TinyML 기반 저전력 PPG 신호 분석을 통한 혈압 추정 디바이스 설계

권문희·두진원

TinyML-Based Low-Energy Device for Blood Pressure Estimation Using PPG Signal Analysis

Moon-Hee Kwon·Jin-Won Doo

요 약

1. 서론

1.1. 연구 배경 및 필요성

혈압은 인체 건강 상태를 반영하는 가장 중요한 생리 신호 중 하나로, 고혈압은 뇌졸중, 관상동맥질환, 당뇨병 등 다양한 **만성질환의 주요 위험 요인**으로 알려져 있다. 따라서 **정확하고 연속적인 혈압 측정**은 질병의 예방과 조기 진단, 그리고 치료 효과 평가에 있어 필수적이다. 그러나 혈압은 신체 활동, 감정 변화, 수면 패턴 등 다양한 요인에 의해 시시각각 변동하므로, 병원에서의 일회성 측정만으로는 개인의 실제 혈압 상태를 제대로 반영하기 어렵다. 이로 인해 백의 고혈압(white coat hypertension)이나 가면 고혈압(masked hypertension)과 같은 문제가 발생하며, 이는 **불필요한 치료 혹은 치료 지연으로** 이어질 수 있다.[1)]

기존 혈압 측정 방식은 크게 침습적·비침습적 방법으로 나뉜다. **침습적 방식**은 정확도가 높으나 감염 위험과 불편함으로 인해 일상적 모니터링에 적합하지 않으며, **비침습적 방식**인 cuff 기반 진동법 역시 부피와 불편감으로 인해 연속적 측정에 한계가 있다. 이에 따라 최근에는 PPG(Photoplethysmography)와 같은 광학적 생체 신호를 활용하여 혈압을 추정하는 cuffless 방식이 활발히 연구되고 있다.[2)] 특히 스마트워치와 같은 모바일 기기 기반 혈압 측정 기술은 병원 외부에서의 장시간 모니터링을 가능하게 하여 고혈압 관리에 새로운 가능성을 제시하고 있다.

그러나 이러한 웨어러블 디바이스는 높은 연산 자원과 배터리 소모 문제, 그리고 주기적 재보정 필요성 등의 한계가 남아 있다. 따라서 저전력 환경에서도 실시간 혈압 추정이 가능하도록, 경량화된 모델을 활용하는 TinyML 기반 접근이 요구된다. TinyML은 **제한된 하드웨어 자원**에서도 머신러닝 모델을 효율적으로 동작시킬 수 있어, 저전력 웨어러블 디바이스에 최적화된 기술이다. 이에 본 연구는 TinyML 기반 저전력 PPG 신호 분석을 통한 혈압 추정 디바이스 설계를 목표로 한다. 이는 기존의 cuff 기반 방식의 불편함을 해소하면서도, 연속적이고 개인 맞춤형 혈압 관리가 가능한 차세대 헬스케어 솔루션으로서 의학적, 공학적 유용성을 가진다.

1.2. 연구 목표

본 연구의 최종 목표는 PPG 신호를 수집하는 착용형 디바이스를 설계하고, 연결된

---

1) Lee, H.-Y. (2024). 고혈압 진료에서의 모바일 디바이스 혈압계 이용의 현재와 미래 [A novel cuffless blood pressure monitoring device in hypertension care]. Journal of the Korean Medical Association, 67(7), 475-481.

2) Seo, Y., Kwon, S., Sunarya, U., Park, S., Park, K., Jung, D., Cho, Y., & Park, C. (2023). Blood pressure estimation and its recalibration assessment using wrist cuff blood pressure monitor. Biomedical Engineering Letters, 13, 221-233.

MCU에서 구동 가능한 TinyML 모델을 구현하여, 별도의 cuff 없이 실시간으로 혈압을 추정,표시하는 통합 시스템을 완성하는 데 있다.

1. 이를 위해 먼저 UCI 공개 데이터셋의 PPG-혈압 데이터를 MATLAB 환경에서 체계적으로 전처리하고, **LSTM과 GRU 기반의 순환신경망 모델을 구축,비교**하여 PPG 단일 신호만으로 평균동맥압을 안정적으로 추정하는 모델을 도출한다. 전처리는 0.5-8 Hz Bandpass Filtering, 구간 정규화 및 박동 단위 segmentation을 포함하며, 파형의 상승 시간, 첨두 간 간격, 맥파 폭/기울기, 파생 곡선 특성 등 시간,형태 기반 특성들을 후보로 정의한다. 모델 설계 단계에서는 입력 window 크기, hidden layer 수 및 unit 수, learning rate 등을 그리드/베이지안 탐색으로 최적화하여 일반화 성능을 검증한다. 성능 평가는 MAE/RMSE, Pearson correlation coefficient 분석을 사용하며, 기준 장비와의 오차 한계가 임상 적용 가이드라인을 향해 수렴하도록 목표치를 설정한다. 또한 모델 구축 후에는 실제 실험자의 PPG 신호를 적용해 초기 정확도를 검증하며, 이 정확도를 기준선으로 삼아 이후 경량화 과정을 거친 뒤에도 유사한 수준의 정확도를 유지할 수 있는지를 중점적으로 확인한다.

2. 도출된 모델은 MCU 상시 구동을 위해 정수 양자화(INT8), 가중치 pruning, 연산자 최적화(CMSIS-NN 등)를 적용해 **모델 크기와 지연 시간을 최소화**한다. 이때 연산 복잡도, flash 점유, runtime, RAM 사용량을 정량화하고, 목표 사양(모델 크기 < 200 KB, RAM < 128 KB, 1 Hz 이상 업데이트, 단말 지연 < 200 ms)을 만족하도록 반복 개선한다. 전력 소비는 sampling frequency, window 처리 주기, LED 구동 duty ratio, BLE 송신 주기 등을 변수로 하여 시나리오별 평균, 피크 전류를 측정하고, 기능별 예산(센서, 연산, 무선)을 책정해 총 배터리 수명을 추정한다.

3. 하드웨어는 PPG 센서 모듈과 이를 구동할 수 있는 MCU Board를 중심으로 구성한다. 손가락 또는 손목 측정 폼팩터를 검토하여 광학 경로와 기계적 고정을 최적화하고, sampling frequency(100-200 Hz)와 LED 구동 전류 설정을 결정한다. Firmware는 실시간 파형 수집, 신호 전처리, TinyML 추론, 결과 formatting 및 표시로 이루어지며, 결과 표시 방식은 보드 장착 OLED/segment display 또는 BLE를 통한 mobile application 연동 중 하나 혹은 병행으로 구현한다.

4. **실시간 측정 및 정확도 검증은** 나노 보드에서 동일 전처리,모델을 구동한 상태로, 기준 혈압계와 동시 측정 protocol을 설계해 진행한다. 정적 상태, 가벼운 움직임, 체위 변화 등 실제 사용 시나리오를 포함하여 sample을 수집하고, on-device 추정치와 기준치의 오차 통계를 산출한다. 환경,피험자 다양성(피부 톤, 체온 등)에 따른 민감도를 분석하여 보정 규칙이나 적응형 임계값을 도입하며, 소프트웨어 업데이트로 반영 가능한 개선 항목을 선택하여 정리한다.

1.3. 논문 구성

본 보고서는 다음과 같이 구성된다. 2장에서는 관련 이론및 선행 연구를 고찰하고, 3장에서는 모델 설계, 경량화 기법, MCU 탑재 과정을 상세히 기술한다. 4장에서는 PC 및 MCU 환경에서의 실험 결과를 비교 분석하며, 5장에서는 결론을 맺는다.

2. 이론적 배경

2.1. PPG 및 혈압 추정 원리

Photoplethysmography(PPG)는 피부 표면에 빛을 조사하고, 조직 내 혈액 용적 변화에 따라 반사 또는 투과되는 광량 변화를 측정하는 비침습적 생체 신호 측정 방식으로, 심박과 말초 혈류역학을 반영하는 대표적 광학 신호이다. 심장이 수축할 때 말초 혈관으로 전달되는 맥파는 혈관 직경 및 혈액량을 순간적으로 증가시키며, 이로 인해 검출기에 도달하는 광 신호의 변화가 발생한다. 이러한 파형은 심박수뿐 아니라 혈관탄성도, 말초 저항 및 맥파의 전도 특성 등 혈류역학적 요소를 포함하고 있어, 단순한 심혈관계의 활성 지표를 넘어 혈압 변동을 추정할 수 있는 정보원을 제공한다. 특히 PPG 파형의 상승 시간, 첨두 간 간격, 파형의 기울기 및 반사파의 존재 여부 등 파형의 형태학적 특징(morphology)은 동맥의 탄성도와 맥파반사 특성과 밀접하게 연관되며, 이는 수축기·이완기 혈압의 변화에 따라 민감하게 변하는 특성을 보인다.

PPG 기반 혈압 추정 방식은 전통적으로 맥파전달시간(Pulse Transit Time, PTT) 또는 맥파전달속도(Pulse Wave Velocity, PWV)를 이용한 방식이 우세하였다. 이 방법은 심장에서 말초까지 맥파가 이동하는 시간을 측정함으로써 동맥 강직도 및 혈압과의 상관관계를 이용하는 것으로, ECG와 PPG를 조합하거나, 서로 다른 위치의 PPG 센서를 사용하여 두 지점 간 시간 지연을 산출하는 구조를 가진다. PTT는 혈압과 반비례하는 경향이 있어 혈압 추정에 유용하지만, 다중 센서, 정밀한 시간 동기화, 체위 및 생리 상태 변화에 대한 높은 민감도 등 구현상의 한계가 존재하였다. 이러한 이유로 최근 연구 흐름은 단일 PPG 파형으로부터 혈압을 추정하는 파형 분석(Pulse Wave Analysis, PWA) 기반 접근으로 이동하고 있다.

PWA 기반 접근은 단일 PPG 신호의 형태학적 특징 및 비선형 동적 변화를 분석하여 혈압과의 관계를 학습하는 방식으로, 심혈관계 상태를 반영하는 다양한 특성(예: 파형의 상승기, 반사파 도달 시간, 파형의 기울기, 파생 파형의 피크 구조 등)을 활용한다. 특히 딥러닝 기반 모델은 이러한 복잡한 비선형 관계를 효과적으로 학습할 수 있어, 최근 웨어러블 환경에서 유망한 대안으로 주목된다. El-Hajj [3]등

3) C. El-Hajj, G. Aya, and F. Karameh, “Cuffless blood pressure estimation from PPG signals and its derivatives using deep learning models,” BioMedical Engineering OnLine, vol. 20, no. 1, pp. 1-27, 2021, doi: 10.1186/s12938-020-00845-1.

(2021)은 단일 PPG 신호와 그 1차 및 2차 미분 신호를 입력으로 하는 LSTM 기반 모델을 제안하여, 공개 데이터셋을 통해 SBP와 DBP를 전통적인 임상 기준에 부합하는 수준으로 추정할 수 있음을 보고하였다. 이 연구는 단일 센서 기반 혈압 추정의 가능성을 보여주었을 뿐 아니라, 파형의 파생 신호가 혈압 예측에 유용한 정보를 제공함을 실증했다는 점에서 의미가 있다.

한편, 임상적으로 적용 가능한 지속적 혈압 모니터링을 위해서는 개별 사용자에 대한 반복적인 보정 의존도를 줄이는 것이 중요한 과제이다. 이와 관련하여 Joung 등(2023)은 손목형 PPG 신호만을 이용하여 사용자 보정 없이 혈압을 추정하는 딥러닝 모델을 제안하였고,[4] 대규모 데이터를 활용한 검증을 통해 국제적 기준(AAMI/BHS)을 충족하는 성능을 보고하였다. 이 연구는 단일 채널, 단일 센서 기반의 연속적 혈압 추정이 실생활 환경에서도 가능함을 보여주었으며, 웨어러블 디바이스 중심의 헬스케어 기술 발전 흐름과 부합한다는 점에서 주목받고 있다.

이러한 연구들은 PPG 파형이 단순한 심혈관 생체신호를 넘어 혈압 추정을 위한 풍부한 정보를 제공할 수 있음을 뒷받침하며, 특히 딥러닝 및 기계학습 기반 모델이 파형에 내재된 복잡한 비선형 구조를 활용하여 높은 정확도를 달성할 수 있다는 가능성을 보여주었다. 그러나 PPG 기반 혈압 추정은 여전히 피부 특성, 센서 압력, 체위 변화, 말초 혈관 저항 등 다양한 외부 요인에 영향을 크게 받으며, 개인 간 차이 및 시간에 따른 생리적 변화를 적절히 대응하기 위한 모델의 일반화 성능 확보가 핵심 과제로 남아 있다. 따라서 임상적 활용을 위해서는 정확도 향상뿐 아니라, 다양한 사용 환경에서의 안정적인 측정 성능, 저전력 구현, 실시간 처리 능력 등이 동시에 고려되어야 한다.

2.2. TinyML

TinyML(Tiny Machine Learning)은 마이크로컨트롤러, 센서 노드, 웨어러블 디바이스와 같은 자원 제한형 임베디드 시스템에서 머신러닝 모델을 직접 실행하는 기술 분야를 의미한다. 기존 머신러닝은 고성능 연산 자원을 기반으로 데이터를 처리하였기 때문에, 데이터 전송 지연, 연결성 문제, 개인정보 유출 가능성, 전력 소모 증가 등 여러 한계를 가지고 있었다. TinyML은 이러한 문제를 해결하기 위해 데이터를 클라우드로 전송하지 않고, 디바이스 내부에서 실시간으로 처리하는 방식을 목표로 한다.

TinyML의 핵심은 모델의 정확도를 손상시키지 않는 범위 내에서 연산량과 메모리 사용량을 극도로 줄이는 것이다. 이를 위해 정수 양자화, 가중치 가지치기, 연산 최적화와 같은 경량화 기술이 활용되며, 이러한 기법을 통해 모델은 수십에서 수백

---

4) H. Joung, J. Lee, and Y. Park, “Continuous cuffless blood pressure monitoring using wristband photoplethysmography and calibration-free deep learning,” Scientific Reports, vol. 13, Art. no. 4321, 2023, doi: 10.1038/s41598-023-35492-y.

KB 수준의 플래시 메모리와 제한된 RAM을 가진 마이크로컨트롤러에서도 실시간으로 실행될 수 있다. 또한 초저전력 환경에서도 동작할 수 있어, 배터리 기반의 장치에서 장시간 운용이 가능하다는 장점이 있다.
TinyML은 데이터 처리 지연을 최소화하고, 개인정보 보호를 강화하며, 네트워크 연결이 불안정한 환경에서도 안정적으로 동작할 수 있다는 점에서 실시간 모니터링 시스템에 적합하다. 이러한 특성 때문에 헬스케어, 웨어러블 기기, 스마트 홈, 농업, 산업 자동화 등 다양한 분야에서 활용 가능성이 확대되고 있다. 반면 TinyML을 실제 시스템에 적용하기 위해서는 모델 성능뿐 아니라 플래시 용량, RAM 크기, 처리 지연시간, 전력 소모 등 시스템 수준의 제약을 동시에 고려해야 한다. 또한 센서 구동 방식, 신호 샘플링 빈도, 무선 통신 주기와 같은 요소는 전체 전력 예산에 직접적인 영향을 미치므로, 소프트웨어와 하드웨어를 통합적으로 설계할 필요가 있다.
생체 신호 기반 헬스케어 분야에서 TinyML은 고성능 알고리즘을 휴대 가능한 장치에 직접 적용할 수 있는 실용적 대안으로 주목받고 있다. 생체 신호는 잡음과 변동성이 크기 때문에 복잡한 모델이 필요하지만, 웨어러블 기기에서는 작은 메모리, 낮은 전력 소모, 실시간 처리 능력이 요구된다. 이러한 이유로 최근에는 순환 신경망, 경량화된 합성곱 신경망 또는 이를 압축한 하이브리드 모델을 마이크로컨트롤러에서 동작 가능한 형태로 변환하려는 연구가 활발히 이루어지고 있다.

본 연구에서는 TinyML을 활용하여 PPG 신호로부터 혈압을 추정하는 모델을 마이크로컨트롤러에서 실시간으로 구동하는 것을 목표로 한다. 이를 위해 학습된 모델을 정수 양자화하고, 가중치 및 연산 구조를 최적화하여 제한된 메모리와 전력 환경에서도 안정적으로 동작할 수 있도록 설계할 예정이다. 또한 초저전력 환경에서 1초 이상의 주기를 갖는 연속 추론을 수행하고, 모델 단축 후에도 기존 PC 환경과 유사한 정확도를 유지하는 것을 주요 성능 목표로 설정하였다. 이러한 접근은 웨어러블 장치가 실시간으로 혈압을 측정하고 사용자에게 피드백을 제공할 수 있도록 하여, 기존 커프 기반 방식의 불편함을 줄이고, 장시간 모니터링이 가능한 새로운 형태의 헬스케어 솔루션 구현을 가능하게 한다.

2.3. nRF52840 + Segger Embedded Studio for ARM

nRF52840은 Nordic Semiconductor에서 개발한 ARM Cortex-M4 기반 초저전력 무선 마이크로컨트롤러로, BLE, Thread, ZigBee, ANT, IEEE 802.15.4 등 다양한 무선 프로토콜을 지원하는 것이 특징이다. 이 칩은 64MHz의 프로세서와 부동소수점 연산 유닛(FPU), 최대 1MB의 플래시 메모리, 256KB의 RAM을 제공하여, 저전력 환경에서도 비교적 복잡한 연산 및 실시간 데이터 처리가 가능하다. 특히, FPU를 내장하고 있다는 점은 신호 처리 및 경량화된 딥러닝 모델과 같은 수학적 연산이 많은 알고리즘을 구현하는 데 유리하다. 전력 측면에서는 광고 간격, 송신 전력,

슬립 모드 구성에 따라 수십 μA 수준까지 평균 전류를 낮출 수 있어, 배터리 기반 웨어러블 장치의 장시간 운용에 적합하다.

nRF52840은 다양한 주기역 아날로그·디지털 인터페이스를 지원하며, 저잡음 ADC(Analog-to-Digital Converter)를 통해 생체 신호 센서 데이터를 안정적으로 수집할 수 있다. 이와 함께 독립적인 전력관리 유닛과 고도로 최적화된 무선 스택을 제공하여, 센서 측정, 신호 처리, 통신 기능을 병렬적으로 운영하면서도 전체 소모 전력을 최소화할 수 있다. 특히 BLE 통신이 내장되어 있어, 생체 신호 측정 데이터 또는 혈압 추정 결과를 모바일 기기와 실시간으로 연동할 수 있다는 점은 웨어러블 응용 연구에서 중요한 장점으로 작용한다.

Segger Embedded Studio for ARM은 nRF52840을 비롯한 ARM 기반 MCU 개발을 위한 통합 개발 환경으로, C/C++ 컴파일러, 디버거, 프로젝트 빌드 시스템, 프로파일링 도구 등을 포함한다. 이 환경은 Nordic SDK와 연동되어 동작하도록 최적화되어 있으며, BLE 스택, 전력 관리, 주변장치 제어 등 다양한 기능을 라이브러리 형태로 제공한다. 또한 Segger의 J-Link 디버거와 결합될 경우, 저수준 메모리 접근, 실시간 추적, 전력 소비 분석과 같은 정밀 디버깅이 가능하여, 임베디드 시스템 개발 과정에서 오류 분석과 성능 최적화를 효율적으로 수행할 수 있다.

Segger Embedded Studio는 코드 사이즈 최적화 기능과 정적 분석 도구를 제공하여, 제한된 플래시 메모리를 사용하는 환경에서 실행 파일의 크기를 억제하는 데 도움이 된다. 이러한 특성은 TinyML 기반 모델을 포함한 복잡한 소프트웨어 모듈을 MCU 내부에서 실행해야 하는 본 연구에서 중요한 요소로 작용한다. 또한 실시간 디버깅을 통해 센서 데이터 수집 주기, 전처리 과정, 모델 추론 루틴의 지연 시간 등을 확인할 수 있어, 시스템 전체의 성능 평가와 병목 지점을 분석하는 데 유용하다.

본 연구에서는 nRF52840을 핵심 플랫폼으로 활용하여 PPG 신호를 실시간으로 수집하고, 이를 임베디드 환경에서 전처리 및 TinyML 모델을 통해 혈압을 추정하는 파이프라인을 구현한다. Segger Embedded Studio는 이러한 소프트웨어 개발의 주 개발 환경으로서, 펌웨어 작성, 모델 이식, 디버깅 및 성능 측정에 사용될 예정이다. 특히 BLE 통신 기능은 혈압 추정 결과를 외부 장치로 전송하는 데 활용되며, 초저전력 하드웨어 구조는 연속적인 생체 신호 모니터링과 장시간 운용을 가능하게 한다. 이러한 조합은 웨어러블 환경에서 실시간 혈압 모니터링을 구현하기 위한 하드웨어 및 개발 환경으로서 적합한 선택이라 할 수 있다.

3. 연구 방법 및 시스템 구현

3.1. UCI Dataset

본 연구에서는 PPG 신호로부터 혈압을 추정하는 모델을 학습하기 위해 UCI에서

공개된 cuffless blood pressure estimation dataset을 사용하였다. 이 데이터셋은 광용적맥파(PPG)와 동맥혈압(ABP) 신호를 동일 시간 축에서 동시에 기록한 생체 신호 데이터로 구성되어 있으며, 커프(cuff) 기반 혈압 측정 없이도 혈압을 추정하는 알고리즘 개발을 목적으로 제공된 대표적인 공용 데이터셋 중 하나이다. 데이터는 원래 병원 환경에서 다수의 피험자를 대상으로 수집된 것으로, 약 900명 이상의 환자 기록을 포함하며, 총 수천 개의 신호 세그먼트로 구성되어 있다. 각 신호는 125 Hz의 샘플링 주파수로 측정되어 비교적 높은 시간 해상도를 가지며, 이를 통해 맥파의 세부적인 형태 변화 및 혈압 변동을 보다 정밀하게 분석할 수 있다.

데이터셋의 가장 큰 특징은 PPG 신호뿐 아니라 실제 동맥혈압 파형이 함께 제공된다는 점으로, 이를 통해 PPG 신호로부터 혈압을 회귀(regression) 형태로 직접 예측할 수 있는 학습 환경이 마련된다. 특히 ABP 신호는 수축기·이완기 혈압뿐 아니라 평균동맥압(MAP) 산출에도 사용될 수 있어, 다양한 혈압 관련 지표를 생성하는 것이 가능하다. 이러한 동시 측정 구조는 PPG와 혈압 사이의 시계열적 관계를 모델이 직접 학습할 수 있도록 해주기 때문에, 단순한 맥박 기반 추정보다 훨씬 높은 표현력을 가진 모델을 설계할 수 있다는 장점이 있다. 또한 데이터가 다양한 길이의 세그먼트로 분할되어 제공되기 때문에, 윈도우 기반 시계열 분석, 슬라이딩 처리, 또는 비트(beat)−단위 분석을 수행하기 용이하다.

UCI dataset은 공개 데이터셋이기 때문에 접근성이 높고, 다른 연구들과의 비교 및 재현성 확보에 도움이 된다. 데이터의 양이 충분히 크고 다양한 환자군을 포함하고 있어, 머신러닝 및 딥러닝 기반 모델의 학습과 일반화 능력 평가에 유리한 환경을 제공한다는 점 또한 중요한 장점이다. 특히 본 연구처럼 단일 PPG 신호로부터 혈압을 추정하는 기법을 개발할 때, 대규모·실측 기반 데이터셋은 모델의 성능을 객관적으로 검증하고 신뢰성을 확보하는 데 필수적이다.

그러나 해당 데이터셋은 환자 개별의 생리적 특징이나 상태 정보를 포함하지 않기 때문에, 피험자 특성에 따른 혈압 변동성이나 모델의 개인별 일반화 능력을 명확히 분석하는 데는 제한이 있다. 또한 병원 환경에서 측정된 신호라는 특성상, 실제 웨어러블 환경에서 발생할 수 있는 움직임, 센서 압력 변화, 외부 잡음 등과는 차이가 존재한다. 따라서 UCI dataset 기반의 학습 결과를 실제 착용형 장치 환경으로 확장하기 위해서는 추가적인 데이터 수집과 검증 과정이 필요하다.

그럼에도 불구하고 UCI dataset은 PPG 기반 혈압 추정 연구를 위한 표준적인 벤치마크 데이터셋으로 널리 활용되고 있으며, 본 연구에서도 모델 설계와 학습, 평가의 초기 단계에서 안정적인 기반 데이터로 사용되었다. 본 연구는 이 데이터셋을 통해 구축된 대규모 윈도우 기반 데이터셋을 활용하여 모델의 성능을 검증하고, 이후 이를 임베디드 환경에서 경량화된 형태로 구현함으로써 실제 웨어러블 디바이스에 적용 가능한 형태로 확장하는 것을 목표로 한다.

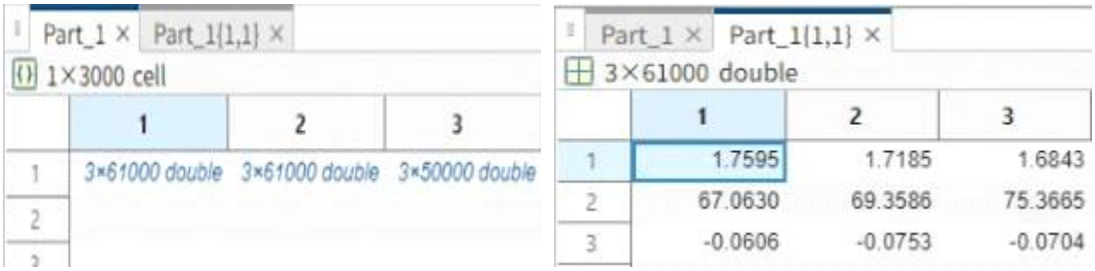

Fig1. UCI dataset Raw data format

3.2. 데이터 수집 및 전처리

전체 전처리 과정은 (1) 원시 MATLAB 형식 데이터 로딩, (2) 유효 구간 필터링 및 PPG/ABP 추출, (3) 대역통과 필터링 및 슬라이딩 윈도우 분할, (4) 정규화 및 라벨(MAP) 생성, (5) 학습/검증/시험 세트 분할 및 HDF5 파일 저장으로 구성된다.

먼저, 샘플링 주파수는 데이터셋 특성에 맞추어 125 Hz로 설정하고, 8초 길이의 분석 윈도우를 사용하였다. 이에 따라 하나의 윈도우는 1000개의 시계열 샘플(FS × WIN_SEC = 125 × 8)로 구성된다. 연속 신호를 윈도우 단위로 나누기 위해 윈도우 간 겹침은 50%로 설정하였으며, 이는 1000샘플 길이의 윈도우를 500샘플씩 이동시키며 추출하는 방식으로 구현되었다. 원본 데이터는 “Part_1.mat”부터 “Part_4.mat”까지 네 개의 파일로 분할되어 있으며, 각 파일을 순차적으로 읽어들여 동일한 전처리 과정을 적용하였다.

데이터 로딩 단계에서는 h5py를 이용하여 MATLAB 형식의 .mat 파일을 열고, 내부에 “Part”라는 키를 포함하는 상위 그룹을 찾아 각 세그먼트에 접근하였다. 세그먼트는 참조 테이블(“#refs#”)을 통해 실제 데이터 배열에 연결되며, 각 세그먼트는 2차원 배열 형태로 저장되어 있다. 이 중 첫 번째 열은 PPG 신호, 두 번째 열은 ABP(동맥혈압) 신호로 사용하였다. 세그먼트의 길이가 설정한 윈도우 길이(1000샘플)보다 짧은 경우에는 충분한 길이의 시계열 정보를 포함하지 못하므로 분석 대상에서 제외하였다. 또한 PPG 또는 ABP 신호의 표준편차가 0.01 미만인 구간은 거의 변화가 없는 신호(센서 오류, 평탄 구간, 포화 구간 등)로 간주하여 제거하였다. 이 과정을 통해 노이즈가 심하거나 정보량이 매우 낮은 세그먼트들을 사전 필터링하였다.

슬라이딩 윈도우 단계에서는 유효하다고 판단된 각 세그먼트에 대해 PPG와 ABP 신호를 동시에 처리하였다. 먼저 PPG 신호에는 0.5–8 Hz 대역을 통과시키는 3차 Butterworth 대역통과 필터를 적용하였다. 이는 호흡 관련 저주파 성분과 고주파 노이즈를 제거하고, 심박 관련 주파수 대역을 중심으로 신호를 정제하기 위함이다. 필터링에 실패하는 예외 상황(예: 신호가 비정상적인 형식인 경우)은 전체 파이프

라인의 중단을 방지하기 위해 해당 세그먼트만 건너뛰도록 처리하였다.

필터링된 PPG와 원본 ABP 신호에 대해, 1000샘플 길이의 윈도우를 500샘플 간격으로 슬라이딩하며 잘라내어 입력-라벨 쌍을 생성하였다. 각 윈도우마다 PPG와 ABP의 표준편차를 다시 확인하여, 둘 중 하나라도 변동성이 0.01 미만인 경우 해당 윈도우는 제외하였다. 이후 PPG 윈도우는 평균을 0으로, 표준편차를 1로 맞추는 구간 정규화를 수행하였으며, 극단값의 영향을 줄이기 위해 정규화된 값을 −5에서 5 사이로 클리핑하였다. 이렇게 전처리된 길이 1000의 PPG 배열은 모델의 입력으로 사용된다.

라벨은 동일한 구간의 ABP 신호를 활용하여 계산된 평균동맥압으로 정의하였다. 코드에서는 윈도우 내 ABP 값을 단순 평균하여 MAP 값으로 사용하였다. 즉, 각 8초 윈도우에 대해 하나의 실수형 MAP 값이 생성되며, 이는 해당 구간에서의 평균적인 혈압 상태를 대표하는 타깃으로 해석할 수 있다. 결과적으로, 전처리 파이프라인은 [윈도우 단위 PPG 시퀀스]-[해당 구간의 MAP 스칼라 값] 쌍을 다수 생성하며, 이들은 각각 X(입력 배열)와 Y(타깃 벡터)에 누적된다.

모든 파일에 대해 슬라이딩 윈도우 처리가 끝난 후, 개별 파일별로 생성된 X와 Y를 하나의 전체 데이터셋으로 병합하였다. 최종적으로 생성된 전체 데이터셋의 크기를 확인한 뒤, 이를 학습용, 검증용, 시험용 세트로 분할하였다. 분할은 scikit−learn의 train_test_split 함수를 사용하여 수행하였으며, 먼저 전체 데이터의 80%를 학습용으로, 나머지 20%를 임시 세트로 분할하였다. 이후 이 임시 세트에 대해 검증 세트 10%, 시험 세트 10%가 되도록 비율을 재조정하여 다시 분할함으로써, 최종적으로 학습:검증:시험의 비율을 8:1:1로 맞추었다. 모든 분할은 무작위 셔플을 수행하되, 재현성을 위해 랜덤 시드는 42로 고정하였다.

마지막으로, 분할된 각 데이터셋(X_train, X_val, X_test와 Y_train, Y_val, Y_test)은 h5py를 이용해 HDF5(.h5) 파일로 저장하였다. 각 파일에는 “X”와 “Y”라는 이름의 데이터셋을 생성하고, 데이터 타입은 float32로 설정하였으며, 저장 공간 효율을 위해 gzip 압축을 사용하였다. 결과적으로 “ppg_train.h5”, “ppg_val.h5”, “ppg_test.h5” 세 개의 파일이 생성되며, 이후 모델 학습 및 성능 평가 단계에서 바로 불러 사용할 수 있는 형태의 정제된 PPG-MAP 데이터셋이 구축되었다.

코드는 Appendix 1 에 첨부하였다.

3.3. 혈압 추정 모델 설계

3.4. 모델 경량화

3.5. 임베디드 시스템 탑재 (MCU)

본 연구에서는 PyTorch 환경에서 학습된 혈압 추정용 딥러닝 모델을 저전력 마이

크로컨트롤러인 nRF52840에서 실시간으로 실행하기 위해 임베디드 시스템 구현을 수행하였다. 이를 위해 모델의 경량화, 임베디드 환경에 적합한 포맷 변환, 펌웨어 개발 환경 구성, 추론 엔진 구축 및 기능 검증을 단계적으로 진행하였으며, 현재까지 MCU 기반 펌웨어의 빌드 및 정적 테스트가 완료된 상태이다. 이후 실제 장치 상에서의 실시간 구동 및 동작 안정성 검증을 수행할 예정이다.

PC 환경에서 학습된 PyTorch 모델은 부동소수점 연산과 대규모 메모리를 요구하기 때문에 MCU에서 직접 실행이 불가능하다. 이러한 제약을 해결하기 위해 모델을 ONNX(Open Neural Network Exchange) 형식으로 변환하여 연산 그래프를 고정한 뒤, onnx2tf 도구를 이용해 TensorFlow Lite(.tflite) 포맷으로 변환하였다. 변환 과정에서는 nRF52840의 메모리 구조를 고려하여 정수 양자화와 연산 구조 간소화를 적용하였으며, 최종적으로 모델을 C 헤더 파일(model_data.h) 형태의 배열 데이터로 변환하여 펌웨어에서 직접 참조할 수 있도록 구성하였다.

펌웨어 개발 환경은 Segger Embedded Studio(SES)와 nRF5 SDK v17.1.0을 기반으로 구축하였다. SDK에서 제공되는 기본 예제를 기반으로 프로젝트를 구성하되, 모델 추론 및 메모리 관리가 용이하도록 C++로 재구성하였다. TFLM 라이브러리 통합 과정에서는 PC 환경을 위한 불필요한 소스 파일이 빌드에 포함되면서 발생하는 의존성 문제를 해결하기 위해 Dynamic Folder와 Exclude Filter 기능을 활용하여 MCU 실행에 필요한 최소 소스만 포함되도록 정리하였다. 또한 중복된 SDK 경로로 인해 발생한 로그 관련 컴파일 오류는 SDK_ROOT 매크로 재정의 및 인클루드 경로 수정으로 해결하여 정상적인 빌드 환경을 확보하였다.

추론 엔진은 model_data.h를 로드하고, 필요한 연산자를 등록한 뒤 인터프리터를 초기화하는 방식으로 구현하였다. nRF52840의 제한된 RAM 용량을 고려하여 텐서 아레나 크기를 약 80KB로 설정하였으며, 이는 모델 추론 시 메모리 초과를 방지하면서도 실시간 실행이 가능하도록 조정한 값이다. 모델의 기능 검증을 위해 PC 환경에서 전처리된 테스트 데이터(ppg_test.h5) 중 일부 샘플을 추출하여 test_data.h 파일로 변환하고, 이를 펌웨어 내부에서 순차적으로 입력하도록 구성하였다. 각 샘플 입력에 대해 모델의 추론 결과와 실제 값의 차이를 평균절대오차(MAE)로 계산하여 로그로 출력하도록 설계하였고, 이를 통해 MCU 환경에서 정상적으로 추론이 수행됨을 확인하였다.

4. 실험 결과 및 성능 평가

4.1. 평가 환경

PC환경, MCU 환경, 평가 지표 등

4.2. 혈압 추정 정확도 평가

경량화 이전vs이후

4.3. MCU 구동 성능 및 검증

모델 준비 단계에서는 Python 환경에서 PPG 신호 기반 혈압 추정을 위한 딥러닝 모델을 학습하였다. 학습이 완료된 모델은 PyTorch 형식의 .pt 파일로 저장되었으며, 이후 임베디드 환경에서의 활용을 고려하여 ONNX(Open Neural Network Exchange) 형식으로 변환하였다. ONNX 모델은 프레임워크 독립적인 중간 표현 포맷으로, 이후 다양한 디바이스 및 경량화 도구와의 호환성을 확보하는 데 중요한 역할을 한다.

모델 경량화는 ONNX 모델을 TensorFlow Lite(TFLite) 형식으로 변환하는 과정에서 수행되었다. 변환에는 onnx2tf 툴체인을 사용하였으며, 이 과정에서 Int8 정수 양자화를 적용하여 모델의 연산을 부동소수점 기반에서 정수 연산 기반으로 변환하였다. 해당 양자화는 입력 텐서의 범위를 기반으로 수행되어, 모델의 크기와 연산 복잡도를 줄이는 동시에 성능 저하를 최소화하도록 설계되었다. 이를 통해 float32 기반 TFLite 모델과 Int8 양자화 TFLite 모델을 각각 생성하였으며, 두 모델은 이후 성능 비교 실험에 사용되었다.

생성된 TFLite 모델의 기본 동작 및 예측 성능을 검증하기 위해 실제 PPG 신호 데이터를 활용한 테스트를 수행하였다. 원본 데이터는 h5 형식으로 저장되어 있었으나, 임베디드 환경에서의 활용을 고려하여 PC 환경에서 데이터 로딩 및 전처리를 수행하였다. 이후 모델 입력 텐서 크기에 맞게 일부 구간을 슬라이싱하고 정규화 과정을 거쳐 CSV 형식으로 변환하였으며, 해당 데이터를 순차적으로 모델에 입력하여 예측 결과를 확인하였다.

성능 평가 결과, 회귀 모델의 Accuracy는 float32 모델과 Int8 양자화 모델 모두 약 0.64 수준으로 유사하게 나타났다. 이는 정수 양자화를 적용한 이후에도 모델의 전반적인 예측 성능이 유지되었음을 의미한다. 다만 Int8 양자화 모델의 경우, Mean Squared Error(MSE)와 Mean Absolute Error(MAE)가 float32 모델 대비 소폭 증가하는 경향을 보였는데, 이는 양자화 과정에서 발생하는 일반적인 정밀도 손실에 따른 결과로 분석된다. 그러나 해당 오차 증가는 실제 임베디드 응용에서 허용 가능한 범위로 판단되었다.

MCU 자원 사용 측면에서의 분석 결과, 경량화 모델의 장점이 더욱 뚜렷하게 확인되었다. nRF52840 MCU 기준으로 평가한 결과, Int8 양자화 모델의 RAM 사용량은 약 122 KB, Flash 사용량은 약 180 KB로 측정되었으며, 이는 해당 MCU가 제공하는 256 KB RAM 및 1 MB Flash 용량 대비 충분한 여유를 가지는 수준이다. 특히 float32 모델 대비 메모리 풋프린트가 감소함에 따라 실행 시 메모리 부담이 줄어들고, MCU 상에서의 안정적인 모델 로딩 및 추론 수행이 가능함을 확인하였다.

이와 같은 결과는 MCU 환경에서의 실시간 추론 성능과 저전력 운용을 동시에 고려할 때, 정수 양자화 기반 모델 경량화가 매우 효과적인 접근 방법임을 보여준다. 연산량 감소에 따른 추론 시간 단축과 전력 소모 감소 효과가 기대되며, 이는 배터

리 기반 엣지 디바이스에 적용하는 데 있어 중요한 이점으로 작용한다.

5. 결 론

5.1.

6. 참고문헌

7. Appendix

[1] preprocessing.py

```
import h5py
import numpy as np
from scipy.signal import butter, filtfilt
from tqdm import tqdm
from sklearn.model_selection import train_test_split

# --------- CONFIG ---------
FS = 125
WIN_SEC = 8
WIN = FS * WIN_SEC            # 1000 samples
STRIDE = WIN // 2             # 50% overlap
FILES = ["Part_1.mat", "Part_2.mat", "Part_3.mat", "Part_4.mat"]

TRAIN_RATIO = 0.8
VAL_RATIO = 0.1    # test = 0.1
# ---------------------------


def bandpass_ppg(x, fs=125, low=0.5, high=8, order=3):
    b, a = butter(order, [low/(fs/2), high/(fs/2)], btype='band')
    return filtfilt(b, a, x)


def load_signals(file):
    f = h5py.File(file, "r")
    part_key = [k for k in f.keys() if "Part" in k][0]
    part = f[part_key]
    refs = f["#refs#"]

    ppgs, abps = [], []

    for i in range(len(part)):
        try:
            idx = part[i][0]
            seg = np.array(refs[idx])
        except:
            continue

        if seg.ndim != 2 or seg.shape[1] < 2:
            continue

        ppg = seg[:, 0]
        abp = seg[:, 1]

        if len(ppg) < WIN:
            continue

        if np.std(ppg) < 0.01 or np.std(abp) < 0.01:
            continue

        ppgs.append(ppg.astype(np.float32))
        abps.append(abp.astype(np.float32))

    f.close()
    return ppgs, abps


def sliding(ppgs, abps):
    X, Y = [], []

    for ppg, abp in zip(ppgs, abps):
```

```
        try:
            ppg = bandpass_ppg(ppg)
        except:
            continue

        for s in range(0, len(ppg) - WIN, STRIDE):
            p = ppg[s:s+WIN]
            a = abp[s:s+WIN]

            if np.std(p) < 0.01 or np.std(a) < 0.01:
                continue

            p = (p - np.mean(p)) / (np.std(p) + 1e-6)
            p = np.clip(p, -5, 5)

            map_val = np.mean(a)

            X.append(p)
            Y.append(map_val)

    return np.array(X, dtype=np.float32), np.array(Y, dtype=np.float32)


# ================ MAIN ================
print("Start UCI PPG to MAP preprocessing")

allX, allY = [], []

for f in FILES:
    print("\nProcessing", f, "...")
    ppgs, abps = load_signals(f)
    X, Y = sliding(ppgs, abps)

    print("Windows extracted:", len(X))

    if len(X) > 0:
        allX.append(X)
        allY.append(Y)

if not allX:
    raise RuntimeError("Error: No windows created. Check data.")

X = np.concatenate(allX)
Y = np.concatenate(allY)

print("\nFinal dataset size:", X.shape)


# --------- Split Train / Val / Test ---------
X_train, X_tmp, Y_train, Y_tmp = train_test_split(
    X, Y, test_size=(1 - TRAIN_RATIO), shuffle=True, random_state=42
)

val_ratio_adjusted = VAL_RATIO / (1 - TRAIN_RATIO)
X_val, X_test, Y_val, Y_test = train_test_split(
    X_tmp, Y_tmp, test_size=(1 - val_ratio_adjusted), shuffle=True, random_state=42
)

print("\nSplit result:")
print("Train:", X_train.shape)
print("Val   :", X_val.shape)
print("Test :", X_test.shape)


# -------- Save ---------
def save_h5(filename, X, Y):
    with h5py.File(filename, "w") as hf:
        hf.create_dataset("X", data=X, dtype="float32", compression="gzip")
        hf.create_dataset("Y", data=Y, dtype="float32", compression="gzip")
    print("Saved:", filename)


save_h5("ppg_train.h5", X_train, Y_train)
save_h5("ppg_val.h5",    X_val,    Y_val)
save_h5("ppg_test.h5",   X_test,   Y_test)

print("\nDone. Train/Validation/Test datasets created.")
```