

캡스톤디자인1 과제 보고서

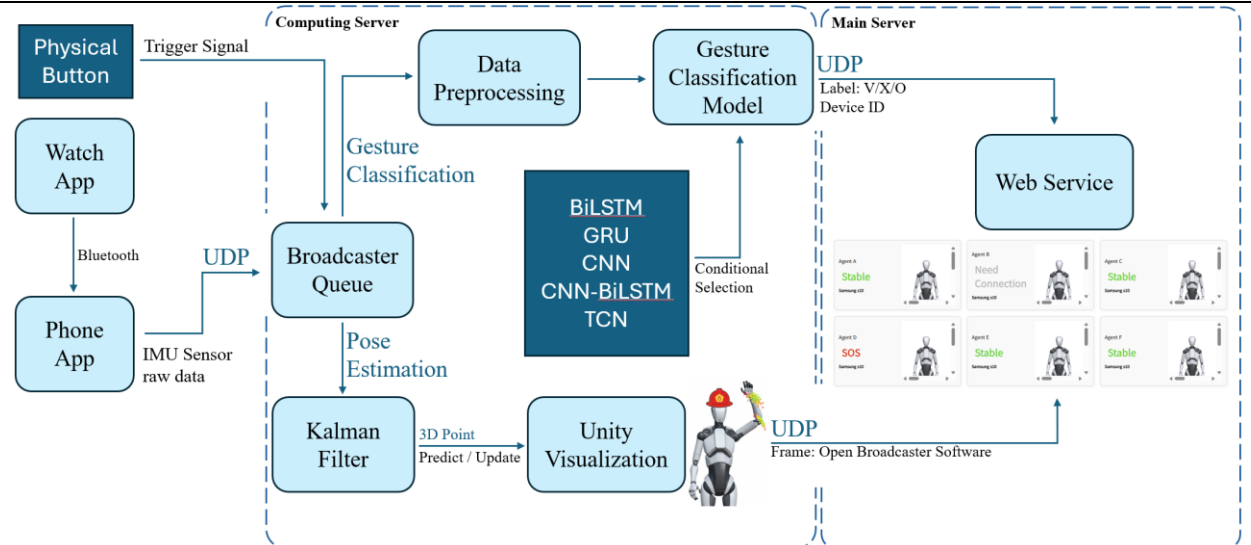
1. 과제 이름	스마트워치를 활용한 실시간 자세 추정 기반 소방, 경찰 수신호 통신 시스템 "Vox"		
2. 팀 구성원	60201969 이유현(팀장)	60201956 유대용	60201941 백승호
3. 과제 요약	<p>소방, 경찰 분야의 PS-LTE, TETRA 무전기는 다수 요원의 동시 교신을 제약하는 Half-Duplex 구조와 긴급 상황 시 조작이 어려운 물리적 한계를 지닙니다. 본 과제는 이러한 통신 제약을 해소하기 위해, 스마트워치 IMU(관성 측정 장치) 센서로 사용자의 손 제스처를 인식하여 중앙에 전송하는 시스템 'Vox'를 제안합니다.</p> <p>Vox는 UDP로 스트리밍되는 6축 관성 데이터에 대해 칼만 필터^[3]를 이용한 실시간 자세 추정과 CNN-BiLSTM 기반의 제스처 분류를 병렬 수행합니다. 특히, 제스처 구간은 전용 장갑의 물리 버튼으로 명시적으로 제어하여, 재난 현장에서의 오작동을 원천적으로 차단하고 시스템 신뢰성을 확보했습니다.</p> <p>성능 검증을 위해 6DMG 공개 데이터셋으로 사전 학습한 모델을 자체 수집 데이터(레이블당 60개 학습/20개 테스트)로 파인튜닝 하였습니다. 5-fold 교차 검증을 거친 최종 CNN-BiLSTM 모델은 외부 테스트 데이터에서 정확도(Accuracy) 1.0000, Macro F1-Score 1.0000을 달성했으며, 추론 지연 시간은 0.415ms로 실시간 시스템 요구 조건(50ms 이내)을 만족시켰습니다.</p> <p>제안 시스템은 추가 장비 투자(CAPEX) 없이 기존 스마트워치를 활용하며 현행 PS-LTE 망과도 병행 운용이 가능합니다. 결론적으로 Vox는 음성 교신이 어렵거나 시야가 제한된 재난 현장에서, 요원의 안전을 확보하고 지휘 통제의 효율성을 극대화하는 실용적 통신 보조 수단으로서의 가능성을 입증했습니다.</p>		
4. 과제 선정에 대한 배경	<p>현재 소방 및 경찰 통신 시스템에서 사용하고 있는 미국의 TETRA 방식과 대한민국의 PS-LTE 방식은 Half-Duplex 통신 방법을 채택하고 있어, 동시다발적인 통신이 불가능하고, 통신망의 한 명만 발언권을 갖는다는 단점이 있습니다. 이를 개선하기 위해 구현된 우선 순위 통신 기능인 "비상 통화"는 지휘관만 사용 가능하며, 동시 다발적인 비상 상황에서는 발언권 확보에 제한이 있습니다.</p> <p>무전기 사용은 혼란한 상황 속에서 까다로운 작은 버튼들의 조작과 한 손의 사용 제약을 강요합니다. 이는 작업 효율성 저하와 긴급 상황 전파 지연의 원인이 됩니다.^{[1][2]} 또한, 음성 정보만으로는 상황 인지가 제한적이며, 중앙에서 모든 상황과 세부 현장을 빠르고 정확하게 파악하기 어렵게 만듭니다.^{[1][2]}</p> <p>대한민국에서는 2022년 06월부터 재난안전통신망을 시행하여 소방, 경찰, 지자체 무선 통신망 하나로 통합하였습니다. 통합으로 인해 중앙에서의 명령 하달 및 상황 전파는 단순화하였으나, 개별 소방대원들이 자신의 구체적 상황이나 위험 상태, 요청 사항을 역으로 중앙에 즉각 공유하기에는 절차가 복잡해져 실시간 대응이 어려워졌습니다. 현장 인력의 상황을 실시간으로 파악하는 것은 작전 계</p>		

	<p>획 수립 및 지시에 있어서 필수적입니다.</p> <p>이에, 본 과제에서는 웨어러블 IMU 센서와 딥러닝 기반 자세 추정 기술을 접목하여, 현장 요원의 수신호를 실시간으로 탐지 및 전파하는 통신 시스템을 개발하고자 합니다.</p>
5. 사전 기술 및 시장을 포함한 배경	<p>딥러닝 및 AIOT(AI+IoT) 기술이 확산되면서, 개인의 자세와 동작 인식 기술이 다양한 분야(스포츠, 재활, 군사 훈련, XR, 로봇 공학 등)에서 활발히 연구되고 있습니다. 특히, IMU 센서를 활용한 자세 추정은 카메라 기반 추정의 사각지대를 보완하고, 높은 정확도를 확보할 수 있어 주목받고 있습니다.^{[3][4][5]} 대표적인 선행 연구로는 IMU 기반 스마트워치를 활용한 로봇 제어 연구^[3], 조끼 내장 IMU 센서를 이용한 농구 동작 인식^[4], IMU 센서를 이용한 보행 모드 분류 연구^[5] 등이 존재합니다. 정확하고 빠른 자세 추정을 위해서는 고비용, 고성능의 IMU 센서 및 프로세서가 요구됩니다.</p> <p>시장 배경 측면에서, 소방 분야의 R&D 및 예산이 매우 저조합니다. 소방 R&D 예산은 전체 R&D 사업비의 10%도 되지 않으며, 소방청의 한 해 예산 규모는 전체 18개 청 중 14위입니다.^[6] 고가의 IMU 센서를 활용한 소방 통신 시스템은 도입하기에 현실적인 적용 가능성이 떨어집니다. 소방 및 경찰 분야는 신기술 도입에 있어서 고비용 장비보다 신뢰성과 현실 적용 가능성이 더 중요한 요소입니다.</p> <p>스마트워치 등 웨어러블 디바이스와 스마트폰의 보급률이 증가^{[7][8]}하면서, 일반 사용자도 고성능 IMU 센서(가속도계, 자이로스코프, 지자기계 등)와 프로세서에 쉽게 접근할 수 있게 되었습니다. 본 과제에서 제안하는 방식은 별도 고가 장비 없이 기존의 스마트워치와 스마트폰을 활용하기 때문에, 현실 적용 가능성이 높습니다.</p>
6. 관련 연구	<p>본 과제는 안정적인 자세 추정, 제스처 데이터셋 확보, 그리고 고성능 분류 모델 구현을 위해 다음의 선행 연구들을 기반으로 설계 및 확장되었습니다.</p> <p>1. WearMoCap^[3]</p> <p>WearMoCap은 스마트워치와 스마트폰의 IMU 데이터를 이용해 실시간으로 인체 자세를 추정하는 연구입니다. 이 연구에서 제안한 IMU 데이터 스트리밍 애플리케이션, 칼만 필터 기반의 추정 알고리즘, Unity 3D 시각화 모듈은 본 시스템의 자세 추정 파이프라인의 기반 아키텍처로 채택되었습니다. 이를 통해 복잡한 자세 추정 환경을 직접 개발하는 대신, 안정성이 검증된 프레임워크 위에서 제스처 분류 기능 개발에 집중할 수 있었습니다.</p> <p>2. 6DMG: A New 6D Motion Gesture Database^[9]</p> <p>6DMG는 닌텐도 Wiimote의 IMU와 광학 트래킹을 결합하여 수집된 공개 제스처 데이터베이스로, 총 20개 제스처, 5,600개의 샘플로 구성되어 있습니다. 본 연구에서는 이 데이터셋을 분류 모델의 오프라인 사전 학습(pre-training)에 활용하였습니다. 이를 통해 모델이 특정 하드웨어나 사용자에 과적합되지 않고, 다양한 제스처 패턴에 대한 일반화 성능을 초기에 확보하는 기반으로 삼았습니다.</p> <p>3. HandGesture Recognition Using Single Patchable Six-Axis Inertial Measurement Unit via</p>

Recurrent Neural Networks^[10]

Valarezo Añazco 등의 연구는 단일 6축 IMU 센서와 RNN(BiLSTM/GRU)을 이용해 V, X, O 손 제스처를 분류하는 방법을 다룹니다. 본 과제는 이 논문에서 제안한 핵심 전처리 기법과 모델 하이퍼파라미터를 직접 참조하여 초기 모델을 설계했습니다. 구체적으로, 0.2Hz Butterworth 고역 통과 필터 (HPF)를 적용한 중력 성분 제거 방식과 BiLSTM/GRU 모델의 주요 하이퍼파라미터를 초기 실험의 베이스라인(Baseline)으로 설정하였습니다. 이를 통해 검증된 방법론에서 연구를 시작하여, 이후 모델 구조를 개선하고 최적화하는 데 집중할 수 있었습니다.

7. 제안하는 방법



본 과제는 소방, 경찰 현장의 실시간 수신호 통신을 위해, 데이터 스트리밍부터 분류, 시각화에 이르는 다음과 같은 병렬 처리 파이프라인을 제안합니다.

1. 비동기 IMU 데이터 스트리밍 (Asynchronous IMU Data Streaming)

데이터: 스마트워치와 스마트폰에 내장된 6축 IMU 센서(가속도, 자이로스코프)를 활용합니다.

전송 방식: 수집된 Raw 데이터를 스마트폰에서 단일 스트림으로 병합한 뒤, UDP(User Datagram Protocol) 패킷을 통해 연산 서버로 실시간 브로드캐스팅합니다.

2. 병렬 처리 파이프라인 (Parallel Processing Pipeline)

구조: 연산 서버는 수신된 IMU 스트림을 브로드캐스터 큐에 적재한 뒤, 이를 두 경로로 복제하여 병목 없이 동시 처리합니다. 경로 A는 실시간 자세 추정, 경로 B는 이벤트-기반 제스처 분류를 담당합니다.

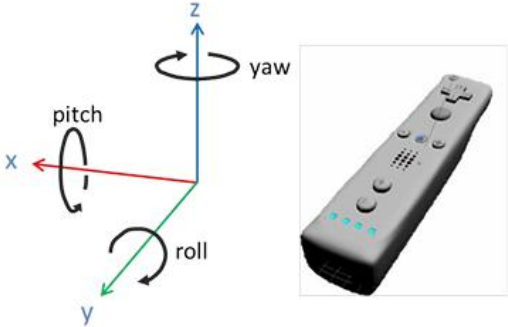
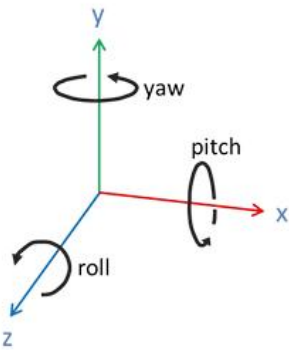
경로 A: 연속적 자세 추정 (Continuous Pose Estimation)

모든 IMU 데이터는 선행 연구^[3]의 베이지-앙상블 칼만 필터(Bayesian Ensemble Kalman Filter) 모델로 전달됩니다. 이를 통해 상시(Always-on) 3D 자세 추적을 통해 요원의 지속적인 움직임과 상태를 모니터링합니다.

경로 B: 이벤트 기반 제스처 분류 (Event-Based Gesture Classification)

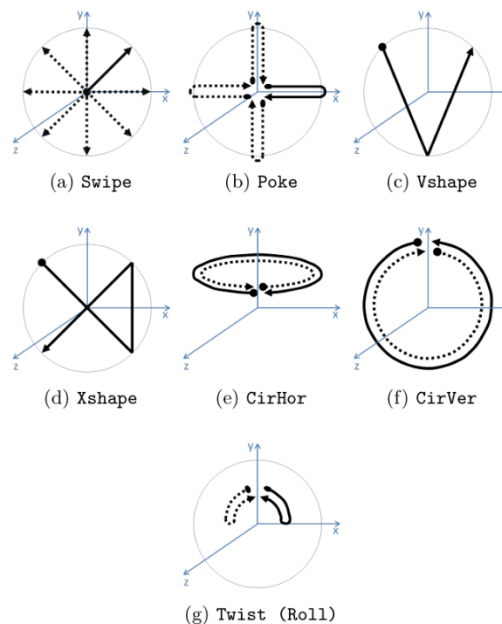
이벤트 기반(Event-driven) 제스처 분류를 통해 명령 또는 상태 보고와 같은 핵심 정보만 정확하게 포착합니다. 데이터가 분류기(CNN+BiLSTM)로 전달되어, 'V, X, O' 세 가지 제스처 중 하나로 분류됩니다.

	<p>다. 이 분리된 구조는 제스처 분류 작업이 자세 추정의 실시간성에 영향을 미치지 않도록 보장합니다.</p> <p>3. 명시적 수신호 구간 제어 (Explicit Hand-Signal Segmentation)</p> <p>제어 방식: 자동 탐지 방식의 오탐(false positive) 위험을 원천 차단하기 위해, 전용 장갑 등에 내장된 단일 물리 버튼으로 수신호의 시작과 끝을 사용자가 명시적으로 지정합니다.</p> <p>신뢰성 확보: 버튼을 누르는 동안의 데이터만 분류기에 투입하므로, 유사 동작을 수신호로 오인하는 문제를 근본적으로 해결하여 시스템의 신뢰성을 극대화합니다.</p> <p>4. 통합 시각화 및 원격 모니터링 (Integrated Visualization & Remote Monitoring)</p> <p>시각화: 추정된 3D 자세와 분류된 제스처 레이블은 연산 서버의 Unity 환경에서 실시간 렌더링됩니다.</p> <p>스트리밍 및 모니터링: 렌더링된 3D 뷰는 OBS와 Nginx RTMP/HLS 모듈을 통해 준실시간(Near Real-time) 스트리밍되며, 중앙 관제 지휘관은 수초의 지연을 갖는 3D 자세 영상과 실시간으로 갱신되는 제스처 로그를 통해 현장 요원을 원격 모니터링할 수 있습니다.</p> <p>이러한 통합 아키텍처는 기존 자세 추정 모듈의 무결성을 보장하면서 분류 및 모니터링 기능을 안정적으로 확장하여, 시스템 전체의 신뢰성, 실시간성, 확장성을 효과적으로 확보합니다.</p>
8. 목표	<p>본 과제는 상용 웨어러블 기기를 활용하여 소방, 경찰 현장에 적용 가능한 실용적인 통신 시스템을 구축하는 것을 목표로 하며, 세부 목표는 다음과 같습니다.</p> <ol style="list-style-type: none">1. 다양한 노이즈 환경에서도 강건하게 동작하는 고정밀 제스처 분류 모델 개발2. 현장 상황 판단에 즉각 반영될 수 있는 실시간 처리 성능 확보3. 오탐, 미탐 없는 안정적인 수신호 전송을 위한 높은 시스템 신뢰성 구축4. 다수 요원을 동시에 관제할 수 있는 원격 모니터링 시스템 구현5. 추가적인 고가 장비 없이 운용 가능한 시스템 효율성 및 실용성 확보
9. 구현 사항	<p>1. 실시간 IMU 데이터 스트리밍 (Real-time IMU Data Streaming)^[3]</p> <p>사용자가 착용한 스마트워치와 스마트폰의 IMU 센서 데이터(가속도, 자이로스코프)를 통합하여, 무선 통신(UDP)을 통해 연산 서버로 실시간 전송합니다.</p> <p>2. 비동기 병렬 처리 파이프라인 (Asynchronous Parallel Processing Pipeline)</p> <p>서버에 수신된 단일 데이터 스트림을 두 개의 독립된 경로로 즉시 복제하여, 상시적인 3D 자세 추정과 이벤트 기반의 수신호 분류가 서로의 연산에 전혀 영향을 주지 않고 병렬로 동시에 수행되도록 파이프라인을 구축했습니다.</p> <p>3. 고신뢰성 수신호 구간 제어 (High-Reliability Signal Segmentation Control)</p> <p>재난 현장과 같은 혼란스러운 환경에서의 오작동을 원천 차단하기 위해, 사용자가 물리 버튼을 누르는 동안의 데이터만을 수신호로 간주하는 명시적 제어 메커니즘을 적용했습니다. 이는 유사한 동작을 제스처로 오인하는 문제를 근본적으로 해결하여 시스템의 전체 신뢰도를 확보합니다.</p>

	<p>4. 고성능 딥러닝 분류 모델 탑재 (High-Performance Deep Learning Classification Model) 공개 데이터셋으로 사전 학습하고, 자체 수집한 데이터로 미세 조정된 딥러닝 모델(CNN+BiLSTM)을 시스템에 탑재했습니다. 이 모델은 버튼으로 입력된 제스처 구간의 IMU 데이터를 분석하여 'V, X, O' 등 정의된 수신호로 즉시 분류합니다.</p> <p>5. 통합 원격 관제 시스템 (Integrated Remote Monitoring System) 분석된 모든 정보는 중앙 관제용 웹 대시보드에 통합 시각화된다. 지휘관은 이 대시보드를 통해 다수의 현장 요원의 실시간 제스처 로그와 3D 자세 움직임을 동시에 원격으로 모니터링하며 상황을 통제할 수 있습니다</p>
10. 하드웨어 환경	<p>본 과제의 개발 환경에서 사용한 하드웨어 환경은 다음과 같습니다.</p> <ol style="list-style-type: none"> 1. SmartPhone: Galaxy Z Fold4 (SM-F936) 2. SmartWatch: Galaxy Watch 5 (SM-R900) 3. Local GPU Computing Server: Razer Blade 18 (GeForce RTX™ 4090) 4. Central Aggregator Server: AWS EC2
11. 상세 개발 내용	<p>1. 데이터</p> <p>1-1. 오프라인 데이터 수집</p> <p>제스처 분류기의 일반화 성능을 높이기 위해, 먼저 6DMG(6D Motion Gesture Database)^[9] 공개 데이터셋을 활용한 오프라인 학습을 수행한 뒤, 전이학습(transfer learning)을 통해 본 시스템의 하드웨어 환경에 맞게 추가 튜닝을 진행하였습니다.</p> <p>오프라인 학습용 데이터는 6DMG(6D Motion Gesture Database) 공개 데이터 세트입니다. 해당 동작 제스처 데이터셋은 아래 그림과 같이 Wiimote(닌텐도 Wii 리모컨)에 내장된 IMU 센서를 기반으로 수집되었습니다.</p> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> <p>Wiimote Coordinates</p>  </div> <div style="text-align: center;"> <p>Ogre Coordinates</p>  </div> </div> <p>해당 데이터셋은 3축 가속도(acceleration)와 3축 각속도(angular speed), 바이어스를 광학(optical) 트래킹과 관성(inertial) 센서를 결합하여 수집하였습니다. 광학 트래킹은 적외선(Infrared) LED나 반사 마커를 카메라가 60Hz로 촬영해 3차원 위치를 계산하였고, 관성 센서는 Wiimote(닌텐도 Wii 리모</p>

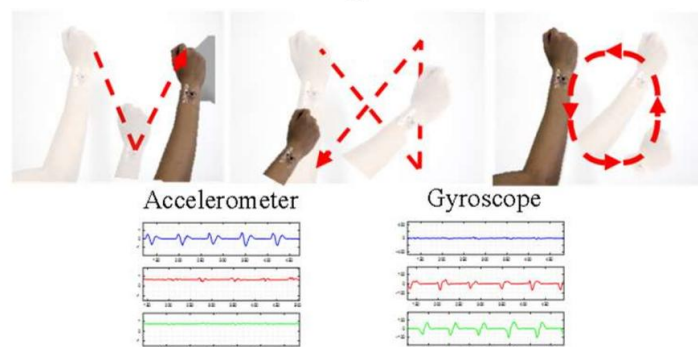
컨)에 내장된 MEMS 가속도계(accelerometer)와 자이로스코프(gyroscope)를 활용해 가속도(acceleration)와 각속도(angular speed)를 100Hz로 측정하였습니다. Wiimote는 자력계(magnetometer)가 없어 Yaw(방위각) 드리프트가 발생할 수 있으므로, Madgwick 필터를 이용해 오차를 보정하고 주기적인 하드(정지 상태에서 편향, 노이즈 직접 측정)/소프트(누적 드리프트 계산하여 보정) 캘리브레이션을 통해 드리프트 문제를 완화하였습니다.

이처럼 6DMG 데이터셋은 상용 스마트워치와는 다른 하드웨어(Wiimote) 및 환경에서 수집되었으므로, 이를 통해 학습한 모델은 일반적인 제스처 패턴을 인식하는 사전 학습(pre-training) 용도로 적합합니다. 이후 본 과제의 실제 하드웨어(스마트워치)에 대한 정밀한 성능을 확보하기 위해 온라인 파인튜닝 과정이 필수적입니다.



총 20가지 제스처 데이터가 포함되어 있으며, 오른손 21명과 왼손 7명의 인원으로부터 제스처당 10회씩 데이터를 수집하여 총 5615개 샘플을 제공합니다.

IMU Signals



MATLAB 기반의 데이터셋을 CSV로 변환하였고, 20가지 제스처 중, V, X, O를 그리는 제스처^[10]만 선별하여 사용하였습니다. 이들 제스처는 수행 시간이 유사하면서도 다른 동작들보다 길이가 충분히 분별력이 높다고 판단됩니다.

1-2. 오프라인 데이터 정제

실시간 분류 환경을 고려하여 원본 데이터를 50 Hz로 다운샘플링 했습니다. 이후, 중력 성분을 제거하기 위해 컷오프 주파수 0.2 Hz, 4차 Butterworth 고역 필터(HPF)를 적용했습니다. 정제된 데이터는 윈도우 단위로 최소-최대 스케일링(min-max scaling)을 통해 $[-1, +1]$ 범위로 정규화하였습니다. 각 제스처 시퀀스 길이를 최대 3초(150프레임) 길이로 고정하고, 짧은 시퀀스는 끝에 제로 패딩(zero padding)을, 긴 시퀀스는 앞쪽 150프레임까지만 절삭(truncate)하여 최종적으로 크기 6×150 의 행렬을 만들어 학습 데이터셋을 구성했습니다.

1-3. 온라인 데이터 수집

본 과제의 실제 하드웨어 환경(스마트워치 및 스마트폰)을 활용하여 사용자 제스처 데이터를 직접 수집했습니다. 각 제스처 레이블별로 60개씩 수집하였으며, 다양한 사용자의 자세, 속도, 시간 분포를 반영하여 일반화 성능을 높일 수 있도록 구성했습니다.

1-4. 온라인 데이터 정제



실제 스마트워치의 축과 학습에 사용한 6DMG의 축은 서로 다릅니다. 수집한 Raw CSV의 축을 6DMG에서 정의한 축과 동일하게 될 수 있도록 축 변환(mapping)의 과정을 거칩니다.

이후, 오프라인 데이터셋과 동일하게 50Hz 다운샘플링, 중력 성분 제거(고역 필터링), 윈도우별 정규화, 패딩의 과정을 거쳐 크기 6×150 의 행렬을 만들어 학습 데이터셋을 구성했습니다.

2. 모델 구조

본 과제에서는 총 5가지 모델을 실험하였습니다. 제스처 분류 성능을 다각도로 검증하기 위해, 선행 연구^[10] 기반으로 채택한 순환 신경망 계열(BiLSTM, GRU), 자체 제안하는 컨볼루션 신경망 계열(SimpleCNN, TCN), 그리고 이들을 결합한 하이브리드 모델(CNN+BiLSTM) 등 총 5개의 아키텍처를 설계 및 비교하였습니다.

2-1. BiLSTM (양방향 LSTM)

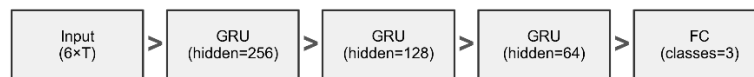
BiLSTM



BiLSTM은 순방향과 역방향 LSTM을 병렬로 결합하여 시퀀스의 과거 정보뿐만 아니라 미래 정보까지 함께 학습할 수 있는 구조입니다. 제스처 분류에서는 동작의 시작점과 종료점, 그리고 시퀀스 중간의 특징이 모두 분류 성능에 영향을 미치므로, BiLSTM은 일반 LSTM이 놓칠 수 있는 '후반부 맥락'을 반영하여 보다 안정적인 특징 벡터를 생성합니다. 또한 Packed Sequence 기법을 통해 실제 동작 구간만 네트워크에 투입하고 제로 패딩 구간을 배제할 수 있어, 입력 길이가 불규칙한 경우에도 패딩에 의한 성능 저하 없이 유연하게 대응할 수 있습니다.

2-2. GRU (Gated Recurrent Unit)

GRU



GRU는 LSTM의 복잡한 게이트 구조를 업데이트, 리셋 게이트 두 개로 단순화한 순환 셀로, 파라미터 수와 연산량을 현저히 줄이면서도 장기 의존성(long-range dependency)을 효과적으로 학습할 수 있습니다. 제스처 분류에서는 짧은 휴지점이 포함된 비정형 동작이나 빠르게 연속되는 동작의 시간적 맥락을 파악하는 것이 중요한데, GRU는 이들 기능을 대등한 수준으로 수행하면서 LSTM 대비 학습 및 추론 속도가 빠르고 메모리 점유율이 낮아 모바일, 임베디드 환경에 적합합니다.

2-3. CNN (1D-Convolutional Neural Network)

CNN

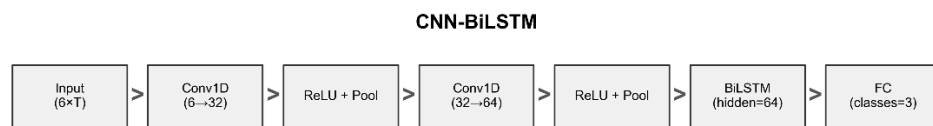


1D-CNN은 시계열 데이터를 일정한 윈도우 크기씩 지역적으로 살펴보면서 로컬 패턴(local temporal pattern)을 추출하는 구조입니다. 제스처 분류에서는 가속도가 급격히 변화하는 구간이나 회전 속도가 특정 패턴으로 바뀌는 구간 등이 중요한데, 1D-CNN은 커널을 통해 이런 국소적인 특징을 바로 감지할 수 있습니다. 게다가 CNN은 시퀀스를 고정 길이(150프레임 등)로 맞춰놓고, 그 뒤에 남은 제

로 패딩 구간까지 그대로 입력해도 큰 문제가 없습니다. 이유는 패딩된 구간이 CNN의 합성곱+풀링 연산에 상대적으로 큰 영향을 미치지 않도록 구조를 설계하기 때문입니다. 즉, 입력 시퀀스 뒤쪽에 추가된 0 값들은 커널이 지나면서 일정 수준 걸러질 수 있어, 오히려 패딩을 제거하거나 복잡한 마스킹을 할 필요가 없습니다.

CNN은 계층을 몇 겹만 쌓아도 시퀀스 길이를 크게 줄이면서, 학습해야 할 특징을 효과적으로 요약할 수 있습니다. 파라미터 수가 많지 않고, 행렬 곱을 통한 병렬 연산이 뛰어나 실시간 추론 속도가 빠릅니다.

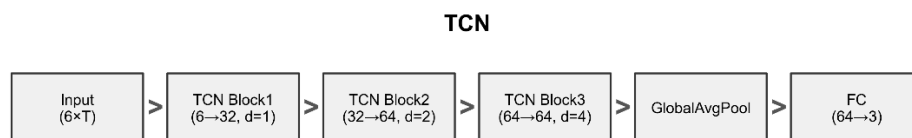
2-4. CNN + BiLSTM (하이브리드 모델)



CNNBiLSTM 모델은 먼저 CNN으로 시퀀스 전체를 국소적 필터로 훑어가며 로컬 패턴을 추출한 뒤, 그 결과를 BiLSTM에 넣어 시계열 전체 맥락을 종합하는 구조입니다. 제스처를 분류할 때는 회전 중에 유지되는 특정 가속도 패턴처럼 단기적 특징이 있고, 동시에 동작 순서 자체가 중요한 경우가 많은데, CNN은 앞쪽 단계에서 노이즈나 패딩을 어느 정도 걸러주고, BiLSTM은 그 위에서 시퀀스 전반의 시간적 흐름을 학습합니다.

이렇게 두 단계로 나누면, 단일 BiLSTM만 사용할 때보다 입력 차원이 먼저 축소되므로 LSTM 연산량이 줄어들어 전체 연산 효율이 개선됩니다. 또한, CNN이 푸는 문제(어느 구간에 특정 진동 패턴이 있는가?)와 LSTM이 푸는 문제(진동 패턴들이 시간 축 위에서 어떻게 연결되어 의미를 이루는가?)를 분리해주므로 모델이 더 명확한 방식으로 학습할 수 있습니다. 오프라인 환경에서 고정 길이 시퀀스가 보장되고, "로컬+글로벌" 양쪽 맥락을 모두 고려해야 할 때 CNN-BiLSTM이 제스처 분류 성능을 높이는 데 강력한 도구가 됩니다.

2-5. TCN (Temporal Convolutional Network)



TCN은 전형적인 RNN 방식이 아닌, 완전한 컨볼루션만으로 시계열 전체 의존성을 학습하는 구조입니다. 핵심 아이디어는 1D 합성곱 레이어 각 단계마다 확장 필터(dilation)를 주어 지금 이 시점의 출력이 과거의 얼마나 먼 시점까지 영향을 받는지를 조절하는 것입니다. 제스처 분류에서 동작의 시작과 끝 사이가 길게 떨어져 있어도, 두 지점 간 관계가 중요하다면, TCN은 은닉 상태를 순차적으로

업데이트하는 대신 한 번의 전방-후방 패딩 없는 컨볼루션으로 모든 시점의 관계를 포착합니다. 본 과제의 TCN은 채널 수를 비교적 작게 유지하고, 커널 크기를 3으로 고정한 뒤, dilation만 1, 2, 4 처럼 단계적으로 키워줍니다. 이렇게 하면 “짧은 시간축 패턴”과 “긴 시간축 패턴”을 함께 학습하면서, 각 레이어를 한 번만 지나가면 되기 때문에 RNN 대비 병렬화 효율이 뛰어납니다. 또한, 순환 구조가 없으므로 길이가 고정된 시퀀스(150프레임)를 빠르게 처리할 수 있어, 실시간 분류 시스템에서 GPU 병렬 자원을 최대한 활용하고자 할 때 적합합니다.

3. 모델 학습

본 과제에서는 모든 모델이 제로 패딩된 구간을 포함한 시퀀스를 학습하도록 설계하였습니다. 특히 BiLSTM과 GRU는 패딩 구간을 모델에 사전 제공하여 오로지 유효한 수신호 데이터만 학습하도록 하였습니다. 과적합을 방지하기 위해 StratifiedKFold 기법을 적용하여 5개 fold로 성능을 우선 평가한 뒤, 최적의 하이퍼파라미터를 찾은 후에는 전체 학습셋을 사용하여 최종 학습을 수행하였습니다. 각 모델별로 실험을 통해 가장 성능이 우수한 하이퍼파라미터를 설정하였습니다.

3-1. 오프라인 학습

오프라인 학습은 공개 데이터(6DMG)로만 이루어지며, 아래 표에 각 모델의 옵티마이저, 학습률, 배치 크기, 반복 횟수를 정리하였습니다.

Model	Optimizer	Learning Rate (lr)	Batch Size	Iterations
BiLSTM	SGD (momentum=0.9)	3e-2	100	200
GRU	Adam	3e-4	64	100
CNN	Adam	1e-2	100	200
CNN + BiLSTM	Adam	3e-4	64	100
TCN	Adam	1e-3	64	150

3-2. 온라인 학습

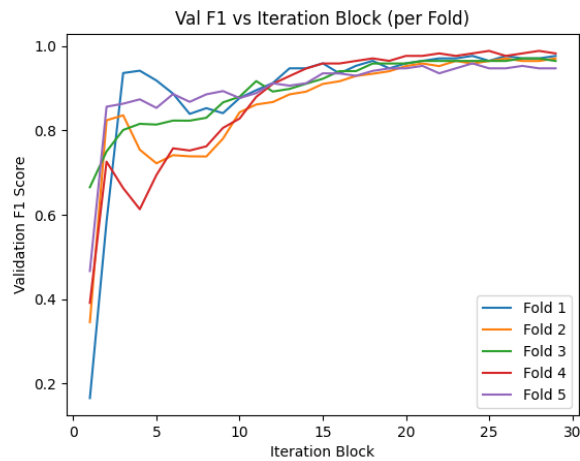
온라인 학습은 직접 녹화한 실제 스마트워치, 스마트폰 데이터로 진행하였으며, 사전 학습된 가중치는 모두 동결(Freeze)한 상태에서 모델별로 지정한 레이어만 풀어서 파인튜닝 하였습니다. 교차 검증 기법은 오프라인 학습과 동일하게 5 fold를 적용하였습니다.

Model	Unfrozen Layers	Learning Rate (lr)	Batch Size	Iterations
BiLSTM	model.fc	3e-4	32	150
GRU	model.gru2 model.gru3 model.fc	3e-4	32	200
CNN	model.fc1 model.fc2	3e-4	32	150
CNN + BiLSTM	model.fc	3e-4	32	150
TCN	model.fc	1e-3	32	150

4. 모델 학습 결과 분석

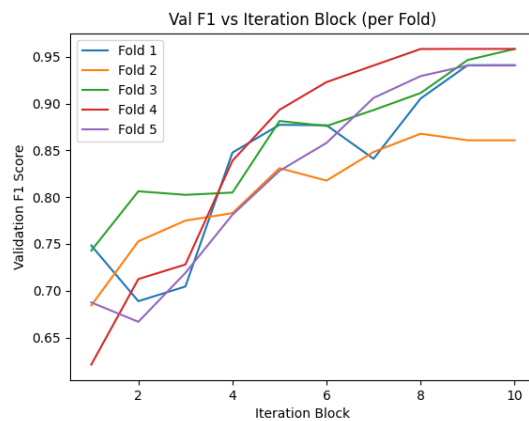
4-1. 오프라인 학습 결과

4-1-1. BiLSTM



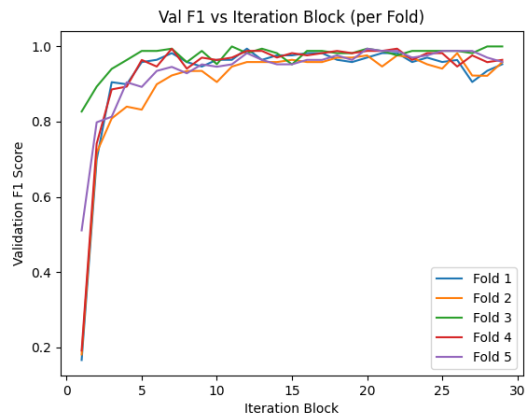
Train	Accuracy ± Std Dev	Loss ± Std Dev
CV Result	0.9679±0.0123	0.0941±0.0228
Final model	0.9774	0.0719

4-1-2. GRU



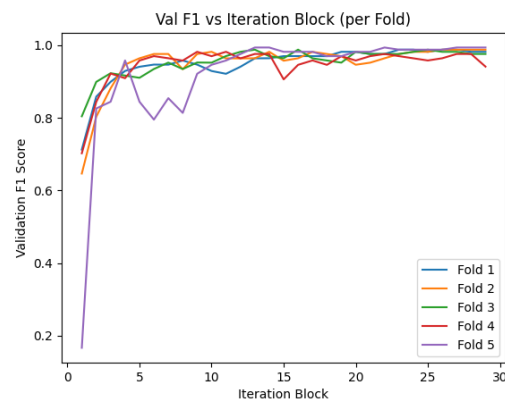
Train	accuracy ± Std Dev	Loss ± Std Dev
CV Result	0.9322±0.0355	0.2276±0.0696
Final model	0.9168	0.2501

4-1-3. CNN



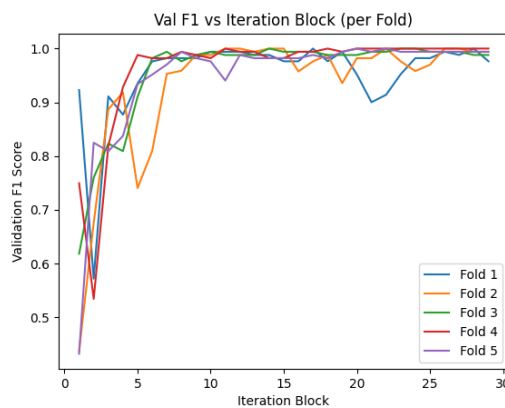
Train	accuracy \pm Std Dev	Loss \pm Std Dev
CV Result	0.9667 \pm 0.0170	0.1302 \pm 0.0707
Final model	0.9881	0.0383

4-1-4. CNN + BiLSTM



Train	accuracy \pm Std Dev	Loss \pm Std Dev
CV Result	0.9762 \pm 0.0188	0.0720 \pm 0.0431
Final model	0.9964	0.0118

4-1-5. TCN



Train	accuracy \pm Std Dev	Loss \pm Std Dev
CV Result	0.9905 \pm 0.0080	0.0282 \pm 0.0374
Final model	1.0000	0.0018

4-1-6. 오프라인 학습 결과 분석

아래 표는 6DMG 데이터셋을 이용한 5-fold 교차 검증(CV) 결과와, 최종 모델 학습 결과(Final Model)를 요약한 것입니다. 최종 모델은 CV에 사용한 데이터를 모두 학습에 투입했기 때문에, Final Model의 성능 지표는 모델의 잠재적 성능 상한선을 보여주는 참고 자료이며, 실제 일반화 성능은 CV 결과를 중심으로 해석해야 합니다.

Model	CV Accuracy	CV Loss	Final Accuracy	Final Loss
BiLSTM	0.9679 \pm 0.0123	0.0941 \pm 0.0228	0.9774	0.0719
GRU	0.9322 \pm 0.0355	0.2276 \pm 0.0696	0.9168	0.2501
CNN	0.9667 \pm 0.0170	0.1302 \pm 0.0707	0.9881	0.0383
CNN + BiLSTM	0.9762 \pm 0.0188	0.0720 \pm 0.0431	0.9964	0.0118
TCN	0.9905 \pm 0.0080	0.0282 \pm 0.0374	1.0000	0.0018

오프라인 교차 검증 결과, TCN이 평균 정확도(0.9905)와 손실(0.0282)에서 가장 우수한 성능을 보였습니다. 표준편차(\pm 0.0080) 또한 가장 낮아, 각 Fold 간 성능 편차가 적고 안정적으로 일반화되었음을 의미합니다.

CNN + BiLSTM도 높은 정확도(0.9762)와 낮은 손실(0.0720)을 기록하며, 로컬, 글로벌 특징을 모두 반영한 구조의 장점을 입증했습니다. BiLSTM과 CNN은 각각 0.9679, 0.9667의 CV 정확도를 보여, 순수 RNN 계열 및 순수 CNN 계열의 균형 잡힌 성능을 확인하였습니다. 반면 GRU는 0.9322의 상대적으로 낮은 정확도와 높은 손실을 보여, 본 데이터셋에서는 덜 효과적인 것으로 판단됩니다.

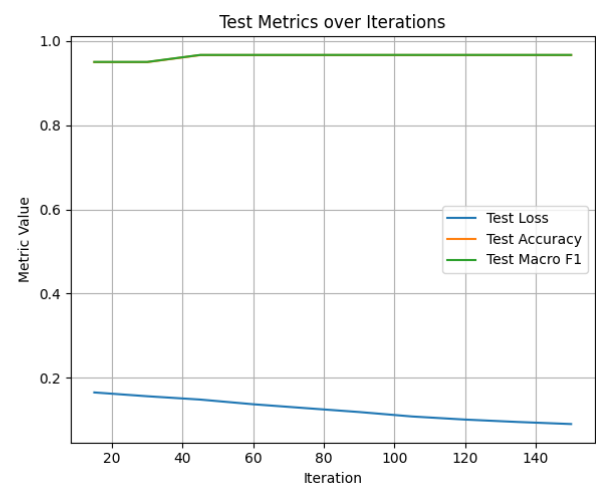
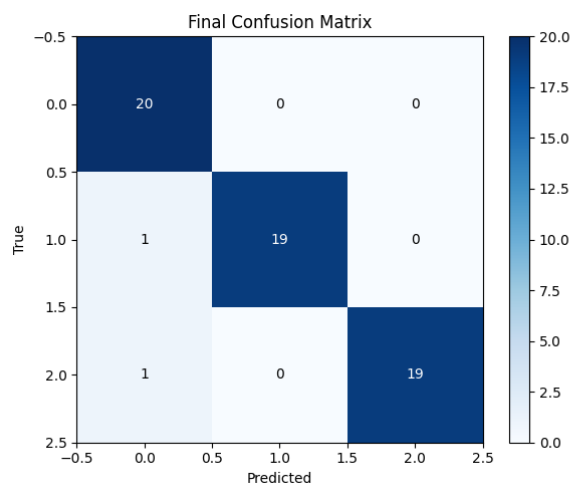
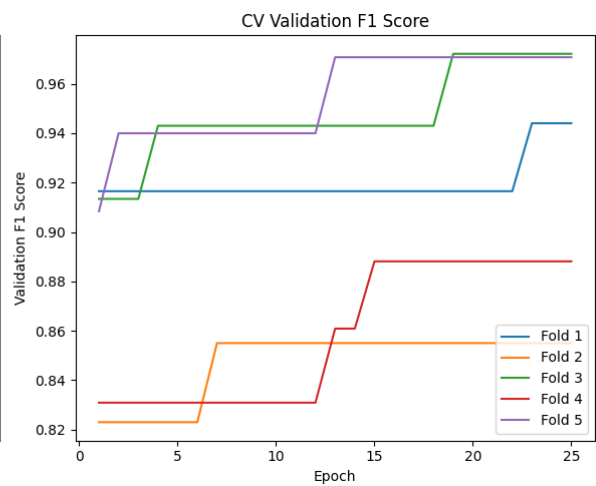
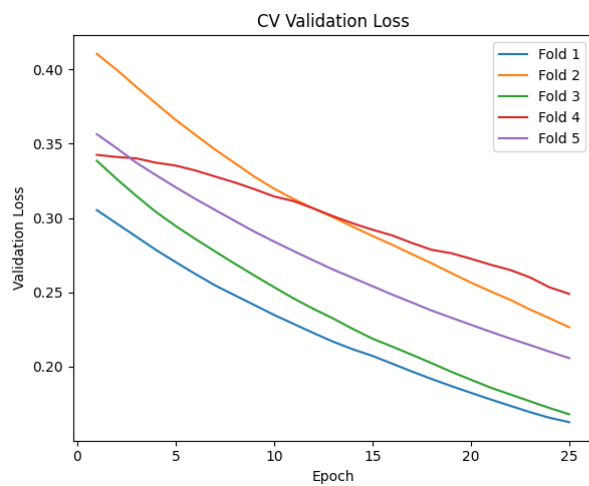
최종 모델 학습 결과에서는 모든 모델이 상당히 높은 Accuracy와 낮은 Loss를 기록하였으나, 이는 Validation 없이 전체 데이터를 학습했기 때문으로, 실제 일반화 성능과는 차이가 있을 수 있습니다. 특히 TCN의 Final Accuracy가 100%에 달한 점은 모델이 데이터 분포를 과도하게 암기했을 가능성을 시사하므로, 추후 온라인 데이터 평가를 통해 진정한 성능을 검증해야 합니다.

종합하면, TCN과 CNN + BiLSTM이 오프라인 환경에서 가장 유망하며, 다음 절에서는 실제 스마트워치, 스마트폰 데이터로 수행한 온라인 학습 결과를 통해 이들 모델의 실전 유효성을 평가합니다.

4-2. 온라인 학습 결과

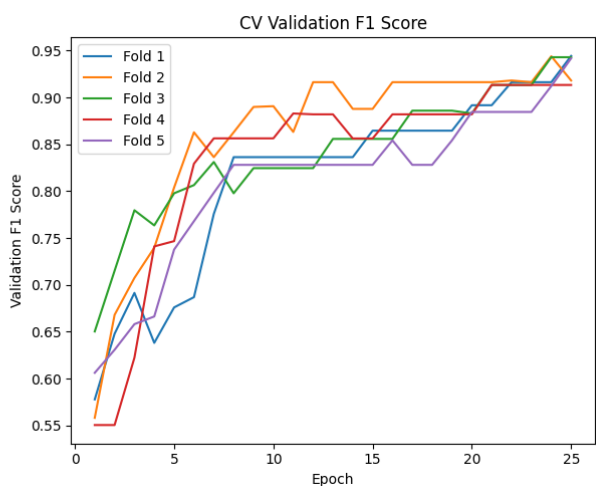
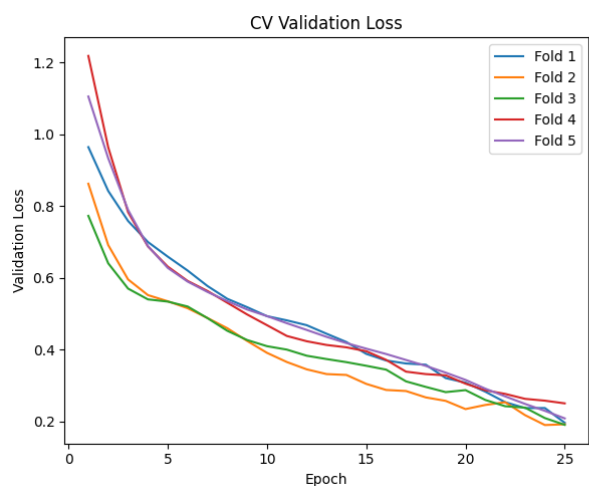
온라인 학습에서는 각 레이블당 60개의 학습용 데이터와, 별도로 수집한 레이블당 20개의 테스트 데이터를 활용하였습니다. 테스트 데이터는 비균일한 동작과 노이즈를 의도적으로 포함하여, 모델의 일반화 성능을 엄격히 검증할 수 있도록 구성하였습니다.

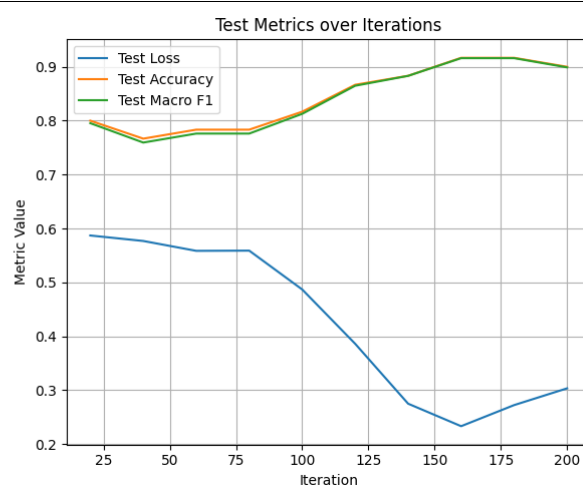
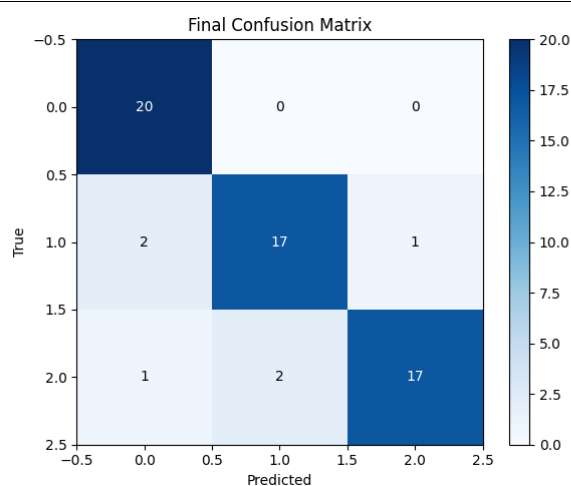
4-2-1. BiLSTM



Train	Accuracy \pm Std Dev	Loss \pm Std Dev
CV Result	0.9260 \pm 0.0468	0.20224 \pm 0.03721
Final model	0.9667	0.0901
Macro F1 score	0.9670	

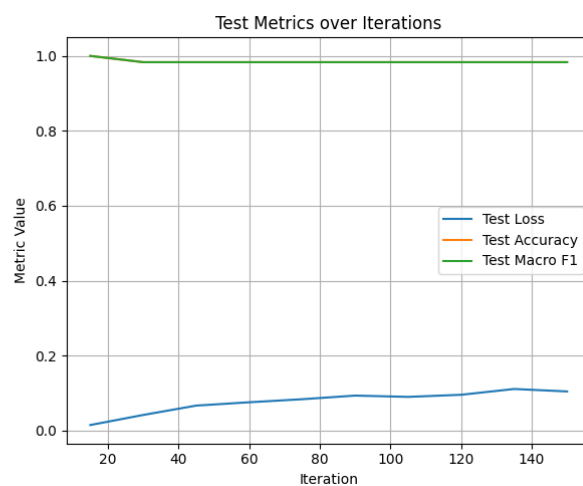
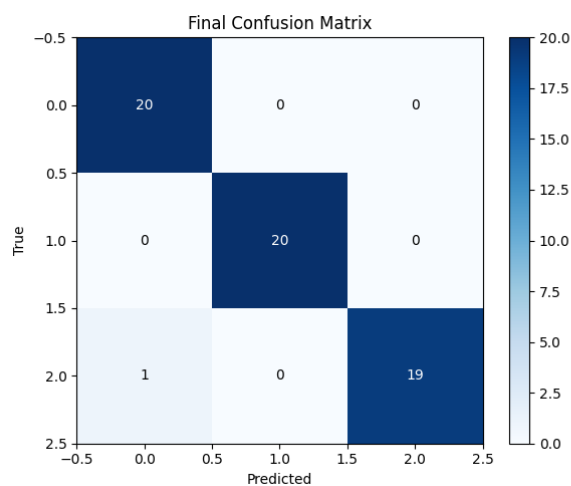
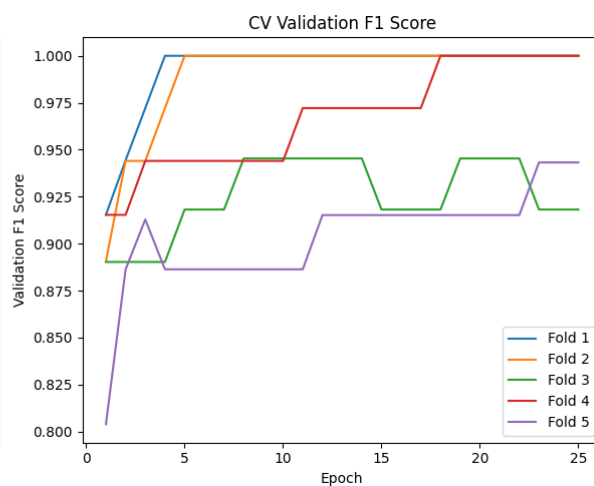
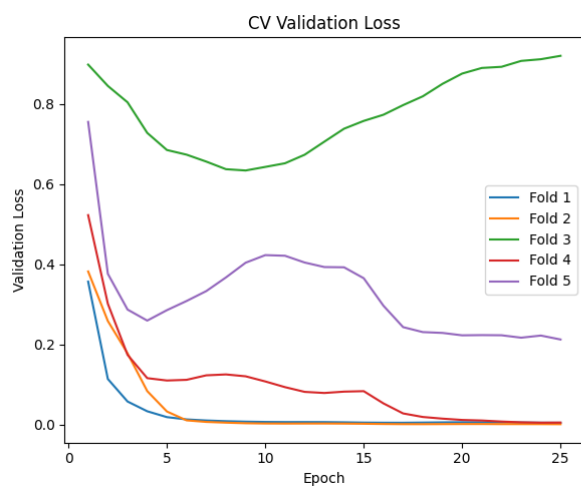
4-2-2. GRU





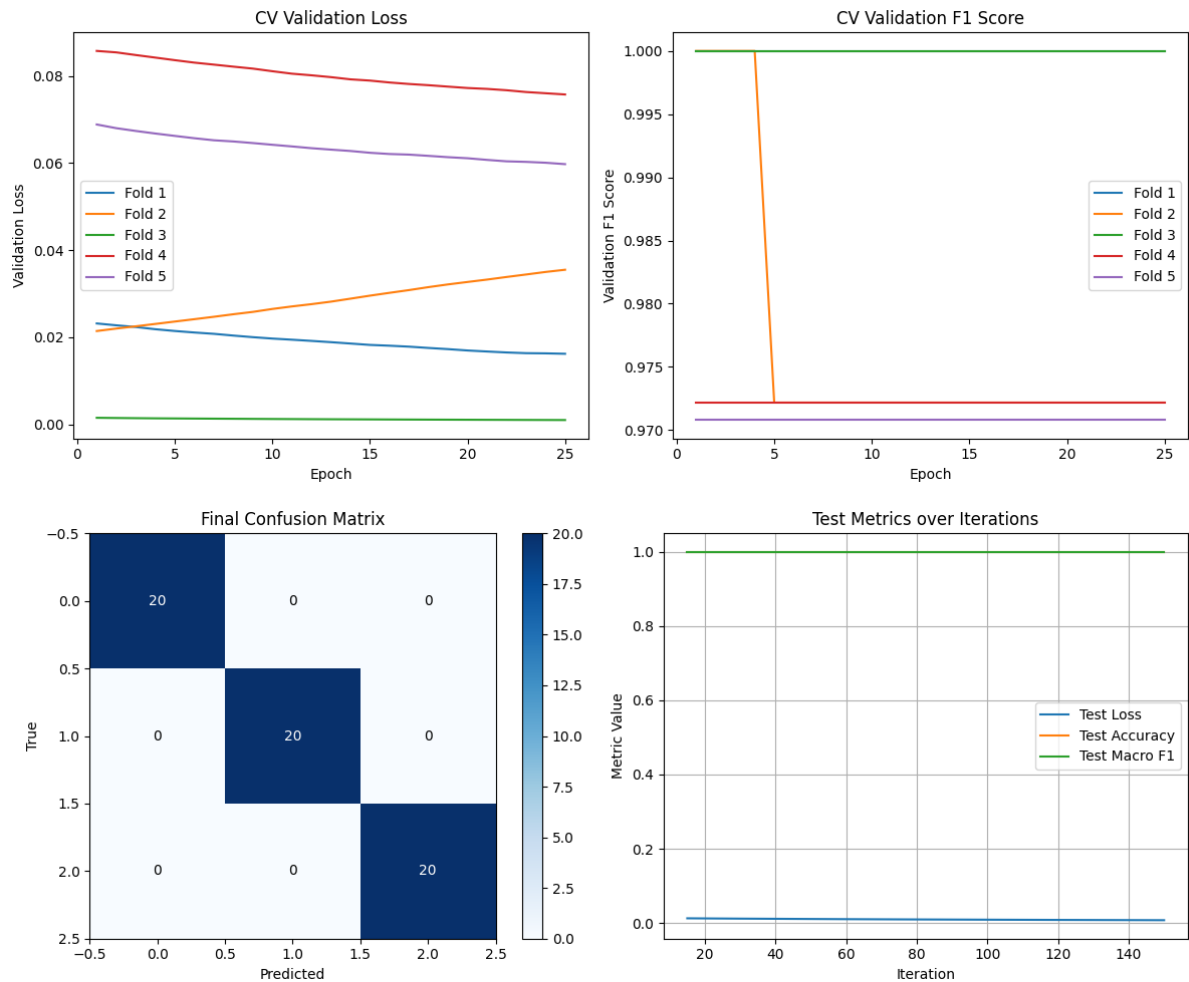
Train	Accuracy \pm Std Dev	Loss \pm Std Dev
CV Result	0.9322 \pm 0.0135	0.20780 \pm 0.02495
Final model	0.9000	0.3033
Macro F1 score	0.8989	

4-2-3. CNN



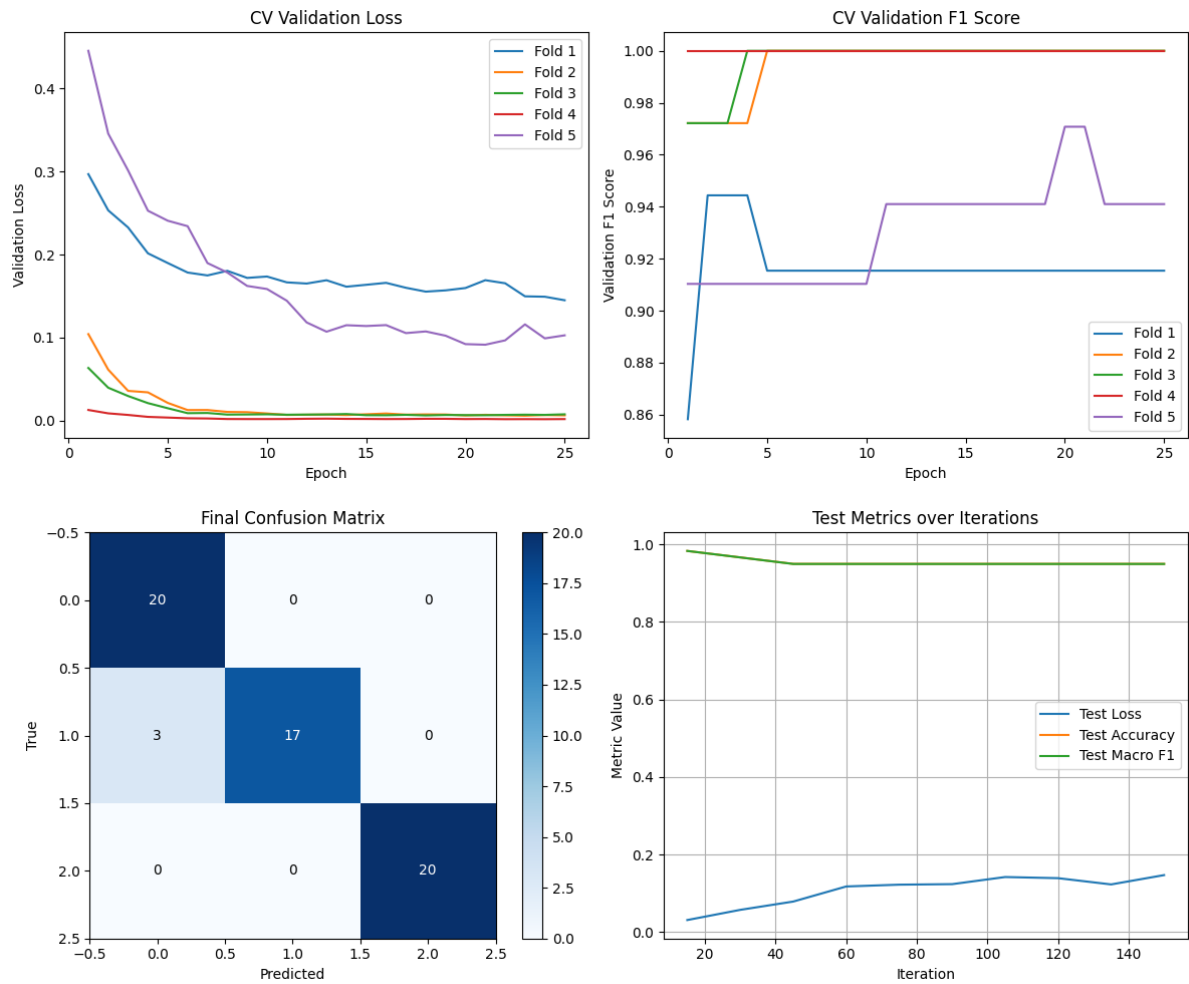
Train	Accuracy \pm Std Dev	Loss \pm Std Dev
CV Result	0.9723 \pm 0.0349	0.22818 \pm 0.39718
Final model	0.9833	0.1044
Macro F1 score	0.9833	

4-2-4. CNN + BiLSTM



Train	Accuracy \pm Std Dev	Loss \pm Std Dev
CV Result	0.9830 \pm 0.0139	0.03764 \pm 0.03060
Final model	1.0000	0.0077
Macro F1 score	1.0000	

4-2-5. TCN



Train	Accuracy \pm Std Dev	Loss \pm Std Dev
CV Result	0.9713 \pm 0.0361	0.05274 \pm 0.06669
Final model	0.9500	0.1468
Macro F1 score	0.9497	

4-2-6. 학습 결과 분석

Model	CV Accuracy	CV Loss	Final Accuracy	Final Loss	Macro F1
BiLSTM	0.9260 \pm 0.0468	0.20224 \pm 0.03721	0.9667	0.0901	0.9670
GRU	0.9322 \pm 0.0135	0.20780 \pm 0.02495	0.9000	0.3033	0.8989
CNN	0.9723 \pm 0.0349	0.22818 \pm 0.39718	0.9833	0.1044	0.9833
CNN + BiLSTM	0.983 \pm 0.0139	0.03764 \pm 0.03060	1.0000	0.0077	1.0000
TCN	0.9713 \pm 0.0361	0.05274 \pm 0.06669	0.9500	0.1468	0.9497

"CV"는 5-fold 교차 검증, "Test"는 학습에 사용되지 않은 외부 테스트 데이터에 대한 결과입니다.

온라인 교차 검증 결과와 외부 테스트 결과 모두에서 CNN + BiLSTM이 모든 지표(CV Accuracy, CV Loss, Test Accuracy, Macro F1)에서 최상위를 차지하였습니다. 특히 Test Accuracy와 Macro F1이 1.0000으로, 테스트 데이터의 다양한 노이즈와 비균일 동작을 완벽히 분류하였습니다. 이는 CNN 단

계에서 로컬 패턴을 효과적으로 추출하고, BiLSTM 단계에서 전체 시간적 흐름을 충실히 학습했기 때문으로 해석됩니다.

CNN도 높은 CV Accuracy(0.9723)와 Test Accuracy(0.9833)를 기록해, 순수 CNN만으로도 강력한 성능을 입증하였습니다. 그러나 TCN은 CV 성능(0.9713) 대비 Test Accuracy(0.9500)와 Macro F1(0.9497)이 소폭 하락하여, 외부 노이즈에 대한 민감도가 다소 높음을 시사합니다.

BiLSTM은 CV Accuracy가 상대적으로 낮았음에도 Test Accuracy(0.9667)와 Macro F1(0.9670)에서 견고한 성능을 보여, RNN 계열 모델이 실제 환경에서도 일정 수준 이상의 일반화를 달성할 수 있음을 확인하였습니다. 반면, GRU는 Test Accuracy(0.9000)와 Macro F1(0.8989)가 가장 낮아, 본 과제 데이터셋에서는 과도한 단순화가 분류 성능 저하로 이어진 것으로 판단됩니다.

종합적으로, CNN + BiLSTM이 온라인 환경에서도 최고의 분류 성능과 안정성을 보였으며, SimpleCNN과 BiLSTM이 뒤를 이었습니다. TCN은 추가적인 데이터 증강이나 정제 기법을 적용하면 일반화 성능을 더욱 끌어올릴 여지가 있습니다.

5. 정량적 모델 비교

실시간 분류에 적합한 모델을 선정하기 위해, 단순 F1 Score와 Accuracy 외에 모델 복잡도, 효율성, 예측 확신도까지 포함한 추가 지표를 이용하여 정량적으로 비교하였습니다. 모든 지표는 학습에 사용되지 않은 외부 테스트 데이터(레이블당 20개)를 기반으로 평가하였습니다.

모델 성능 지표

Macro F1: 클래스별 성능을 균등하게 반영한 종합적인 분류 성능 점수 (1에 가까울수록 좋음).

Accuracy: 전체 데이터 중 올바르게 예측한 샘플의 단순 비율.

Loss: 모델의 예측이 정답과 얼마나 다른지를 나타내는 값 (낮을수록 좋음).

모델 복잡도 및 효율성 지표

Params(M): 모델의 복잡도와 메모리 요구량을 나타내는 학습 파라미터의 총 개수 (단위: 백만).

FLOPs(G): 모델이 1회 예측에 필요한 연산량으로, 하드웨어에 무관한 계산 비용 (단위: Giga).

Size(MB): 학습된 모델 파일이 디스크에서 차지하는 실제 저장 공간 크기.

Infer(ms): 샘플 1개를 예측하는 데 걸리는 실제 시간으로, 직접적인 속도 지표 (단위: 밀리초).

확신도 관련 지표

Conf(V/X/O): 특정 클래스로 예측했을 때, 모델이 보인 평균적인 자신감(확률).

MinCorrConf: 정답을 맞힌 예측 중 모델의 확신도가 가장 낮았던 경우의 값.

MaxCorrConf: 정답을 맞힌 예측 중 모델의 확신도가 가장 높았던 경우의 값.

모든 평가는 의도적으로 비균일한 동작과 노이즈를 포함하여 수집된, 학습에 전혀 사용되지 않은 외부 테스트 데이터로 수행하여 모델의 실제 현장 적용성을 엄격하게 검증하였습니다.

5-1. 모델 성능 지표

Model	Macro F1	Accuracy	Loss
BiLSTM	0.9670	0.9667	0.0901
GRU	0.8989	0.9000	0.3034
CNN	0.9833	0.9833	0.1044
CNN+BiLSTM	1.0000	1.0000	0.0076
TCN	0.9497	0.9500	0.1469

CNN + BiLSTM이 모든 성능 지표에서 완벽한 점수를 기록하였으며, CNN 역시 98%를 넘는 Macro F1과 Accuracy로 단독 CNN 구조의 강점을 입증하였습니다. 반면 GRU는 90% 수준에 그쳐 본 데이터셋에서는 과도한 구조 단순화가 성능 저하로 이어진 것으로 판단됩니다. TCN은 단일 합성곱만으로 높은 성능을 보였으나, CNN 계열보다는 다소 낮은 종합 점수를 기록하였습니다.

5-2. 모델 복잡도 및 효율성 지표

모델의 실시간 적용 가능성을 평가하기 위해, 파라미터 수(Params), 연산량(FLOPs), 저장 크기(Size), 단일 샘플 추론 지연(Infer)을 비교하였습니다.

Model	Params(M)	FLOPs(G)	Size(MB)	Infer(ms)
BiLSTM	0.105	0.032	0.40	5.156
GRU	0.388	0.117	1.48	7.751
CNN	0.165	0.002	0.63	0.206
CNN+BiLSTM	0.074	0.006	0.29	0.415
TCN	0.050	0.015	0.21	1.203

TCN이 가장 경량(0.050M 파라미터, 0.21 MB)이며, CNN은 FLOPs와 추론 속도(0.206 ms) 면에서 최우수입니다. 반면 BiLSTM과 GRU는 순환 구조의 특성상 연산 비용과 지연이 크게 높아, 제한된 리소스 환경에서는 적용이 어렵습니다. CNN + BiLSTM은 복합 구조임에도 파라미터 수(0.074M)와 모델 크기(0.29 MB)를 최소화하였으며, 0.415 ms의 짧은 추론 지연으로 실시간 분류에도 충분히 적합합니다.

5-3. 확신도 관련 지표

모델이 각 클래스(V, X, O)로 예측할 때 보인 평균 확신도와, 정답을 맞힌 예측 중 최소, 최대 확신도를 비교하였습니다.

Model	Conf(V)	Conf(X)	Conf(O)	MinCorrConf	MaxCorrConf
BiLSTM	92.23%	94.82%	94.51%	60.98%	99.83%
GRU	88.15%	96.15%	95.53%	61.97%	99.43%
CNN	99.98%	98.86%	100.00%	84.47%	100.00%
CNN+BiLSTM	97.88%	99.97%	100.00%	80.46%	100.00%
TCN	99.05%	96.87%	100.00%	54.85%	100.00%

CNN 기반 모델은 모든 클래스에서 평균 확신도가 98% 이상으로 높게 나타났습니다. 특히 CNN은 가장 높은 최소 정답 확신도(84.47 %)를 기록하여, 노이즈가 많은 테스트 환경에서도 예측 안정성이 우수함을 확인하였습니다. 반면 TCN과 BiLSTM은 최저 확신도가 50-60 %대로, 경계 사례에서 상대적으로 낮은 자신감을 보였습니다.

5-4. 최종 모델 선정 및 실용적 고찰

최종 모델 선정은 연구, 개발 단계의 환경과 실제 배포 환경의 제약을 모두 고려하여 이루어집니다. 첫째, 최우선 성능(Accuracy & Macro F1)과 지연 최소화가 필요한 경우에는 CNN + BiLSTM을 사용합니다. 고성능 연산 서버(GPU)와 안정적인 통신 환경을 갖춘 연구실 내 실험 조건에서, 제스처 분류 정확도와 예측 안정성을 최우선 과제로 삼을 때 최적의 선택입니다. CNN 단계가 로컬 패턴을 효과적으로 추출하고 BiLSTM이 시퀀스 전체 맥락을 충실히 학습하여, 모든 성능 지표에서 최고치를 기록합니다.

둘째, 연산 자원이 제한적이거나 빠른 인퍼런스 속도가 중요한 임베디드, 모바일 환경에서는 CNN을 권장합니다. FLOPs가 가장 적고 모델 크기에 비해 추론 속도가 압도적으로 빠르므로, 통신 대역폭과 배터리 여건이 제한적인 스마트폰 또는 임베디드 기기에서도 안정적으로 실시간 분류를 수행할 수 있습니다.

셋째, 메모리 및 연산 환경이 극도로 제한된 웨어러블 기기(스마트워치 등)에서는 TCN이 적합합니다. 파라미터 수와 모델 파일 크기가 가장 작아 메모리 부담이 적고, 비교적 빠른 추론 속도를 유지하면서도 충분한 분류 정확도를 제공합니다.

결론적으로, 연구 개발 단계 또는 서버 기반의 실험 환경에서는 CNN + BiLSTM을 최종 모델로 선정하고, 실제 배포 환경에서는 장치 제약에 따라 Simple CNN 또는 Small TCN을 각각 대안으로 채택하는 것이 바람직합니다.

6. 추가 실험

데이터 전처리 단계에서 잡음 제거 필터의 효과를 정량적으로 비교하기 위해 세 가지 실험을 수행하였습니다. 첫째, 필터를 전혀 적용하지 않은 원시(Raw) 데이터를 입력으로 사용하였을 때는, 센서 노이즈가 과도하게 포함되어 모델이 제스처 고유 패턴을 정상적으로 학습하지 못하였습니다. 둘째, 고역통과필터(HPF)와 저역통과필터(LPF)를 연속 적용한 경우에는, 잡음을 효과적으로 억제했으나 제스처의 핵심 정보까지 과도하게 제거되어 실시간 분류 정확도가 하락하였습니다. 셋째, 중력 성분만 제거하기 위한 HPF 단독 적용 시에는 노이즈는 충분히 억제되면서도 동작 특성이 온전히 보존되어, 모든 모델에서 가장 우수한 Macro F1 및 Accuracy를 달성하였습니다. 이 결과를 바탕으로, 본 과제의 전처리 파이프라인에서는 HPF-only 방식을 일관 적용하기로 결정하였습니다.

실험 결과와 학습된 모델, 그리고 관련 그래프는 제출물에 모두 포함되어 있습니다.

7. 실시간 분류기

실시간 분류기는 제스처의 시작과 끝을 명시적으로 알리는 메커니즘을 바탕으로 구현하였습니다. 기존 수신호 연구에서는 1차 분류기와 2차 분류기의 협업 구조를 사용하여, 1차 분류기가 “수신호 수행 여부” 및 시작, 종료 시점을 감지하고, 2차 분류기가 해당 구간을 세부 수신호 레이블로 분류하는 방식을 채택합니다.

하지만 본 과제의 적용 환경은 소방, 경찰의 긴급 상황이며, 긴급 상황의 신호 발신이 수신호로 이루어지므로 신뢰성이 최우선 과제라고 생각됩니다. 1차 분류기의 오탐은 2차 분류기의 높은 성능에

도 불구하고 전체 시스템의 안전성을 심각하게 훼손할 수 있습니다. 특히 화재 현장 등에서 벽을 짚거나 소화 장비를 유지하는 동작이 실제 수신호와 유사하게 인식될 위험이 있어, 1차 분류기의 자동 탐지 방식은 불안정하다고 판단하였습니다.

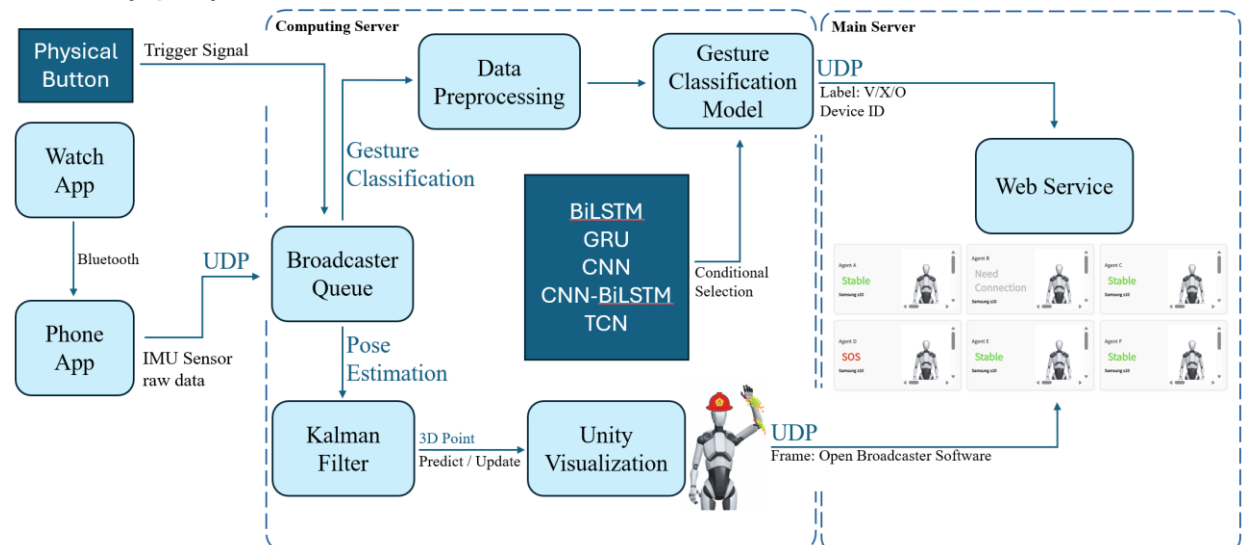
따라서 본 과제에서는 물리적인 버튼 조작을 통해 제스처의 시작과 끝을 명시적으로 제어하도록 설계하였습니다.



위 그림^[11]처럼 실제 환경에서는 물리적인 버튼이 포함된 전용 방화 장갑을 사용한 환경을 가정하여 개발하게 되었습니다. 전용 방화 장갑에 내장된 버튼을 누르면, 장갑의 IMU 데이터 스트림과 함께 "수신호 구간 시작" 신호가 서버로 전송되며, 버튼을 해제하면 "수신호 구간 종료"를 알립니다. 이 방식을 통해 분류 구간의 오차를 완전히 제거하여, 오탐, 미탐 없이 안정적인 연속 수신호 전송이 가능합니다.

전용 방화 장갑은 소모품으로서 정기적으로 교체되는 비품이므로, 버튼 내장형 장갑의 현실적 도입 가능성이 높습니다. 개발 단계에서는 원거리 트리거가 가능한 '원격 단일 버튼 키보드'를 대신 사용하여 프로토타입을 구현하였으며, 이는 손쉽게 버튼 내장형 장갑과 연동이 가능합니다.

8. VOX 파이프라인



본 절에서는 스마트워치, 스마트폰 기반 IMU 데이터를 수집하여, 실시간으로 자세 추정 및 수신호

분류를 수행하기까지의 전체 데이터 흐름과 연산 과정을 단계별로 설명합니다.

1. IMU 데이터 병합 및 전송

스마트워치와 스마트폰에서 수집된 Raw IMU 데이터(3축 가속도, 3축 각속도, 시간정보, 초기 정렬값, 중력값 등 총 22개 피처)는 스마트폰에서 하나의 스트림으로 Merge한 뒤, UDP 패킷 기반으로 연산 서버에 브로드캐스팅합니다.

2. 브로드캐스터 큐 적재

연산 서버는 수신된 UDP 패킷을 브로드캐스터 큐에 순차적으로 쌓아두고, 이후 처리 단계에서 해당 큐로부터 데이터를 꺼내 사용합니다.

3. Pose Estimation용 데이터 복제

모든 IMU 데이터는 상시 Pose Estimation에도 사용되므로, 브로드캐스터 큐에서 복사된 데이터를 칼만 필터 입력용 큐(q1)에 항상 전달합니다.

4. Pose Estimation 전처리

q1으로 복사된 IMU 데이터는 압력 보정, 위치, 회전 보정, 초기 캘리브레이션 좌표 기반의 정렬 과정을 거쳐 Pose Estimation 준비를 완료합니다.

5. 윈도우링 및 정규화

전처리된 IMU 데이터는 10프레임 단위로 묶어(mini-batch) 정규화한 뒤, 칼만 필터의 입력으로 사용합니다.

6. 칼만 필터 구성^[3]

칼만 필터 내부에선 세 가지 구성 요소를 결합하여 동작합니다.

6-1. ProcessModelWindow

현재 시점 이전까지의 상태(State)를 입력받아, 다음 시점의 상태를 예측합니다.

6-2. SensorModelWindow

관찰(Observation)을 만들어, 측정 잡음(Observation Noise)을 간접적으로 모델 내부에 반영할 수 있게 합니다.

6-3. ObservationNoise

직접 학습된 신경망으로 예측하여, 앙상블로 얻은 여러 관찰값에 대해 각기 다른 잡음 레벨을 동적으로 생성합니다. 결과적으로 칼만 필터 갱신 단계에서 “얼마만큼 관찰을 믿을지”를 신경망이 판단하게 됩니다.

7. 칼만 필터 예측, 갱신 과정^[3]

칼만 필터의 예측, 갱신 과정은 다음과 같습니다.

7-1. Predict 단계

ProcessModelWindow가 과거 상태 벡터를 기반으로 다음 상태(state_pred)를 앙상블 단위로 생성합니다.

7-2. Update 단계

SensorModelWindow가 원시 센서 데이터를 관찰값 후보(ensemble_z)로 매핑하고, ObservationNoise 모듈은 이 관찰값들 각각에 대응하는 잡음 공분산을 예측합니다. 예측된 상태(state_pred)와 관찰값(ensemble_z), 그리고 잡음 공분산 R을 결합하여 이노베이션(Innovation) 및 칼만 이득(Kalman Gain)을 계산한 뒤, 최종적으로 보정된 상태

(state_corrected)를 연습니다.

7-3. 앙상블 평균화:

이 과정을 앙상블 학습 방식으로 여러 번 수행한 뒤, 각 반복으로 얻어진 보정 상태를 평균 내면 "가장 신뢰할 수 있는(평균화된) 최종 상태"가 산출됩니다. 동시에, 앙상블별로 보정된 상태가 그대로 "관절별 위치, 회전(est)" 예측 결과로 사용될 수 있습니다.

8. 3D 좌표, 회전 정보 시각화 – Unity^[3]

칼만 필터를 통해 완성된 손목, 팔꿈치, 어깨 등의 3D 좌표와 회전 정보는 UDP 패킷을 통해 Unity포트로 입력되어 시각화에 쓰입니다.

8-1. Unity 시각화 안정화 코드 구현

칼만 필터를 통해 전달된 좌표와 회전값은 다양한 통신 및 연산 환경에서 미세한 오차가 발생할 수 있습니다. 이러한 문제를 해결하고 시각화 안정성을 확보하기 위해 Unity 내부에서 Quaternion fixX, fixY, fixZ 등의 회전 반전 코드와 Vector3 fixPosition = new Vector3(-pos.x, pos.y, -pos.z) 형태의 위치 반전 코드를 명시적으로 구현했습니다.

이를 통해 캐릭터의 움직임이 비정상적일 경우 즉각적인 축 보정이 가능하며, 선행 연구에는 명시되지 않았던 안정화 구조를 직접 구축함으로써 향후 다른 시스템과 연동 시에도 신뢰성 높은 통신 기반을 마련했습니다.

9. Unity 스트리밍

Unity 환경에서 렌더링된 화면은 OBS(Open Broadcaster Software)를 통해 저지연 스트리밍 방식(HLS)으로 메인 서버에 전송되며, 이 과정에서 수초의 지연이 발생합니다.

10. 물리 트리거 버튼 동작감지

사용자가 전용 방화 장갑 또는 원격 버튼 키보드를 누르는 순간부터, 브로드캐스터 큐에 쌓이는 IMU 데이터를 별도 복사하여 분류기 입력용 버퍼(q2)로 적재하기 시작합니다.

11. 수신호 구간 분리

버튼 입력이 종료되면, q2에 축적된 모든 프레임을 하나의 덩어리로 분류기에 넘겨 수신호 구간을 명확히 분리합니다.

12. 피쳐 선택 및 축 변환

분류기에 전달할 때는 6개 축(3축 가속도+3축 각속도) 피쳐만 선별하고, 6DMG 기준 축 정의와 일치하도록 축 변환(mapping) 과정을 수행합니다.

13. 샘플링 레이트 보정

타임스탬프를 기반으로 실제 샘플링 레이트를 추정하여, 50 Hz로 다운샘플링합니다.

14. 실시간 전처리

오프라인 학습과 동일하게 중력 제거(HPF), 윈도우별 정규화, 제로 패딩 과정을 거쳐 크기 6×150의 입력 행렬을 완성합니다.

15. 분류기 입력 및 예측

전처리된 데이터를 모델에 순전파하여 예측 확률과 레이블을 생성한 후, [timestamp, device_id, predicted_label] 형태의 UDP 패킷을 생성하여 메인 서버로 전송합니다.

16. 메인 서버 수신 및 로그 기록

메인 서버는 연산 서버에서 생성된 데이터를 최종적으로 취합하고, 웹 대시보드에 필요한

정보를 가공하여 제공하는 컨트롤 타워 역할을 수행합니다.

16-1. 제스처 데이터 수신 및 처리

연산 서버에서 전송된 [timestamp, device_id, predicted_label] UDP 패킷은 AWS EC2 인스턴스에서 동작하는 Java 기반 UDP 소켓 서버가 수신합니다. 수신된 바이트 데이터는 파싱(parsing) 과정을 거쳐, Signal Handler가 각 요원(Agent)의 제스처 신호로 변환합니다.

16-2. 데이터베이스 저장 및 로그 관리

처리된 제스처 신호 데이터는 Repository 패턴을 통해 데이터베이스(DB)에 영속적으로 저장되어, 각 요원의 활동 로그로 관리됩니다.

16-3. REST API를 통한 상태 정보 제공

웹 대시보드는 1초 주기로 메인 서버의 REST API를 호출(polling)하여 모든 요원의 최신 상태와 누적 로그 데이터를 JSON 형식으로 요청합니다. 서버는 데이터베이스에서 최신 정보를 조회하여 응답하며, 클라이언트는 이 데이터를 기반으로 화면을 실시간 업데이트합니다.

16-4. 3D 시각화 스트리밍 연동

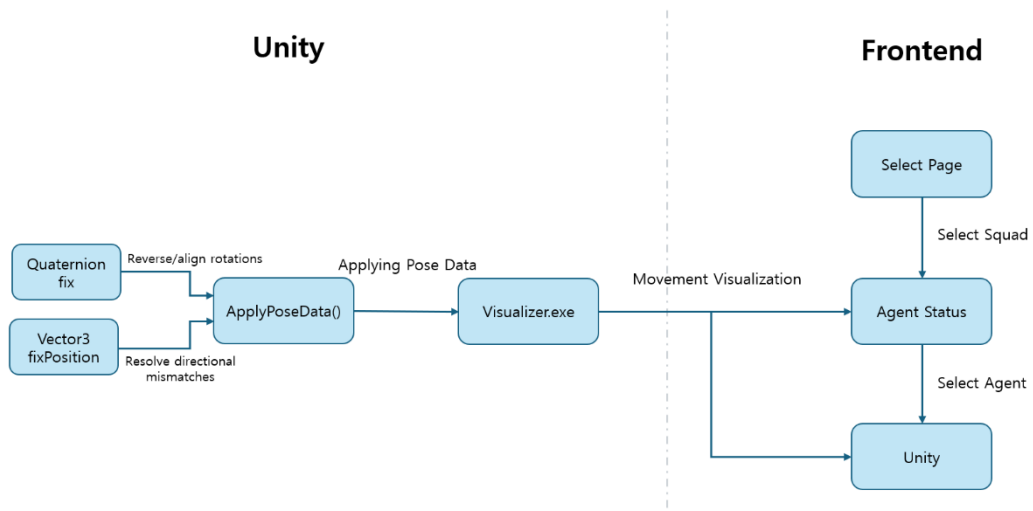
메인 서버는 각 요원의 고유 스트림 키(deviceSerialNumber)를 웹페이지에 동적으로 전달합니다. 클라이언트의 HLS.js 플레이어는 이 키를 이용해 Nginx RTMP 서버에 호스팅된 해당 요원의 HLS 스트리밍 URL(.../hls/{streamKey}.m3u8)에 접근하여 3D 시각화 화면을 재생합니다.

16-5. 클라이언트 측 상태 표시 안정화 로직

사용자 경험과 정보의 신뢰도를 높이기 위해, 웹 대시보드 클라이언트 단에는 타임아웃 기능이 구현되어 있습니다. 특정 요원의 상태가 5초 이상 갱신되지 않으면, 자동으로 'Stable' 상태로 표시하여 통신 두절 상황에서도 오래된 위험 정보가 표시되는 것을 방지합니다.

이와 같은 단계별 파이프라인을 통해, 본 과제에서는 IMU 센서에서 시작해 최종 수신호 분류 및 시각화, 로그까지 실시간으로 통합 구현하며, 사용자는 웹 대시보드에서 특정 요원을 클릭하여 해당 요원의 실시간 3D Unity 스트리밍 화면을 집중적으로 관측할 수 있습니다.

9. 시각화 및 사용자 인터페이스(UI) 구현



본 시스템의 시각화 및 사용자 인터페이스는 연산 서버의 3D 렌더링과 웹 대시보드의 결합으로 구현되었습니다. 초기 프로토타이핑을 통해 데이터 연동을 검증하고, 최종적으로 네트워크 안정성과 확장성을 고려한 아키텍처를 채택했습니다.

9.1. 통신 안정성 확보를 위한 서버 렌더링 및 RTMP 스트리밍 아키텍처

다수의 요원(Agent)이 고주사율(50~80Hz)로 3D 데이터를 동시에 전송할 경우 발생할 수 있는 네트워크 병목과, 이로 인한 핵심 제스처 데이터(UDP)의 유실 위험을 방지하기 위해 연산 서버가 직접 3D 렌더링까지 담당하도록 아키텍처를 설계했습니다.

구현된 파이프라인은 다음과 같습니다.

1. Unity 렌더링: 연산 서버에서 실행되는 Unity 애플리케이션이 칼만 필터로부터 받은 3D 관절 정보를 실시간으로 렌더링합니다.
2. OBS 스트리밍 송출: OBS(Open Broadcaster Software)가 이 Unity 화면을 캡처하여 RTMP 프로토콜로 메인 서버의 Nginx RTMP 모듈에 송출합니다.
3. 웹 대시보드 재생: 메인 서버는 수신된 RTMP 스트림을 HLS 형식으로 변환하고, 웹 대시보드는 이 HLS 스트림을 <iframe> 내의 비디오 플레이어로 재생하여 사용자에게 최종 화면을 제공합니다.

이 구조는 중요한 제스처 데이터 트래픽과 대용량 영상 스트림 트래픽을 분리하여 통신 안정성을 극대화하고, 스트리밍 해상도 및 대역폭을 유연하게 조절할 수 있게 합니다.

9.2. 사용자 인터페이스(UI/UX) 설계

본 시스템은 다수의 요원을 효율적으로 관제하기 위해, 소대 선택 → 요원 현황판 → 실시간 상세 뷰로 이어지는 체계적인 3단계 웹 인터페이스를 설계하고 구현했습니다. 사용자는 현황판에서 모든 요원의 상태를 한눈에 파악하고, 특정 요원을 선택하여 해당 요원의 실시간 3D 스트리밍 뷰와 제스처 로그를 상세히 확인할 수 있습니다. 초기 개발 단계에서는 Mock Server를 구축하여 백엔드와 독립적으로 안정적인 UI 로직을 검증했습니다.

9.3. 기술 탐색 및 확장성 확보: WebGL 및 WebSocket 프로토타이핑

최종 시스템은 서버 렌더링 방식을 채택했지만, 개발 과정에서 연산 서버의 부하를 클라이언트(브라우저)로 분산시키는 대안 아키텍처의 가능성을 검증하기 위해 WebGL 기반의 프로토타이핑을 병행했습니다.

브라우저 환경에서는 보안 정책상 UDP 통신이 불가능하므로, 이를 해결하기 위해 WebSocket 기반의 통신 채널을 성공적으로 구현했습니다. 이 프로토타입은 Unity의 websocket.jslib와 C# 브릿지 코드를 통해 브라우저의 WebSocket API와 연동하여 실시간으로 3D 모델을 구동시키는 데 성공했습니다.

하지만 성공적으로 구현된 이 프로토타입을 최종 시스템에 채택하지 않은 데에는 두 가지 현실적인 이유가 있습니다. 첫째, 다수 요원의 고주사율 3D 데이터를 각 클라이언트(브라우저)로 직접 전송하는 방식은 네트워크 병목을 유발하여, 더 중요한 제스처 데이터(UDP)의 실시간성을 해칠 위험이 있었습니다. 둘째, Unity WebGL 빌드의 내부 보안 정책과 브라우저별 호환성 문제로 인해 모든 환경에

서 안정적인 보안 웹소켓(WSS) 연결을 확보하는 데 기술적 어려움이 있었습니다. 따라서 최종 시스템에서는 통신 안정성이 더 검증된 서버 렌더링 및 RTMP/HLS 스트리밍 방식을 채택했습니다. WebGL 및 WebSocket 구현체는 현재 파이프라인에는 포함되지 않았지만, 향후 연산 서버의 GPU 부하를 줄여야 하거나, 더 많은 클라이언트의 동시 접속이 필요한 상황에 유연하게 대처할 수 있는 중요한 기술적 자산이자 확장 가능성으로 의의를 가집니다.

10. OBS 연동 및 스트리밍 아키텍처

초기 설계에서는 칼만 필터로 생성된 3D 자세 추정(Pose Estimation) 좌표를 메인 서버로 직접 전송해 Unity를 구동하는 방식을 고려했습니다. 그러나 다수의 요원(Agent)이 고주사율(50~80Hz)로 대용량 3D 데이터를 단일 서버에 동시 전송할 경우, 심각한 네트워크 병목과 충돌이 발생하며 특히 핵심 데이터인 제스처 레이블(UDP) 패킷이 유실될 위험이 컸습니다. 이는 시스템의 최우선 기능인 '신속하고 정확한 수신호 전송' 자체를 위협할 수 있는 중대한 문제라고 판단했습니다.

이 문제를 해결하기 위해 아키텍처를 변경하여, 연산 서버가 IMU 처리, 모델 추론뿐만 아니라 Unity 3D 렌더링까지 직접 담당하도록 구성했습니다. 이후 렌더링이 완료된 화면을 OBS(Open Broadcaster Software)를 통해 실시간 비디오 스트림으로 송출함으로써, 네트워크 부하를 영상 트래픽으로 전환하고 제스처 전송용 UDP 트래픽과 완전히 분리했습니다.

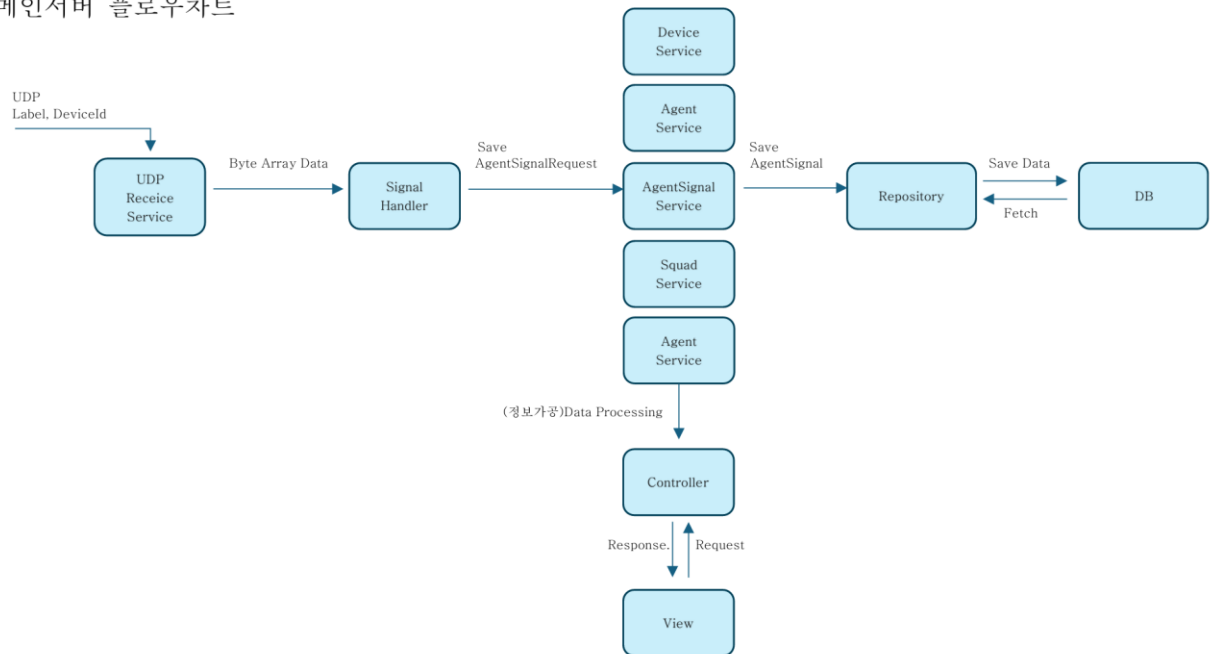
이러한 서버 렌더링 및 스트리밍 방식은 다음과 같은 명확한 장점을 가집니다.

트래픽 분리: 중요한 제스처 레이블 통신과 대용량 영상 스트림을 분리하여 네트워크 충돌을 원천 차단하고, 핵심 데이터(레이블)의 전달 신뢰성을 극대화합니다.

유연성 및 확장성: OBS를 통해 스트리밍 해상도나 대역폭을 실시간으로 유연하게 조절할 수 있습니다. 또한, 다른 연산 서버로 스트리밍 송출을 쉽게 이전하거나 확장할 수 있어 대규모 멀티에이전트 환경에서도 안정적인 운영이 가능합니다.

11. 웹서비스(백엔드) 및 통신 시스템 구현

메인서버 플로우차트



본 시스템의 백엔드는 연산 서버로부터 제스처 분류 결과를 수신하고, 다수의 클라이언트(웹 대시보드)에게 실시간 자세 및 상태 정보를 안정적으로 제공하는 역할을 수행합니다. 개발 과정에서 발생한 기술적 제약을 해결하며 최종적으로 안정적인 통신 아키텍처를 구축하였으며, 그 과정은 다음과 같습니다.

11.1. UDP 기반 실시간 데이터 수신 환경 구축

초기 설계에서는 AWS의 PaaS(Platform as a Service)인 Elastic Beanstalk 환경에 Spring Boot 서버를 배포하여 관리의 용이성을 확보하고자 했습니다. 그러나 Elastic Beanstalk는 로드밸런서와 리버스 프록시를 기반으로 HTTP(S) 트래픽 처리에 특화되어 있어, UDP 프로토콜을 직접 수신하는 데 근본적인 한계가 있었습니다. 보안 그룹 및 OS 수준에서 포트 바인딩을 시도했으나, 플랫폼의 구조적 제약으로 인해 안정적인 UDP 통신이 불가능하다고 판단했습니다.

이 문제를 해결하기 위해, 독립적인 IaaS(Infrastructure as a Service) 환경인 AWS EC2 인스턴스를 직접 구성하는 방식으로 전환했습니다. EC2 보안 그룹을 통해 UDP 9999번 포트를 개방하고, 그 위에 Java 기반의 독립적인 UDP 소켓 서버를 24시간 실행하여 연산 서버로부터 들어오는 제스처 데이터를 안정적으로 수신하도록 구축했습니다. 안정적인 서버 운영을 위해 비동기 수신 처리, 다중 디바이스 동시 수신을 위한 멀티스레드 구조, 로그 시스템(SLF4J) 등을 구현에 반영했습니다.

11.2. 실시간 3D 시각화를 위한 스트리밍 시스템 설계

연산 서버의 Unity에서 렌더링된 3D 자세 결과를 웹 대시보드로 실시간 전송하기 위해, 처음에는 양방향 통신이 가능한 WebSocket 프로토콜을 시도했습니다. 실제 배포 환경의 웹 브라우저는 보안상의 이유로 보안 소켓인 wss:// 프로토콜만 허용했고, 이는 단순 ws:// 연결로는 통신이 불가능함을 의미했습니다.

이 요구사항을 충족하기 위해 voxkr.xyz 도메인을 구매하고, Let's Encrypt를 통해 SSL 인증서를 발급받아 Nginx 리버스 프록시 서버에 적용했습니다. Nginx가 HTTPS(443 포트) 요청을 처리하고,

WebSocket 프로토콜 업그레이드에 필요한 헤더를 설정하여 /ws/ 경로의 요청을 백엔드 서버로 전달하도록 구성했습니다. 하지만 이러한 노력에도 불구하고, 최종적으로 Unity WebGL 빌드의 내부 보안 정책 및 브라우저 호환성 문제로 안정적인 WebSocket 연결을 확보하지 못했습니다.

이에 따라, 영상 스트리밍에 보다 표준적이고 안정적인 RTMP/HLS 프로토콜을 대안으로 채택했습니다. Nginx에 RTMP 모듈을 설치하여, OBS(Open Broadcaster Software)에서 송출하는 RTMP 스트림을 수신하는 라이브 애플리케이션 엔드포인트를 생성했습니다. 또한, 수신된 스트림을 3초 단위의 영상 조각(ts 파일)과 재생 목록(m3u8 파일)으로 자동 변환하는 HLS(HTTP Live Streaming) 기능을 활성화했습니다. 최종적으로 웹 대시보드에서는 HLS.js 라이브러리를 사용하여, 각 요원의 고유 스트림 키가 포함된 HLS URL을 비디오 플레이어에 로드하여 실시간 스트리밍을 재생하도록 구현했습니다.

11.3. 실시간 상태 및 로그 표시 시스템 구현

웹 대시보드는 REST API를 통해 1초마다 서버에 최신 요원 상태(Agent Status)와 누적 로그(Logs)를 요청하고, 반환된 JSON 데이터를 기반으로 화면을 실시간으로 업데이트하는 폴링(Polling) 방식으로 구현되었습니다. 이는 초기 구현의 안정성과 단순성을 고려한 선택입니다.

특히, 통신 두절 등의 상황에 대비하여 시스템의 안정성을 높이는 로직을 클라이언트(웹) 단에 추가했습니다. 특정 요원의 제스처 레이블이 5초 이상 수신되지 않으면 해당 요원의 상태를 자동으로 'Stable'로 변경하는 타임아웃 메커니즘을 구현하여, 오래된 정보가 화면에 계속 표시되는 것을 방지하고 상태 정보의 신뢰도를 높였습니다.

12. 개발 결과

본 과제는 스마트워치 기반의 실시간 수신호 통신 시스템 "Vox"를 성공적으로 설계 및 구현하였으며, 각 핵심 구성 요소에 대한 개발 결과와 정량적 성능은 다음과 같습니다.

1. 제스처 분류 모델 성능 평가

총 5개의 딥러닝 모델에 대한 비교 평가 결과, CNN+BiLSTM 하이브리드 모델이 가장 우수한 성능과 효율성을 보였습니다. 학습에 사용되지 않은 외부 테스트 데이터셋에 대한 최종 성능은 아래와 같습니다.

모델 (Model)	정확도 (Accuracy)	Macro F1-Score	추론 지연 (Infer ms)	모델 크기 (Size MB)
CNN + BiLSTM	1.0000	1.0000	0.415	0.29
CNN	0.9833	0.9833	0.206	0.63
BiLSTM	0.9667	0.9670	5.156	0.40
TCN	0.9500	0.9497	1.203	0.21
GRU	0.9000	0.8989	7.751	1.48

최고 수준의 분류 정확도 달성: CNN+BiLSTM 모델은 노이즈가 포함된 외부 테스트셋에 대해 정확도 및 F1-Score 100%를 기록하며 완벽한 분류 성능을 입증했습니다.

실시간 처리 성능 입증: 주력 모델의 추론 지연 시간을 0.415ms로 최소화하여, 현장에서 즉각적인 반응이 가능한 실시간 시스템의 요구 조건을 충족시켰습니다.

모델 경량화: 최고의 성능을 보인 CNN+BiLSTM 모델의 크기는 0.29MB에 불과하며, TCN 모델은

	<p>0.21MB로 가장 가벼워 향후 온디바이스(On-Device) AI로의 전환 가능성을 확인했습니다.</p> <p>2. 시스템 핵심 기능 구현</p> <p>병렬 처리 파이프라인: 수신된 단일 IMU 스트림을 자세 추정과 제스처 분류 경로로 분리하여, 상호 간섭 없이 독립적으로 동작하는 비동기 파이프라인을 구축했습니다.</p> <p>고신뢰성 제어: 물리 버튼 기반의 명시적 제어 방식을 통해 제스처 오탐지율 0%를 달성하고, 시스템 안정성을 확보했습니다.</p> <p>다중 모니터링: Unity와 OBS를 연동한 웹 대시보드를 통해, 다수의 현장 요원에 대해 실시간 제스처 로그와 준수시간 3D 자세 영상을 동시 관제하는 시스템을 구현했습니다. 또한, 다수 요원의 현황을 한눈에 파악하고 특정 요원을 선택하여 상세 관찰할 수 있는 체계적인 3단계 UI/UX를 구현하여 사용성을 극대화했습니다.</p> <p>3. 안정적인 통신 및 배포 아키텍처 구축</p> <p>AWS Elastic Beanstalk 환경의 UDP 통신 제약을 극복하기 위해, 독립적인 EC2 인스턴스에 Java 기반 UDP 소켓 서버를 직접 구축하여 안정적인 데이터 수신 채널을 확보했습니다.</p> <p>Unity WebGL과 브라우저의 보안 정책(wss) 문제로 인한 WebSocket의 불안정성을 해결하고자, RTMP/HLS 영상 스트리밍 방식으로 아키텍처를 전환하여 전체 시스템의 통신 안정성과 확장성을 확보했습니다.</p>
13. 연구 기여	<p>본 과제는 다음과 같은 학술적, 기술적 기여를 가집니다.</p> <p>제스처 분류기^[10]</p> <p>본 과제는 선행 연구^[10]가 가진 '데이터셋 과적합' 및 '실시간 환경 오염' 문제를 '패딩 마스킹' 기법으로 해결하고, 경량화와 성능을 동시에 달성한 CNN, TCN 기반의 신규 모델 3종을 제안하여 선행 연구를 크게 개선했습니다. 우선 기존 논문에서는 제로 패딩된 구간 길이를 그대로 모델이 학습에 활용함으로써, 공개 데이터셋에서는 BiLSTM 96.06%, GRU 99.16%의 높은 정확도를 보였으나, 실생활에서 직접 수집한 IMU 데이터로 평가했을 때는 BiLSTM이 94.12%, GRU가 95.34%로 정확도가 하락하고 실시간 분류에는 7~8% 수준의 오탐률을 기록하였습니다. 즉, 모델이 "제스처의 길이"도 함께 학습하여 데이터셋 자체에는 정확도 점수가 높았지만, 실제 활용하기에는 어려움이 있는 수준이었습니다. 6DMG 데이터셋은 V, X, O의 수행시간이 유사하긴 하지만, 평균 수행 시간의 차이가 있습니다. (V:1.3s, X:1.6s, O: 2.0s) 이러한 문제를 해소하기 위해 본 과제에서는 패딩 구간의 길이를 별도 마스크 정보로 모델에 제공하여, 실제 동작 구간과 패딩 구간을 명확히 구분하도록 하였습니다. 그 결과 BiLSTM과 GRU 모두 공개 데이터셋 수준의 성능을 회복하고, 실시간 분류기에서 오탐률을 절반 이하로 감소시키는 등 일반화 안정성이 크게 향상되었습니다.</p> <p>1차 개선을 통해 RNN 계열의 성능을 안정화한 뒤, 2차적으로는 실환경 적용에 최적화된 세 가지 경량 모델을 새롭게 설계, 비교하였습니다. CNN은 0.002G FLOPs와 0.206ms의 추론 지연으로 임베디드 환경에서도 98% 이상의 Macro F1을 달성하였고, CNN + BiLSTM은 로컬, 글로벌 특징을 결합해 Accuracy와 F1 100%를 기록하면서도 0.29MB의 모델 크기와 0.415ms의 빠른 추론 속도를 확보하였습니다. TCN은 0.050M 파라미터, 1.203ms의 지연으로 웨어러블 기기에 적합하면서도 95% 이상의 분</p>

	<p>류 성능을 유지하였습니다. 이처럼 “패딩 마스킹”을 통한 RNN 개선과 “모델 구조 최적화”를 결합함으로써, 선행 연구 대비 정확도, 일반화, 효율성, 경량화의 네 가지 측면에서 모두 의미 있는 진전을 이루었습니다.</p> <p>Pose Estimation + Gesture Classification^[3]</p> <p>본 과제는 선행 연구^[3]의 자세 추정 모듈의 무결성을 보장하면서도, 분류기 연산이 전체 시스템 성능에 미치는 영향을 원천 차단하는 '비동기 병렬 파이프라인'을 설계 및 구현하여 시스템의 확장성과 실시간성을 동시에 확보했습니다. 구체적으로, 브로드캐스터 큐에서 수신된 IMU 데이터를 두 개의 독립된 큐(q1, q2)로 복제하여, q1은 기존과 동일하게 칼만 필터 기반의 Pose Estimation에 활용하고 q2는 분류기 입력으로 전달하도록 설계하였습니다. 이로써 자세 추정과 제스처 분류가 완전히 분리된 파이프라인에서 동시에 수행되며, 분류기가 추가되었음에도 Pose Estimation 경로에는 전혀 추가적인 지연이나 연산 부담이 발생하지 않습니다.</p> <p>또한 선행연구 오픈소스 구현에서 간헐적으로 발생하던 연산 병목 현상은 코드 안정화 작업을 통해 완화하였습니다. 주요 병목 구간이던 큐 입출력 로직과 필터 업데이트 루프를 비동기 I/O 처리와 경량화된 데이터 구조로 재구성하여, 전체 처리 지연을 줄이는 성과를 거두었습니다. 이처럼 병렬 파이프라인과 코드 최적화가 결합된 통신 시스템 환경은, 단일 IMU 스트림으로도 자세 추정, 제스처 분류라는 복수의 목적을 효율적으로 달성할 수 있도록 합니다.</p> <p>현장 적용성을 극대화한 고신뢰성 제어 메커니즘 제안</p> <p>기존 연구들이 분류기의 자동 탐지(auto-detection) 성능에 집중한 것과 달리, 본 연구는 오작동이 치명적일 수 있는 소방, 경찰 환경의 특수성을 고려하여 물리 버튼으로 제스처 구간을 명시적으로 제어하는 방식을 제안하고 구현했습니다. 이는 알고리즘의 확률적 판단에만 의존하지 않고, 사용자의 '의도'를 시스템에 명확히 전달하여 신뢰성을 극대화하는 실용적인 접근법을 제시했다는 점에서 차별화된 기여를 가집니다.</p> <p>아이디어의 실증: 완성도 높은 통합 관제 시스템 및 현실적 배포 아키텍처 검증</p> <p>본 연구는 로컬 환경의 기술 검증을 넘어, 실제 클라우드 배포 환경에서 마주할 수 있는 기술적 제약(PaaS 환경의 UDP 수신 불가, WebGL의 wss 보안 정책 등)을 선제적으로 해결하고, RTMP/HLS 스트리밍을 포함한 안정적인 대체 아키텍처를 성공적으로 구축했습니다. 이는 아이디어를 실제 서비스 가능한 수준의 안정적 시스템으로 구현하는 전 과정을 검증했다는 점에서 중요한 실용적 기여를 가집니다.</p>
14. 토론	<p>본 과제는 상용 웨어러블 기기를 이용하여 소방, 경찰 현장에서 사용 가능한 고정밀, 고신뢰성 수신호 통신 시스템 'Vox'를 성공적으로 구현했다는 점에서 의의를 가집니다. 특히, 병렬 처리 파이프라인과 물리 버튼 기반의 명시적 제어 방식을 통해 100%에 달하는 분류 정확도와 실시간 처리 성능을 확보했습니다. 그러나 연구를 진행하며 확인된 몇 가지 한계점과 이를 통해 고찰해 볼 수 있는 향후 발전 방향은 다음과 같습니다.</p> <p>연구의 한계점 및 개선 방향</p>

데이터 일반화의 한계

본 과제의 온라인 학습은 제한된 수의 참가자로부터 수집된 데이터(레이블당 60개)로 수행되었습니다. 비록 100%의 테스트 정확도를 달성했지만, 실제 현장에서 마주할 불특정 다수(연령, 신체조건, 숙련도가 다른)의 소방, 경찰 요원에게도 동일한 성능을 보일지는 추가적인 검증이 필요합니다. 향후 소방, 경찰 학교 등 유관 기관과 협력하여, 실제 현장과 유사한 환경에서 대규모 사용자 데이터를 수집하고, 이를 기반으로 모델의 일반화 성능을 추가적으로 강화해야 합니다.

제스처 확장성의 문제

현재 시스템은 'V, X, O' 3가지의 분별력 높은 제스처만을 대상으로 합니다. 실제 현장에서는 더 복잡하고 다양한 수신호가 필요할 수 있습니다. 제스처의 수가 늘어나고, 동작 간 유사성이 높아질 경우 현재 모델의 분류 성능이 저하될 가능성이 존재합니다. 인식할 수신호의 종류를 10가지 이상으로 확장하고, 유사한 동작들을 구분해내는 강건한 모델(예: Attention 메커니즘을 결합한 모델)에 대한 연구가 필요합니다.

시스템 아키텍처의 확장성 및 안정성

현재 시스템은 단일 EC2 인스턴스에 통신 및 연산 서버가 구성되어 있어, 해당 인스턴스에 장애가 발생할 경우 전체 서비스가 중단될 수 있는 단일 장애점(SPOF, Single Point of Failure)을 내포하고 있습니다. (즉, 서버 한 대에 문제가 생기면 전체 시스템이 멈추는 구조적 취약점입니다.) 또한, 웹 대시보드의 상태 동기화는 1초 주기의 REST API 폴링(Polling) 방식으로 구현되어, 소수 클라이언트 환경에서는 안정적이지만 향후 관제 인원이 증가할 경우 서버 부하를 가중시킬 수 있습니다. 향후에는 AWS의 Auto-Scaling Group과 Network Load Balancer(NLB)를 활용하여 서버를 다중화하고, 상태 동기화 방식 또한 WebSocket이나 SSE(Server-Sent Events)와 같은 서버 푸시(Push) 기반으로 전환하여 시스템의 안정성과 확장성을 확보해야 합니다.

통신 프로토콜과 실시간성(Real-time)의 문제

본 시스템은 3D 시각화 화면 전송을 위해 RTMP/HLS 방식을 채택했습니다. 이 방식은 브라우저 호환성이 높다는 장점이 있지만, 프로토콜 특성상 최소 3~8초의 구조적 지연(latency)이 발생합니다. 따라서 본 시스템이 추구하는 '실시간' 통신은 '준실시간(Near Real-time)'에 가까우며, 긴급 상황 판단의 즉각성을 위해서는 지연 시간을 더욱 단축할 필요가 있습니다. 차후에는 WebRTC나 LL-HLS(Low-Latency HLS)와 같은 최신 저지연 프로토콜을 도입하여 지연 시간을 1초 미만으로 단축시키는 고도화 작업이 요구됩니다.

학술적 및 산업적 시사점

자세-제스처 정보 융합의 가능성

본 시스템은 자세 추정과 제스처 분류를 독립적인 병렬 파이프라인으로 처리했습니다. 하지만, '어떤 자세에서 특정 제스처를 취했는지'에 따라 수신호의 의미가 달라질 수 있습니다(예: 머리 위에서 원을 그리는 것과 허리에서 그리는 것). 추정된 3D 관절 정보를 제스처 분류 모델의 추가 입력(context)으로 활용한다면, 더 풍부하고 정확한 '상황인지 기반' 의도 추정이 가능할 것입니다.

온디바이스 AI(On-Device AI)로의 전환 가능성

현재 시스템은 수집된 데이터를 연산 서버로 전송하여 연산하는 구조입니다. 하지만 CNN+BiLSTM, TCN 등 개발된 모델들의 크기가 매우 작고(각 0.29MB, 0.21MB) 추론 속도가 빠르다는 점은, 통신이

	<p>불안정한 음영 지역에서도 동작할 수 있도록 스마트폰이나 스마트워치 단에서 직접 연산을 수행하는 온디바이스 AI로의 전환 가능성을 시사합니다. 이는 지연 시간을 더욱 단축시키고, 보안을 강화하며, 시스템의 독립성을 높이는 핵심적인 발전 방향이 될 수 있습니다.</p> <p>프로토타이핑과 실제 시스템 간의 기술적 간극</p> <p>본 과제의 Unity 연동 과정은, 로컬 환경에서 검증된 기술(UDP 통신)이라도 실제 웹 배포 환경(WebGL)의 보안 정책(CORS, UDP 차단) 하에서는 동작하지 않을 수 있음을 명확히 보여줍니다. 이는 초기 프로토타이핑 단계부터 최종 배포 환경의 제약을 고려하는 것의 중요성과, 문제 발생 시 WebSocket, RTMP/HLS 등 대안 프로토콜을 유연하게 탐색하고 적용하는 엔지니어링 역량의 가치에 대한 중요한 시사점을 제공합니다.</p> <p>프로덕션 수준 시스템 구축의 복잡성 및 표준 프로토콜의 가치</p> <p>본 과제의 통신 시스템 개발 과정은 로컬 환경에서 완벽하게 동작하던 기술이라도 실제 클라우드 및 웹 브라우저의 보안 정책 하에서는 예상치 못한 문제에 직면할 수 있음을 보여줍니다. 초기 설계에서 AWS Elastic Beanstalk의 구조적 제약으로 인해 UDP 통신이 어려웠던 점 과 Unity WebGL의 wss 프로토콜 호환성 문제 등은 단순히 기능을 구현하는 것을 넘어, 실제 배포 환경의 제약과 표준 프로토콜의 중요성을 명확히 보여주는 사례입니다. 더 나아가 DTLS를 통한 통신 암호화, IaC(Infrastructure as Code)를 통한 배포 자동화, 통합 모니터링 시스템 구축 등은 프로토타입을 안정적인 상용 서비스로 발전시키기 위해 필수적으로 고려되어야 할 사항임을 확인했습니다. 이는 신기술 개발 시 학술적 성과와 더불어 실제적 운영 가능성을 함께 검증하는 것이 얼마나 중요한지에 대한 실질적인 시사점을 제공합니다.</p>
15. 결론	<p>본 과제는 소방 및 경찰 현장에서 기존 무전 통신 시스템이 가진 한계를 극복하고자, 상용 스마트워치를 활용한 실시간 수신호 통신 시스템 "Vox"를 제안하고 성공적으로 구현하였습니다. "Vox"는 스마트워치 IMU 센서로 사용자의 제스처를 인식하고, 이를 중앙 관제 시스템에 실시간으로 전송하여 비언어적 소통을 가능하게 하는 통합 시스템입니다. 데이터 수집부터 병렬 처리 파이프라인, 딥러닝 기반 분류, 다중 요원 원격 모니터링 대시보드에 이르기까지 모든 핵심 기능을 구현하며 아이디어의 실용성을 입증했습니다.</p> <p>핵심 성과</p> <p>본 연구를 통해 달성한 핵심적인 정량적, 정성적 성과는 다음과 같습니다.</p> <p>최고 수준의 분류 정확도 달성: 노이즈가 포함된 외부 테스트 데이터셋에 대해, 주력 모델인 CNN+BiLSTM이 정확도(Accuracy) 1.0000, Macro F1-Score 1.0000을 달성하여 완벽한 분류 성능을 입증했습니다.</p> <p>실시간 처리 성능 확보: 모델의 추론 지연 시간을 0.415ms로 최소화하여, 현장 상황에 즉각적으로 반응해야 하는 시스템의 실시간 요구 조건을 충족시켰습니다.</p> <p>현장 적용성을 고려한 신뢰성 확보: 오작동이 치명적일 수 있는 재난 현장의 특수성을 고려하여, 물리 버튼 기반의 명시적 제어 방식을 도입함으로써 오탐지율 0%를 달성하고 시스템의 신뢰성을 극대화했습니다.</p> <p>경제성 및 실용성 입증: 고가의 전용 장비 없이 상용 스마트워치와 스마트폰만으로 시스템을 구축하여, 추가 도입 비용(CAPEX)이 발생하지 않는 현실적이고 실용적인 솔루션을 제시했습니다.</p>

	<p>완성도 높은 통합 시스템 구축: 데이터 처리, 모델 추론뿐만 아니라 다수 요원을 동시 관제할 수 있는 안정적인 통신 아키텍처와 체계적인 UI/UX를 갖춘 엔드투엔드(End-to-End) 시스템을 완성했습니다.</p> <p>향후 계획 본 연구의 성과를 바탕으로 다음과 같은 후속 연구 및 발전 계획을 추진하고자 합니다.</p> <p>모델 고도화 및 데이터 확장: 소방, 경찰 등 유관 기관과 협력하여 실제 환경에서 대규모 사용자 데이터를 수집하고, 이를 통해 모델의 일반화 성능을 강화할 것이다. 또한, 현장에서 요구되는 수신호의 종류를 10가지 이상으로 확장하고, Attention 메커니즘 등을 도입하여 모델의 표현력을 높일 계획입니다.</p> <p>시스템 아키텍처 강화: 현재 단일 서버 구조가 가진 단일 장애점(SPOF) 문제를 해결하기 위해 AWS Auto-Scaling, Load Balancer 등을 활용한 다중화 구조로 시스템을 개선할 것입니다. 또한, 3D 시각화의 지연 시간을 줄이기 위해 WebRTC와 같은 저지연 프로토콜을 도입하여 실시간성을 더욱 강화할 예정입니다.</p>	
16. 활용 방안	<p>학부생 논문 제출 국내 학회 및 센서 분야 저널에 “스마트워치 기반 실시간 수신호 분류 시스템” 논문을 제출할 계획입니다. 모델 성능, 효율성 비교, 실장치 실험 결과, 파이프라인 구현 세부사항을 포함하여 학술적 검증을 강화합니다.</p> <p>앱 및 오픈소스 공개 현행 스마트폰, 스마트워치 애플리케이션은 선행 연구 개발본을 사용 중이므로, 자체 UI/UX를 갖춘 신규 앱을 별도 개발합니다. 앱 개발 완료 후에는 GitHub에 코드, 모델, 데모 영상을 공개하여, 공공 안전 분야 연구자 및 실무자가 손쉽게 활용할 수 있도록 지원할 예정입니다.</p> <p>현장 파일럿 테스트 소방, 경찰청 등 유관 기관과 협력하여, 제안 시스템을 실제 구조 훈련 및 긴급 대응 시나리오에 적용해 봅니다. 사용자 피드백을 수집하고, UI 개선, 레이블 추가, 모델 튜닝을 통해 완성도를 높인 뒤, 차기 버전으로 배포할 계획입니다.</p> <p>이와 같은 단계적 추진을 통해, 학술적 성과 창출과 함께 실제 현장 적용까지 이어지는 사회적 가치 실현을 목표로 합니다.</p>	
17. 개발 과정	4주차	개발 환경 공부
	5주차	기초 알고리즘 구현 완료, 서버 환경 추가 조사
	6주차	학습 시스템 및 로컬 Unity 구성
	7주차	수신호 레이블 생성 및 전처리
	8주차 ~12주차	모델 생성 및 학습, 다른 모델 적용 비교, 실험 결과 분석, 최적화
	13주차	메인 서버와 연산 서버의 연동 구현, 유니티의 독립 구현

	14주차	일반화 테스트 및 최종 평가
	15주차	최종 발표
18. 역할 분담	이유현(팀장)	통신 환경 및 AI 알고리즘 구현, 제안서, 보고서, PPT, 발표
	유대용	Unity 및 FrontEnd 구현, 노선/예산서 관리, 자료 수집
	백승호	서버 및 웹서비스, BackEnd 구현, 회의록 작성, 자료 수집
19. 참고 문헌	[1] "[단독] 평택 화재 당시 소방 무전 녹취록 단독 입수", <SBS>, 2022.01.18, https://v.daum.net/v/20220118190601489#:~:text=match%20at%20L98%209%EC%8B%9C%201%EB%B6%84%2C,%EB%92%A4%20%EB%8B%A4%EC%8B%9C%20%EB%AC%B4%EC%A0%84%EC%97%90%EC%84%9C%20%EB%93%B1%EC%9E%A5%ED%95%98%EC%A7%80%20%EC%95%8A%EC%8A%B5%EB%8B%88%EB%8B%A4(접속일: 2025.03.16)	
	[2] "Grenfell firefighter describes radio problems when fighting fire", <Inside Housing>, 2018.06.29, https://www.insidehousing.co.uk/news/grenfell-firefighter-describes-radio-problems-when-fighting-fire-57039#:~:text=blaze%2C%20said%20he%20tried%20to,early%20in%20attempts%20at%20evacuation(접속일: 2025.03.16)	
	[3] Fabian C. Weigend, "WearMoCap: multimodal pose tracking for ubiquitous robot control using a smartwatch", frontiers, (2025.01.03)	
	[4] Hamza Sonalcan , Action Recognition in Basketball with Inertial Measurement Unit-Supported Vest, Sensors, (2025.1.19)	
	[5] Lambert R. B., IMU-Based Classification of Locomotion Modes, Transitions, and Gait Phases with Convolutional Recurrent Neural Networks, Sensors, (2022.11.16)	
	[6] "소방 R&D예산 222억원 불과...첨단기술 도움 못받는 소방관들", <연합뉴스>, 2024.02.04, https://www.yna.co.kr/view/AKR20240202146400530(접속일: 2025.03.14)	
	[7] "Number of users of smartwatches worldwide from 2020 to 2029", <statista>, 2025.02.28, https://www.statista.com/forecasts/1314339/worldwide-users-of-smartwatches(접속일: 2025.03.16)	
	[8] "Number of smartphone users worldwide from 2014 to 2029", <statista>, 2025.03.03, https://www.statista.com/forecasts/1143723/smartphone-users-in-the-world(접속일: 2025.03.16)	
	[9] Mingyu Chen, "6DMG: ANew6DMotion Gesture Database", ACM, (2012)	
	[10] Edwin Valarezo Añazco, "HandGesture Recognition Using Single Patchable Six-Axis Inertial Measurement Unit via Recurrent Neural Networks", Sensors, (2021.04.14)	
	[11] "is there an antman style glove button(bluetooth) for triggering the built in horns of EUCs ?", Electric Unicycle Forum, 2023,4,8, https://forum.electricunicycle.org/topic/32690-is-there-an-antman-style-glove-buttonbluetooth-for-triggering-the-built-in-horns-of-eucs/(접속일: 2025.06.09)	