된 환경에서 수집된 데이터를 활용하거나, 실제 차량에서 구현한 것이 아닌 고성능 컴퓨팅 장비를 활용하여 구현한 경우가 많아 실제 서비스에 사용하기에는 제한이 많다.

최근 딥러닝 모델을 활용한 서비스와 디바이스에 대한 개발이 활발하다.[4] 일반적으로 딥러닝은 높은 정확도를 위하여 모델을 크기가 클수록 성능이 좋아진다. 하지만 모델이 클 경우 많은 메모리 공간과 연산량을 필요하기 때문에 실제 딥러닝 모델을 이용하는 로봇, 자율자동차 및 소형 디바이스와 같은 환경에서는 구현에 어려움이 있다. 심층 신경망의 모델의 크기가 클 경우 많은 파라미터 값을 저장하고 있기 때문에 큰 메모리가 필요하다. 또한 딥러닝 모델은 저장된 가중치를 활용해 많은 연산이 필요하다. 하지만 프로세서의 성능이 낮은 경우 메모리가 부족하거나 추론에 많은 시간이 소요된다. 따라서 성능이 낮은 소형장치에서 구현하기 위해서 더 적은 파라미터를 가지고 연산량을 감소시키는 모델 경량화 방법이 필요하다. 따라서 저사양의 MCU (Micro Controller Unit)에서 딥러닝 모델을 구현하는 연구가 필요하다.

운전자 행동을 분석하기 위한 차량의 동작 상태를 인식하기 위해 다양한 센서 정보를 사용한 연구가 있다. 차량 외부 정보는 차량의 운전 정보와 차량 상태를 알아내기 위해 추가로 센서를 부착하여 사용한다. GPS (Global Positioning System)의 경우 인공위성에서 신호를 받아 정밀한 현재 위치정보를 수집할 수 있다.[11] 하지만 GPS는 1초당 1번의 데이터를 수집하기 때문에 순간적인 변화를 인식하기 어렵다. IMU(Inertial Measurement Unit) 센서의 경우 3가지의 축을 기준으로 가속도, 각속도 등의 물리량을 측정하여 데이터로 사용한다.[12] 하지만 차량의 변화에 동일한 변화량을 수집하기 위해 고정된 위치에 장착하거나 오류 값을 보정하는 기법이 필요하다.[13] 최근에는 자율주행 연구를 위해 라이다(Lidar) 센서를 사용하는 연구도 있다.[14,15] 라이다 센서는 발사한 레이저가 목표물에 맞고 되돌아오는 시간을 측정하여 주변의 장애물을 감지하거나 현재 위치를 파악하기 위한 정밀한 데이터를 측정할 수 있다. 하지만 추가적인 센서를 장착하는 차량 외부 정보는 추가적인 비용이 필요하거나 제한적인 공간인 차량에서 추가적인 센서를 장착하는 방법은 적절하지 않다. 차량 내부 정보는 추가로 센서를 장착하지 않고 데이터를 수집한다.

차량 내부 정보는 차량의 OBD 커넥터와 연결하여 차량과 통신 하여 다양한 차량 정보를 수집할 수 있다. 하지만 진단 목적을 위한 표준 OBD-II가 아닌 UDS프로토콜이나 비표준 CAN 정보의 경우 센서 정보에 해당하는 CAN ID를 찾을 필요가 있다. 따라서 차량 내부 정보 추출을 위한 연구가 있다. 운전자가 차량을 조작하여 차량 내부 정보를 비표준 CAN ID 추출 기법을 연구하였다.[9] 또한 UDS 프로토콜을 이용한 실시간 차량 내부 정보를 추출하는 기법을 연구하였다.[10]

## Ⅱ. 본 론

본 논문은 운전자가 주행하는 차량에서 사용 가능한 디바이스를 구현하여 운전자 행동 분석 및 위험운전 인식에 사용할 수 있도록 차량 동작 상태를 분류하는 연구를 수행하였다. 또한 관련 연구에서 발생하는 문제를 해결하기 위해 본 논문은 실제 차량에서 동작되는 현재 자동차 상태를 추론하는 온-디바이스 AI 구현을 제안한다. 이를 위해 차량을 통해 차량 내부 정보를 추출하는 하드웨어를 설계한다. 설계된 하드웨어는 실제 차량에서 테스트를 위해 통신 모듈과 소모 전력을 줄이기 위해 저전력으로 구현하였다. 추출 가능한 차량 내부 정보를 분석하고, 차량 동작 상태를 미리 설정한 차량 동작 상태를 추론하는 딥러닝 모델을 구현 및 학습하였다. 디바이스에서 차량 동작 상태 추론하기 위해서 딥러닝 모델을 경량화하여 탑재하였다. 이를 통해 연산능력에 한계가 있는 소형 디바이스에서 실시간으로 차량 동작 상태를 추론할 수 있게 구현하였으며, 실제 차량 주행을 통해 온-디바이스 AI에서 차량 동작 상태의 실시간 추론 정확도 성능을 평가하였다. 본 장에서는 본 논문에서 구현에 사용된 배경 지식과 설계 및 구현 내용에 대해 설명한다.

### 2.1 차량 내부 정보

본 논문에서는 딥러닝 학습을 위한 차량 내부 정보를 사용한다. 차량 내부 정보는 차량의 OBD-II 커넥터를 사용하여 차량과 통신하여 차량 내부의 센서 정보를 추출하여 사용한다. OBD-II는 커넥터가 SAE J1962로 표준화되어 있으며, 한국에서도 의무화됨에 따라 모든 차량이 OBD-II 포트를 가지고 있다. 보통 운전자가 조작하는 정보인 운전자 조작정보와 현재 차량의 상태 정보를 차량 상태정보가 있다. 운전자 조작정보는 스티어링 휠 각도, 액셀 페달, 브레이크 페달, 방향지시등 정보 등이 있고, 차량 내부 정보는 현재 차량 속도, RPM(Revolution Per Minute), 오도미터 등의 정보가 포함된다.

통신 방법으론 OBD-II 표준, UDS 프로토콜, K-line, 비표준 CAN정보 등이 있다. 표준화된 OBD-II 커넥터에서 7번 핀은 K-line으로 할당되어 있으며, 6번과 14번 핀은 CAN 통신을 위해 할당되어 있다. 차량 내부에 존재하는 전자 제어 장치(Electronic Control Unit)의 통신을 목적으로 국제 표준 규격으로 CAN(Controller Area Network) 프로토콜이 있다. OBD-II(On-Board Diagnostics)는 차량 배기가스 등을 진단하기 위한 시스템이며 차량의 제어부품이나 시스템을 모니터링 한다. OBD-II 표준, UDS 프로토콜, 비표준 CAN정보는 차량과 CAN(Controller Area Network) 통신을 사용하여 데이터를 수집할 수 있다. SAE J1979 OBD-II 표준은 CAN 버스를 이용하여 Parameter ID(OBD-II PID)를 기반으로 차량의 정보를 요청-응답 형식의 통신을 통하여 데이터를 추출한다. 모든 차량과 동일한 통신 방법을 가지지만 초기 목적인 배기량과 배기가스와 같은 환경개선이 목표였기 때문에 차량의 조작 정보가 제외된 제한된 정보만 얻을 수 있다. UDS 통신 프로토콜은 OBD-II와 같이 요청-응답 형식의 CAN 통신을 사용한다. ISO 14230와 ISO 15765과 같은 표준들을 기반으로 ECU와 통신할 수 있도록 만들어진 진단용 통신 프로토콜이다. UDS통신은 기술자들이 차량을 진단하기 위해 만들어졌기 때문에 OBD-II보다 다양한 정보를 추출할 수 있다. K-line의 Keyword Protocol 2000(KWP2000)은 ISO 14230으로 표준화되어 있고 물리 계층은 ISO 9141이다. K-line도 OBD-II 포트를 사용하여 데이터 요청-응답으로 데이터를 수집할 수 있다. 비표준 CAN 정보는 OBD포트의 CAN 버스에서 무분별하게 전송되는 원시 CAN 정보이다. 센서 값에 해당하는 PID를 파악하여 운전자의 조작정보와 차량 상태 정보를 추출할 수 있다. 본 논문에서는 UDS 통신을 사용하여 차량 데이터를 수집한다.

### 2.2 딥러닝 모델 경량화

경량화 하는 방법은 크게 두 종류로 나눌 수 있는데, 학습된 모델의 크기를 줄이는 방법과 네트워크 구조 자체를 효율적으로 설계하는 방법이 있습니다. 본 논문은 DS-CNN(Depthwise Separable Convolution Neural Network) 모델 구조를 사용하였다. 그림 1은 CNN과 DS-CNN의 모델 구조이다. 일반적인 CNN의 경우 커널을 사용하여 입력 데이터의 위치별 컨볼루션 연산을 통해 피처 맵(Feature map)을 생성한다. 입력 데이터의 가로의 크기를 $W$, 세로의 크기를 $H$, 입력 채널수를 $M$, 커널의 크기를 $K$, 출력 채널수를 $N$이라고 정의한

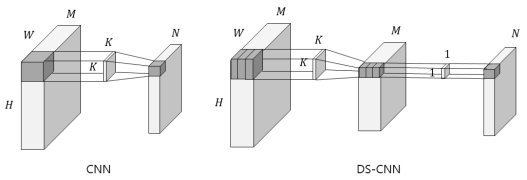



그림 1. CNN 모델과 DS-CNN 모델의 구조
Fig. 1. Structure of CNN Model and DS-CNN Model

다. 이때 CNN의 컨볼루션을 통해 피처 맵을 생성할 때 커널이 $N$개만큼의 컨볼루션 연산을 수행하기 때문에 연산량은 $MWHNK^2$, 파라미터 수는 $NMK^2$이다.

DS-CNN는 Depthwise convolution과 Pointwise convolution으로 구성되어 있다. Depthwise convolution은 입력 데이터의 채널마다 공간(Spatial) 피처를 추출하기 위해 컨벌루션을 수행한다. 입력 채널의 수와 출력 채널의 수가 동일하기 때문에 입력 데이터의 채널의 수와 커널과의 연산만 고려하면 된다. 따라서 Depthwise convolution의 연산량은 $MWHK^2$가 되며, 파라미터의 수는 $MK^2$이 된다. Pointwise convolution는 입력 데이터에 커널의 크기가 1로 고정되어 1x1 convolution을 수행하여 입력 데이터의 채널간의 연산만 수행하게 된다. 따라서 출력 피처 맵은 크기는 변하지 않고 채널의 수를 감소시킬 수 있다. Pointwise convolution의 연산량은 $MWHN$이며 파라미터 수는 $1^2M\times N = MN$이 된다. 따라서 DS-CNN의 연산량과 파라미터 수는 다음과 같다.

$$MWHK^2 + MWHN = MWH(K^2 + N) \quad (1)$$

$$MK^2 + MN \quad (2)$$

연산량을 비교하면 CNN에 비해 $1/N+1/K^2$ 만큼의 이득을 얻을 수 있다.

### 2.3 현재 차량 동작 인식 온-디바이스 AI

본 논문에서 실제 차량에서 차량의 현재 상태 정보와 차량 조작 상태 데이터를 수집하는 디바이스를 설계 및 구현하여 현재 차량 동작 상태를 추론하는 딥러닝을 경량화하여 디바이스에서 추론하는 온-디바이스 AI를 설계 및 구현하였다. 제안하는 온-디바이스 AI의 구성 요소 및 상호작용은 그림 2와 같다. 차량 내부 데이터를 수집하기 위한 디바이스는 차량에 존재하는 OBD-II 커넥터에 연결되어 데이터를 수집한다. 이 디바이스는 CAN 통신을 지원하는 MCU모델과 CAN 트랜시버를 사용하여 CAN통신을 통해 데이터를 전송하거나 수신

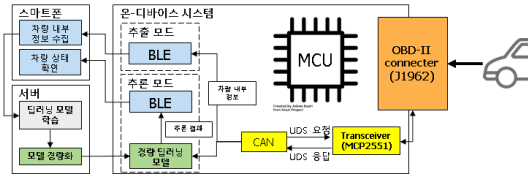



그림 2. 구현된 온-디바이스 AI의 시스템 동작 흐름도
Fig. 2. System Flow Diagram of Implemented Devices

할 수 있다. 수집된 데이터를 사용하여 차량 동작 상태를 추론하는 딥러닝 모델을 서버에서 학습한다. 학습한 딥러닝 모델을 디바이스의 MCU에서 동작시키기 위해 경량화를 수행하여 탑재한다. 구현한 온-디바이스 AI는 실제 차량에 장착하여 실시간으로 차량 동작 상태를 추론하고 추론 결과는 BLE 통신을 사용해 스마트폰에서 결과를 확인하고 기록된다.

### 2.4 차량 내부 정보 데이터 수집

본 논문에서 차량 동작 상태를 인식하기 위한 딥러닝 모델 학습을 위해 총 12가지의 차량 내부 정보 데이터를 수집한다. 수집하는 차량 내부 정보는 차량의 상태를 나타내는 차량 상태 정보와 운전자가 조작하는 정보인 운전자 조작 정보로 분류된다. 수집하는 차량 상태 정보는 차량 속도(speed), 종방향 가속도(longitudinal acc.), 횡방향 가속도(lateral acc.), Yaw 비율(yaw rate), 스로틀 포지션 센서(throttle position), 차량의 각 바퀴의 속도(front/left, front/right, rear/left, rear/right wheel speed)이다. 차량 속도는 차량이 달리고 있는 속도에 대한 정보이다. 종방향, 횡방향 가속도는 현재 주행 중인 차량을 기준으로 수직 방향과 수평 방향에서의 가속도 정보이다. yaw 비율은 차량의 중심에서 수직 방향을 기준으로 변하는 각속도를 의미한다. 스로틀 포지션 센서는 엔진에 전달되는 연료량을 조절하기 위해 개방 각도이며, 엔진의 가속 상태를 조절하는 역할을 하는 정보이다. 각 바퀴의 속도는 주행 중에 나타나는 각 바퀴의 회전 속도이다. 차량 조작 정보는 운전자가 차량을 조작할 때 사용하는 스티어링 휠, 가속 패달, 브레이크 패달의 정보를 사용한다. 스티어링 휠의 각도(steering wheel angle)는 운전자가 스티어링 휠을 조작할 때의 각도 정보이다. 가속 패달과 브레이크 패달은 운전자가 운전하면서 가속 패달과 브레이크 패달을 밟았을 때의 압력값(bar)과 백분율(percent)값이다. 센서 값의 선정은 실제 차량에서 여러 차량 동작 상태를 주행하며 차량 내부 정보를 수집하였을 때 변화량이 큰 값들을 기준으로 선택하였다.

### 2.5 차량 동작 상태 정의

본 논문에서 구현한 현재 차량 상태를 추론하는 딥러닝이 분류하는 현재 차량 상태에 대해 정의한다. 운전자가 운전하는 차량의 동작 상태는 다양한 도로를 주행하면서 현재 차량의 동작 상태가 변한다. 따라서 차량이 주행 될 때 동작 상태를 8가지 동작 상태로 단순화하였다. 그림 3은 정의된 차량 동작 상태를 보여준다. 크게 회전 상태가 없는 직진 도로 주행 상태와 회전 도로 주행 상태로 분류된다. 직진 도로 주행 상태에는 차선을 지키는 직진 상태(straight), 차량이 정지되어 있는 정지 상태(stop), 그리고 왼쪽 및 오른쪽으로 차선을 변경하는 상태(Lame change left, Lame change right)이다. 차량 회전 상태에는 일정한 각도로 회전하는 곡선 도로 주행 상태(Curve left, Curve right)와 교차로나 U턴과 같이 단시간에 차량이 크게 회전하는 도로를 주행하는 회전 상태(Turn left, Turn right)로 정의한다. 본 논문에서는 곡선도로에서 차선 변경하는 상태는 고려하지 않는다.

실제 자동차 주행 과정에서 같은 도로를 주행하더라도 외부 환경에 따라서 도로 형태에 맞게 주행하지 못하는 경우가 있다. 따라서 딥러닝 학습을 위해 정확하게 분할된 학습데이터가 필요하다. 라벨링을 위해 수집된 차량 동작 상태 데이터를 정적 상태와 동적 상태로 분류한다. 정적 상태는 차량이 동작할 때 데이터의 대부분이 같은 값을 유지하는 경우로 직진 도로 주행 상태, 정지 상태, 곡선 도로 주행 상태가 있고, 동적 상태는 차량 동작 상태가 진행되는 동안 운전자가 차량을 계속 조작하는 상태로 차량 회전 상태와 차선 변경 주행상태가 있다.

본 논문에서는 정적 상태와 동적 상태를 차량 내부 정보를 확인하여 경험적으로 라벨링 기준을 설정하였다. 정적상태의 직진 상태일 경우 평탄하지 않은 도로나 미세한 자세제어를 하지 않는 이상은 직진 도로에서 스티어링 휠 각도가 0°를 기준으로 ±4° 정도를 유지하기 때문에 직진 도로 주행 상태로 라벨링한다. 정적상태의 곡선 도로 주행 상태일 경우 시작과 끝 부분의 스티어링 휠 각도가 동작하고, 이상적인 곡선 도로 주행 시 각도가 유지되는 게 정상적이지만 실제 도로에서는 많은 변수로 인해 완만한 곡선을 가지지 못한다. 하지만 스티어링 휠 각도의 좌우 방향이 유지되거나 스티어링 휠 각도의 차이가 크지 않으면 곡선 도로 주행으로 라벨링한다.

동적 상태의 경우 스티어링 휠 각도가 유지되는 구간이 적고 각도 값이 계속 변화되는 상태이다. 우회전 주행 상태 시 스티어링 휠 각도 변화를 보면 급격하게 각도가 변화하여 뚜렷하게 볼록한 패턴을 볼 수 있다. 따라서 0°를 기준으로 다시 0°가 될 때까지 모든 구간을 차량 회전 주행 상태로 라벨링한다. 하지만 완만하고 넓은 회전 도로 주행 시 시작과 끝 부분이 완만하게 각도가 변하여 곡선도로로 오인식 될 수 있고, 반대로 매우 짧은 구간의 곡선도로는 시작과 끝 부분이 급격하게 변하여 차량 회전 주행 상태로 오인식 할 수 있다. 따라서 차선이 모이는 위치가 정면일 경우에 곡선 도로 주행, 아닐 경우에는 회전 도로 주행 상태로 라벨링한다. 차선 변경 주행 상태는 스티어링 휠 각도가 0°를 기준으로 좌우 대칭과 같은 패턴이 만들어질 경우 차선 변경 주행 상태로 인식 하기 위해, 0°에서 시작되어 대칭을 이루고 다시 0°가 될 때까지의 범위를 모두 차선 변경 주행 상태로 라벨링하다.

### 2.6 디바이스 하드웨어 구현

본 절에서는 차량에서 데이터를 수집하고 현재 차량 동작 상태를 추론하기 위한 온-디바이스 AI의 구현에 관해 설명한다. 디바이스의 하드웨어의 구조는 차량 내부 정보 추출부, 전원을 공급하는 전원부, 차량 동작 상태 추론을 수행하는 중앙처리부로 구성되어 있다. 하드웨어 구조는 그림 4와 같다.

차량 내부 정보 추출부는 차량 OBD 커넥터와 연결되어 MCP2551 CAN트랜시버를 통해서 중앙처리 시스템인 MCU에 전달한다. 디바이스의 중앙처리부는 Espressif社의 ESP32 기반의 MCU를 사용한다. ESP32는 와이파이와 블루투스 통신 모듈이 내장되어 있어 소형화에 유리하고 초저전력으로 구현되어 소비전력에 유리하다. BLE 통신을 사용하여 스마트폰에 차량 내부 정보와 차량 동작 상태 추론 결과를 전송하여 기록하게

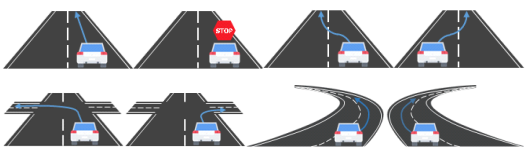



그림 3. 차량 동작 상태 정의
Fig. 3. Definition of vehicle operating status

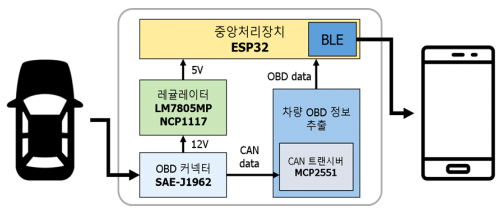



그림 4. 구현 디바이스의 시스템 다이어그램
Fig. 4. Diagrams of implemented device systems

한다. 전원부는 OBD 커넥터를 통해 차량의 배터리에서 12V 전원을 공급받아 레귤레이터를 통해 3.3V의 전원을 사용한다. 그림 5는 구현한 디바이스와 실제 차량의 OBD 포트에 연결한 사진이다.

디바이스에서 차량 동작 상태 추론을 위해 소프트웨어 최적화가 필요하다. 온-디바이스 AI 동작을 위해 차량과 CAN 통신을 사용해 데이터의 요청 및 응답을 수행하고 스마트폰으로 BLE 통신으로 데이터를 송신하고 딥러닝을 통한 추론을 수행하고 있다. 차량 동작 상태의 실시간 추론을 위해서 각각의 기능이 적절하게 수행하는 스케줄링을 통한 최적화가 필요하다. 중앙처리를 담당하는 ESP32에서 FreeRTOS를 사용하여 각각의 기능을 하나의 Task로 구현하고 우선순위를 설정하여 Task scheduler가 각 Task의 실행을 결정하게 된다. 본 논문에서는 12개의 차량 내부 정보 10Hz의 속도로 샘플링하고 있다. 차량에 너무 빠른 데이터 요청을 수행할 경우 정상적인 데이터를 응답하지 못하는 경우가 있어 내부 스케줄링이 중요하다. 따라서 차량 내부 정보 추출 Task, 딥러닝 추론 Task, BLE 통신 Task 순으로 우선순위를 높게 설정하였다.

실시간 차량 동작 상태 추론을 위해 ESP32에서 지원하는 SIMD(Single Instruction Multiple Data) 명령어를 사용하였다. 동일한 형태의 여러 데이터를 한 번에 병렬 처리하여 딥러닝 모델의 연산 속도에 이득이 되게 설계하였다. 구현한 시스템의 소프트웨어 구조는 그림 6과 같다.

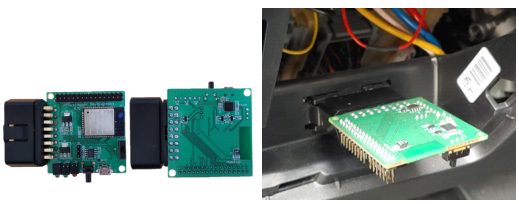


그림 5. 구현된 디바이스
Fig. 5. Implemented Devices

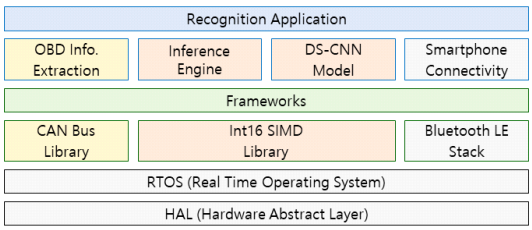



그림 6. 구현된 온-디바이스 AI의 소프트웨어 구조
Fig. 6. Software structure of the implemented device

## 2.7 경량 딥러닝 모델 설계

본 절에서는 구현한 디바이스에 최적화된 딥러닝 모델을 설계에 대해 설명한다. 본 논문에서는 사용한 모델 구조는 그림 7과 같다. 모델 구조는 1개의 CNN 레이어, 4개의 DS-CNN 레이어, Fully connected 레이어로 구성된다.

일반적으로 딥러닝망 모델을 학습할 때 사용하는 대분의 프레임워크들은 기본적으로 32bit 부동소수점(Floating Point, FP32)로 학습을 한다. 부동소수점은 실수를 컴퓨터가 표현하기 위해 사용하는 방법을 말한다. 실수는 무한한 범위를 가지기 때문에 비슷한 숫자의 이진법으로 바꿔서 표현한 뒤 정규화하여 2의 지수를 곱해 변환하는 것을 말한다. 이때 더 큰 데이터 형식을 사용할수록 소수점 이하의 표현 범위가 넓어 지지만 모델 크기가 커져서 계산량과 필요한 메모리 크기가 기하급수적으로 커지게 된다. 이것은 높은 성능이 필요해지고 추론도 오래 걸리는 문제가 발생한다. 따라서 본 논문에서는 양자화를 사용해 모델의 크기와 추론 시간을 감소를 수행한다. 32bit 부동 소수점을 16bit 고정 소수점(Fixed point) 인 정수형 데이터(int16) 로 변화하여 사용하였다. 이를 통해 모델의 크기의 경량화를 확인하였다. 그림 8은 구현한 모델의 경량화 과정이다.

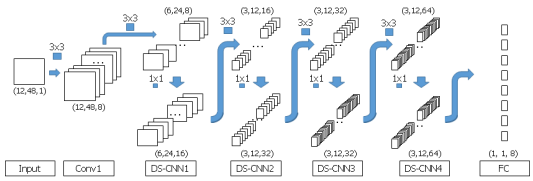



그림 7. 구현에 사용된 딥러닝 모델 구조
Fig. 7. Deep learning model structure used in implementation

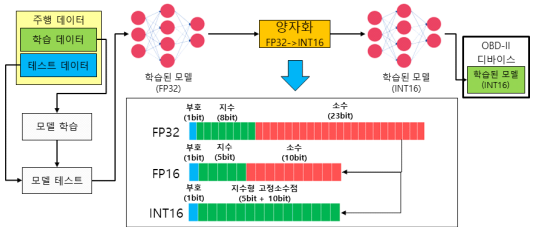



그림 8. 딥러닝 모델의 양자화 과정
Fig. 8. Quantization Process of Deep Learning Model

## Ⅲ. 성능 평가

본 장에서는 제안하는 차량 동작 상태를 인식하는 온-디바이스 AI의 성능 평가 내용을 다룬다.

### 3.1 실험

실험에 사용되는 데이터를 수집하기 위해 실제 차량을 이용하여 실험을 진행하였다. 실험에서 사용한 차량은 2019년 현대 자동차의 코나(KONA)차량이다. 차량의 OBD커넥너와 제안하는 온-디바이스 AI를 연결하여 차량 내부 정보를 수집한다. 직진, 곡선, 차량 회전, 차선 변경 상태의 주행상태가 충분히 수집되도록 수집 장소를 선택하여 데이터를 수집하였다. 수집 장소는 부산 교대역 사거리 주변과 양산 물금읍 물금리에서 데이터를 수집하였다. 그림 9는 데이터 수집에 사용된 경로이다. 데이터 수집은 데이터의 다양성을 위해서 남성 4명에서 수집했으며 교대역 사거리 주변에서 6시간 40분, 경남 양산 물금읍에서 2시간 20분, 총 9시간의 데이터를 수집하였다.

수집된 데이터는 8bit와 16bit의 크기를 가지며 센서별 해상도에 따라 센서값들이 나타난다. 센서의 종류에 따라 범위와 값의 변화량이 다르므로 정규화를 수행한다. 일반적인 최소-최대 정규화 방법은 데이터의 최대값과 최소값을 기준으로 정규화 하지만 본 논문에서는 센서 데이터의 전체 해상도 범위에서 필요한 주요 범위를 설정한다. 이후 입력 데이터를 양자화할 경우 부동소수점을 고정 소수점으로 변경하기 위해 -32~31.25로 재설정 한다. 구현한 디바이스는 차량으로부터 12종류의 차량 내부정보를 10Hz를 주기로 수집되어 윈도우 크기를 설정하여 데이터를 결합(segmentation)하여 딥러닝의 네트워크의 입력으로 사용한다. 그림 10은 입력 데이터 전처리 과정이다.

구현한 심층신경망 모델은 CNN과 DS-CNN을 사용

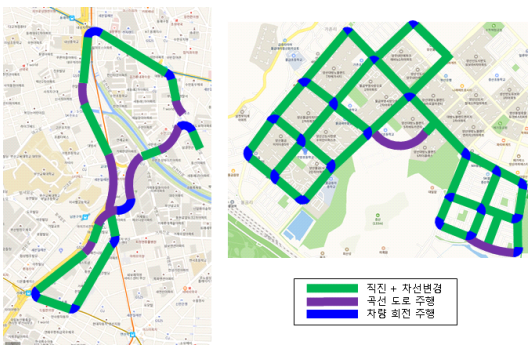



그림 9. 데이터 수집에 사용된 주행 경로
Fig. 9. Drive path used for data collection


그림 10. 입력 데이터의 전처리 과정
Fig. 10. Pre-processing of input data

하여 구현하였다. 딥러닝 모델의 입력데이터는 12개의 센서 데이터에 윈도우 크기를 설정하여 한번에 딥러닝 모델의 입력에 들어가는 데이터양을 조절하였다. 윈도우 크기는 입력 데이터의 길이가 4.8초에 해당되는 48로 설정하였을 경우 실험적으로 가장 높은 정확도를 나타내었다. 또한 모델에 적절한 파라미터값을 설정하기 위해서 실험적으로 파라미터값을 변경하며 실험을 진행하였다. 파라미터는 CNN과 DS-CNN 모델의 연산량에 영향을 주는 커널 사이즈, 채널 수, 스트라이드, 레이어의 개수를 변경해서 실험하였다. 실험 결과 커널 사이즈는 3×3, 5개의 레이어를 사용한 모델이 가장 좋은 성능을 나타내었다. 최종적으로 테스트 데이터를 사용한 선정 모델의 구성과 성능 평가는 다음 표 1과 같다. 모델의 정확도는 85.50%를 나타내었다. 그림 11은 모

표 1. 구현된 모델의 구성 및 평가
Table 1. Configuration and evaluation of the implementation model

| Model Construction | Number of Layers | Number of Channels | Kernel Size | Stride |
|---|---|---|---|---|
| | 5 | 8,16, 32, 64, 64 | 3×3 | (1,1), (2,2), (2,2), (2,2), (2,2) |
| **Evaluation** | **Accuracy** | **Inference Speed** | **Model Size** | **Lightweight Model Size** |
| | 85.50% | 19.72ms | 35.41 kByte | 17.70 kByte |

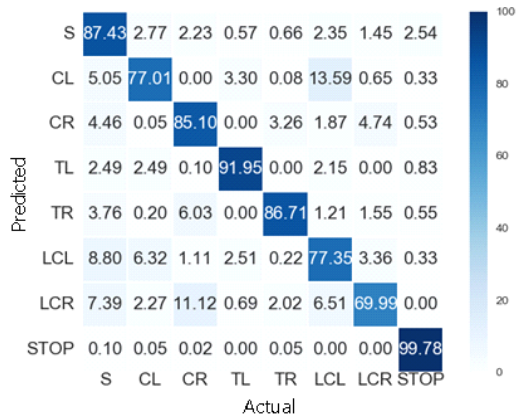



그림 11. 차량 동작 상태 추론 모델의 정확도 성능
Fig. 11. Accuracy performance of vehicle operating status inference model

델의 차량 동작 상태 정확도를 나타낸다. 구현한 차량 동작 상태 인식 성능을 평가하기 위해 실제 차량에서 주행을 통해 실시간으로 추론한 결과를 평가하였다. 그림 12는 실시간 차량 동작 상태 추론 평가를 위해 주행하였던 경로이다. 주행 후 수집한 차량 내부 정보를 사용하여 학습한 모델과 디바이스에서 구현된 경량화 모델의 실시간 추론 결과의 정확도를 비교하였다. 그림 13은 검증을 위해 수집한 차량 내부 정보를 사용하여 서버에서 학습한 모델의 정확도 성능 결과이다. 그림 14는 구현한 디바이스에서의 실시간으로 추론된 경량화된 모델의 정확도 성능이다. 서버에서 학습한 모델의 정확도는 92.81%의 정확도 성능을 가지며 디바이스에서 추론한 정확도 성능은 91.66%의 성능을 나타낸다.

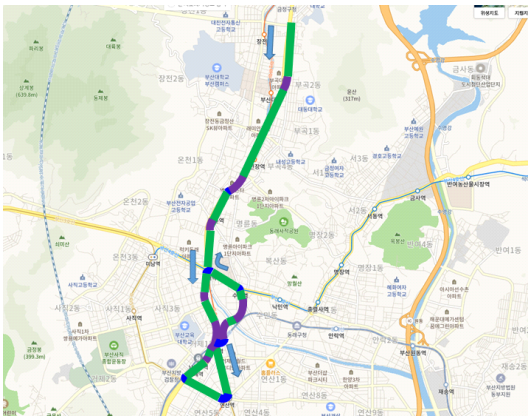


그림 12. 실시간 차량 동작 상태 추론 평가의 주행 경로
Fig. 12. Driving path of real-time vehicle motion status inference evaluation

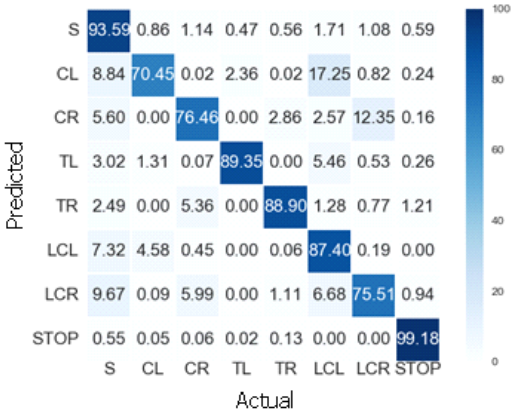



그림 13. 서버 환경에서 차량 동작 상태 추론 모델의 경량화 전 모델의 정확도 성능
Fig. 13. Accuracy performance of the pre-lightweight model of the vehicle operating status inference model in a server environment

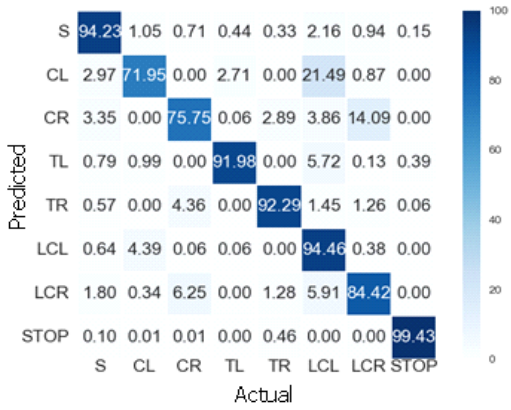



그림 14. 실제 차량 환경에서 구현된 디바이스의 차량 동작 상태 추론 모델의 경량화 모델의 정확도 성능
Fig. 14. Accuracy performance of the lightweight model of the vehicle operating status inference model of the implemented device in the actual vehicle environment

온-디바이스 AI에서는 데이터 추출, 데이터 전처리, 딥러닝 모델 추론, 데이터 통신과 같은 다양한 프로세스를 가지기 때문에 프로세스간 소요시간을 최소화 해야 한다. 구현된 온디바이스 AI는 10Hz로 데이터를 수집하고 있다. 심층신경망 모델을 경량화 한 후 온-디바이스 AI에서 추론속도는 19.72ms로 측정되었다. 실시간 운전자 행동을 인식하기 위해서 189ms[16]보다 빨라야 하며 구현된 디바이스는 센서 데이터를 10ms 주기로 데이터를 수집하기 때문에 추론 시간을 40ms 이내로 수행해야 한다. 구현된 온-디바이스 AI의 추론 속도는 19.72ms로 실시간 구현이 가능함을 확인하였다. 온-디바이스 AI에서 모델크기는 서버에서의 모델크기 35.41kByte와 비교해 경량화 후 17.70kByte로 크기가 감소한 것을 확인하였다.

## Ⅳ. 결 론

본 논문에서는 차량의 현재 동작 상태를 인식하기 위해 차량에서 실시간 데이터 수집과 동시에 추론을 수행하는 디바이스를 설계 및 구현하였다. 디바이스의 하드웨어는 차량의 OBD 커넥터를 통해서 전원을 공급받고 CAN 통신을 통해 차량 내부 정보를 추출을 수행하고 운용을 위해 저전력으로 설계되었다. 제안하는 시스템은 차량 동작 상태를 인식하기 위해 딥러닝 모델을 학습하여 사용한다. 저전력 소형 디바이스에서 딥러닝 사용을 위해 딥러닝을 경량화 하여 탑재할 필요가 있다. 제안하는 시스템에서 딥러닝 모델을 경량화 하기 위해 DS-CNN 모델 구조를 사용하여 파라미터의 수를 감소

시키고 파라미터의 양자화를 통해 모델의 크기를 감소하는 방법을 사용하였다. 모델의 학습을 위해 실제 차량에서 12종류의 차량 내부 정보를 수집하였고 8가지의 차량 동작 상태를 추론하였다. 파라미터를 변경하여 다양한 모델을 학습하였으며 테스트 데이터를 사용하여 85.50%의 정확도 성능의 모델을 최종적으로 사용하였다. 차량 동작 상태의 실시간 추론 성능을 평가하기 위해 실제 차량에서 주행하면서 데이터 수집과 추론을 수행하였고 성능을 평가하였다. 디바이스에서의 정확도 성능은 91.66%로 서버에서 경량화 전의 모델인 92.81%의 정확도 성능과 비교하였을 때 낮은 성능 하락을 확인하였다. 또한 모델의 크기와 추론속도를 고려하였을 때 실제 차량에서 사용하는 목표에 맞게 효율적으로 구현되었음을 확인하였다.

## Ⅴ. 한계점 및 추후 연구

본 논문의 주요 구현 내용은 딥러닝을 경량화하여 현재 차량 동작 상태를 인식하는 온-디바이스 AI의 구현이다. 복잡하고 깊은 딥러닝 모델을 경량화하는 것은 중요한 연구내용이지만 본 논문에서는 모델 경량을 최대화하는 연구가 아닌 특정 소형 저사양 MCU에 딥러닝 모델을 최적화하여 구현하는 것이다. 따라서 딥러닝 모델의 경량 기법에 대한 평가 방법은 고찰이 부족하다. 온-디바이스 AI 구현을 위해서는 딥러닝 모델 구현, 모델 경량화, 딥러닝 모델 포팅(porting), 최적화 등 구현 절차가 존재하고 각 요소 기술이 필요하다. 그러므로 평가를 위한 비교군을 구현하기 위해 모델을 변경하거나 하드웨어가 변경되면 구현을 위한 시간과 기술이 요구합니다. 또한 본 논문에서 구현에서 사용한 딥러닝 모델은 MCU 탑재를 위해 처음부터 단순하고 작은 모델을 사용하였다. 따라서 경량화 후의 모델과 비교하여 급격한 차이가 크지 않았다. 구현 환경에도 서버(PC)와 온-디바이스 AI의 연산 성능의 차이가 크기 때문에 비교 평가방법에 적절하지 않다고 판단되었다. 그러므로 본 논문에서는 모델 비교를 위해 정확도, 추론속도, 모델 크기만 고려하였다. 추후 연구에는 딥러닝 모델 비교를 위해 동일환경에서 에뮬레이터를 구축하여 경량화 전후의 손실 추이에 대한 분석을 수행이 필요하다.

## References


[1] S. Moosavi, "Driving style representation in convolutional recurrent neural network model of driver identification," *ArXiv preprint arXiv: 2102.05843*, 2021. (https://doi.org/10.48550/arXiv.2102.05843)

[2] G. Ren, "A new lane-changing model with consideration of driving style," *Int. J. Intell. Transport. Syst. Res.*, vol. 17, no. 3, pp. 181-189, 2019. (https://doi.org/10.1007/s13177-019-00180-7)

[3] D. Kim, "Driving style-based conditional variational autoencoder for prediction of ego vehicle trajectory," *Sensors*, vol. 20, Sep. 2020. (https://doi.org/10.3390/s20185030)

[4] S. Ullah, "Lightweight driver behavior identification model with sparse learning on in-vehicle CAN-BUS sensor data," *Sensors*, vol. 20, no. 18, Sep. 2020. (https://doi.org/10.3390/s20185030)

[5] Y. Wang, "Driver identification leveraging single-turn behaviors via mobile devices," *ICCCN*, pp. 1-9, 2020. (https://doi.org/10.1109/ICCCN49398.2020.9209713)

[6] National Center for Statistics and Analysis, *NHTSA Traffic Safety Facts*, US Department of Transportation, 2015. (https://anstse.info/nhtsa-traffic-safety-facts)

[7] A. Mammeri, "Design of a semi-supervised learning strategy based on convolutional neural network for vehicle maneuver classification," *IEEE Int. Conf. WiSEE*, vol. 20, no. 18, pp. 65-70, Sep. 2020. (https://doi.org/10.1109/WiSEE.2019.8920301)

[8] A. Mammeri, "Detecting lane change maneuvers using SHRP2 naturalistic driving data: A comparative study machine learning techniques," *Accident Analysis & Prevention*, vol. 142, pp. 65-70, Sep. 2020. (https://doi.org/10.1016/j.aap.2020.105578)

[9] J. Lim, et al., "ID extraction OBD-II vehicle operation information for driver behavior analysis," in *Proc. KICS Winter Conf.*, vol. 2018, pp. 100-101, Jan. 2018.

[10] C. Kim, et al., "Design and implementation of Real-time ECU sensor information acquisition system using UDS protocol," in *Proc. KICS*

*Winter Conf.*, vol. 17, no. 1, pp. 550-551, Feb. 2020.

[11] X. Xuan, "Lane change maneuver detection from probe vehicle DGPS data," *IEEE Intell. Transport. Syst. Conf.*, pp. 624-629, 2006. (https://doi.org/10.1109/ITSC.2006.1706811)

[12] Z. Yang, "Lane-change detection from steering signal using spectral segmentation and learning-based classification," *IEEE Trans. Intell. Veh.*, vol. 2, pp. 14-24, Mar. 2017. (https://doi.org/10.1109/TIV.2017.2708600)

[13] L. Xiong, "IMU-Based automated vehicle body sideslip angle and attitude estimation aided by GNSS using parallel adaptive kalman filters," *IEEE Trans. Veh. Technol.*, vol. 69, pp. 10668-10680, Oct. 2020. (https://doi.org/10.1109/TVT.2020.2983738)

[14] A. Narayanan, "Gated recurrent fusion to learn driving behavior from temporal multimodal data," *IEEE Robotics and Automat. Lett.*, vol. 5, no. 2, pp. 1287-1294, Arp. 2020. (https://doi.org/10.1109/LRA.2020.2967738)

[15] R Soni, "Analysis of overtaking patterns of Indian drivers with data collected using a LiDAR," *Transport. Res. Part F: Traffic Psychology and Behaviour*, vol. 74, pp. 139-150, Oct. 2020. (https://doi.org/10.1016/j.trf.2020.08.016.)

[15] M. T. Corfitsen, "Tiredness and visual reaction time among young male nighttime drivers: A roadside survey," *Accident Analysis & Prevention*, vol. 26, no. 5, pp. 617-624, Oct. 1994. (https://doi.org/10.1016/0001-4575(94)90023-X.)

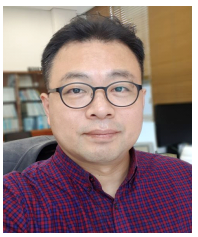
**김 태 구** (Taegu Kim)

2017년 2월 : 인제대학교 전기전자컴퓨터공학과 공학학사
2020년 9월 : 부산대학교 컴퓨터공학과 석사
2020년 3월~현재 : 부산대학교 컴퓨터공학과 박사과정
<관심분야> 임베디드 시스템, 임베디드 AI
[ORCID:0000-0003-0033-8370]

**조 용 훈** (Yonghun Cho)

2020년 2월 : 동의대학교 컴퓨터공학과 공학학사
2020년 9월~현재 : 부산대학교 정보융합공학과 석박사통합과정
<관심분야> 임베디드 시스템, 임베디드 AI
[ORCID:0000-0002-5838-2959]

**백 윤 주** (Yunju Baek)



1990년 2월 : 한국과학기술원 전산학과 공학학사
1992년 2월 : 한국과학기술원 전산학과 공학석사
1997년 2월 : 한국과학기술원 전산학과 공학박사
1999년~2002년 : 네이버 CTO
2003년~현재 : 부산대학교 정보컴퓨터공학과 교수
<관심분야> 임베디드 시스템, 임베디드 AI
[ORCID:0000-0002-3873-2624]