게임콘텐츠 캡스톤디자인

최종 보고서

한채연 최성우 황해연 이해송

실시간 수어 인식 기반 텍스트 출력 시스템

01

프로젝트 개요

02

프로젝트 수행방법

03

수행 결과

04

기대 효과 및 활용 방안

05

결론 및 제언

프로젝트 개요

1

01 | 과제 설계 배경 및 필요성

- 능동적 커뮤니케이션 수단으로서의 수어 인식 기능 필요성

"수어 기반 채팅 기능이 존재한다면 시각적으로 즉각 반응이 가능해져 훨씬 몰입도 높은 플레이가 가능할 것"

청각장애, 언어장애 게이머

채팅 시스템과의 통합

AI 기반 인식 - 텍스트 변환

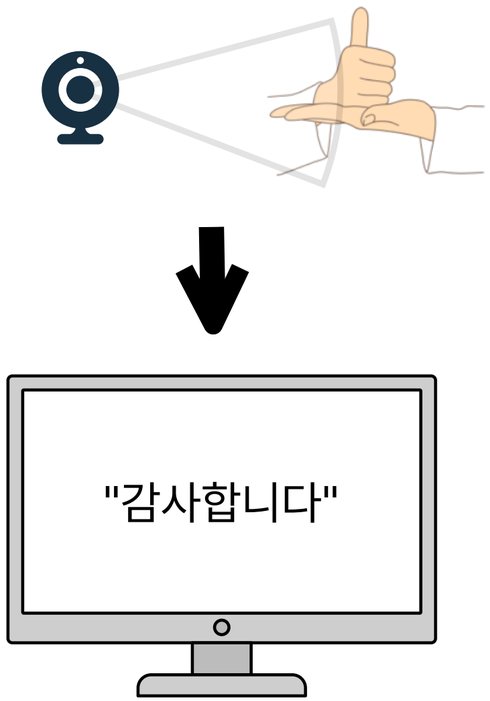
01 | 과제 주요 내용

1. 영상 기반 수어 데이터 전처리

2. 딥러닝 기반 시계열 모델 (LSTM, GRU, LSTM + GRU, Transformer) 구축 & 비교

3. 고성능 모델로 실시간 시스템 구현

01 | 최종 결과물 목표



1. 사용자의 수어 동작 → 웹캠 인식

2. Mediapipe 랜드마크 추출 → 시계열 데이터로 변환

3. 모델의 동작 예측 → 자연어 형태의 텍스트로 변환

과제 수행 방법

2

02 | 과제 수행 방법

AI Hub KETI 데이터 전처리

train+val data:

AI Hub의 수어 데이터셋(0~3000번 영상) 중 사전에 선정한
100개 단어 라벨에 해당하는 영상만 선별 -> 한 단어당 7개, 총 700개 영상

Test data:

AI Hub의 수어 데이터셋(3001~6000번 영상) 중 사전에 선정한
100개 단어 라벨에 해당하는 영상만 선별
-> 한 단어당 3개, 총 300개 영상

Mediapipe 랜드마크 추출

1. Holistic 모델 활용

→ 자세(pose), 얼굴(face), 양손(left hand, right hand)의 랜드마크 정보
543개의 랜드마크 (543×3 = 1629차원 벡터)를 사용

2. MediaPipe를 이용한 키포인트 추출

→ 각 영상마다 .npy파일에 저장

3. 시퀀스 데이터 구성

4. 데이터 정제

→ 모델 학습할 수 있는 입력 꼴로 바꾸기

02 | 과제 수행 방법

WLASL 데이터 전처리

train / val / test data:

WLASL(Word-Level American Sign Language) 데이터셋 사용
WLASL 데이터셋 중 가장 빈번하게 등장하는 상위 50개 단어를 추출
클래스당 균형있는 데이터 분포를 위해 클래스별 동일한 수의 샘플을 선택

1. 엑셀 파일(train/val/test)을 읽어 클래스별 메타데이터 로드
2. 각 영상 파일을 OpenCV로 읽어 프레임 단위 처리
3. MediaPipe Holistic 모델을 적용하여 프레임별 키포인트 추출
4. 추출 실패 시 예외 처리 및 실패 로그 기록
5. 추출된 키포인트를 .npy 포맷으로 클래스별 디렉토리에 저장

Mediapipe 랜드마크 추출

1. Holistic 모델 활용

→ 자세(pose), 얼굴(face), 양손(left hand, right hand)의 랜드마크 정보
543개의 랜드마크 (543×3 = 1629차원 벡터)를 사용

2. MediaPipe를 이용한 키포인트 추출

→ 각 영상마다 .npy파일에 저장

3. 시퀀스 데이터 구성

4. 데이터 정제

→ 모델 학습할 수 있는 입력 꼴로 바꾸기

2 | LSTM

최종 모델

전처리

- Jitterimg
- padding & Masking
- Selective Augmentation
- Z-axis Rotation
- Selective Best Feature Selection

2 | LSTM

최종 모델

모델 구조

- Bidirectional LSTM
- Residual Connections
- Layer Normalization & Dropout
- Global Average Pooling
- Label Smoothing + Learning Rate Scheduling

2 | LSTM

최종 모델

최종 성능

Keti

Test 정확도: 70%

WLASL

Test 정확도: 60.3%

2 | GRU

최종 모델

데이터 전처리

- Padding
- train / validation data split
- RandomOverSampling
- PCA 적용 : feature 값 1000

데이터 증강

- Jittering
- Time Masking
- Time Warping
- Time Shift

```python
# ========================
# Data Augmentation Functions
# ========================
def jitter(x, sigma=0.01):
    return x + np.random.normal(loc=0.0, scale=sigma, size=x.shape)

def time_shift(x, shift_max=5):
    shift = np.random.randint(-shift_max, shift_max)
    if shift > 0:
        return np.pad(x, ((shift, 0), (0, 0)), mode='constant')[:x.shape[0]]
    elif shift < 0:
        return np.pad(x, ((0, -shift), (0, 0)), mode='constant')[:x.shape[0]]
    else:
        return x

def time_warp(x, scale_range=(0.8, 1.2)):
    scale = np.random.uniform(*scale_range)
    orig_len = x.shape[0]
    new_len = max(int(orig_len * scale), 2)
    new_indices = np.linspace(0, orig_len - 1, new_len)
    orig_indices = np.arange(orig_len)
    x_new = np.zeros_like(x)
    for i in range(x.shape[1]):
        interp_signal = np.interp(new_indices, orig_indices, x[:, i])
        x_new[:, i] = np.interp(orig_indices, new_indices, interp_signal)
    return x_new
```

[데이터 증강 코드]

2 | GRU

최종 모델

모델 구조

- 3단 Bi-Directional GRU
- 평균 풀링 (Temporal Mean Pooling)
- MLP 분류
- Regularization 및 학습 전략

학습 조건

- Adam Optimizer
- Focal Loss
- CosineAnnealingLR
- Early Stopping

```python
# ========================
# Model Definition
# ========================
class GRUModel(torch.nn.Module):
    def __init__(self, input_size, num_classes):
        super(GRUModel, self).__init__()
        self.gru1 = nn.GRU(input_size, 128, batch_first=True, bidirectional=True)
        self.gru2 = nn.GRU(128*2, 64, batch_first=True, bidirectional=True)
        self.gru3 = nn.GRU(64*2, 32, batch_first=True, bidirectional=True)
        self.dropout = nn.Dropout(0.3)
        self.bn1 = nn.BatchNorm1d(32*2)
        self.fc1 = nn.Linear(32*2, 512)
        self.elu1 = nn.ELU()
        self.bn2 = nn.BatchNorm1d(512)
        self.fc2 = nn.Linear(512, num_classes)
```

[GRU 구조 코드]

2 | LSTM + GRU

최종 모델

1. 데이터 증강, jittering, time masking, gaussian noise 등 추가

2. 모델구조 개선 -Attention, Bidirectional LSTM 등 추가

3. 시퀀스 길이 60으로 조정하고 샘플링 적용

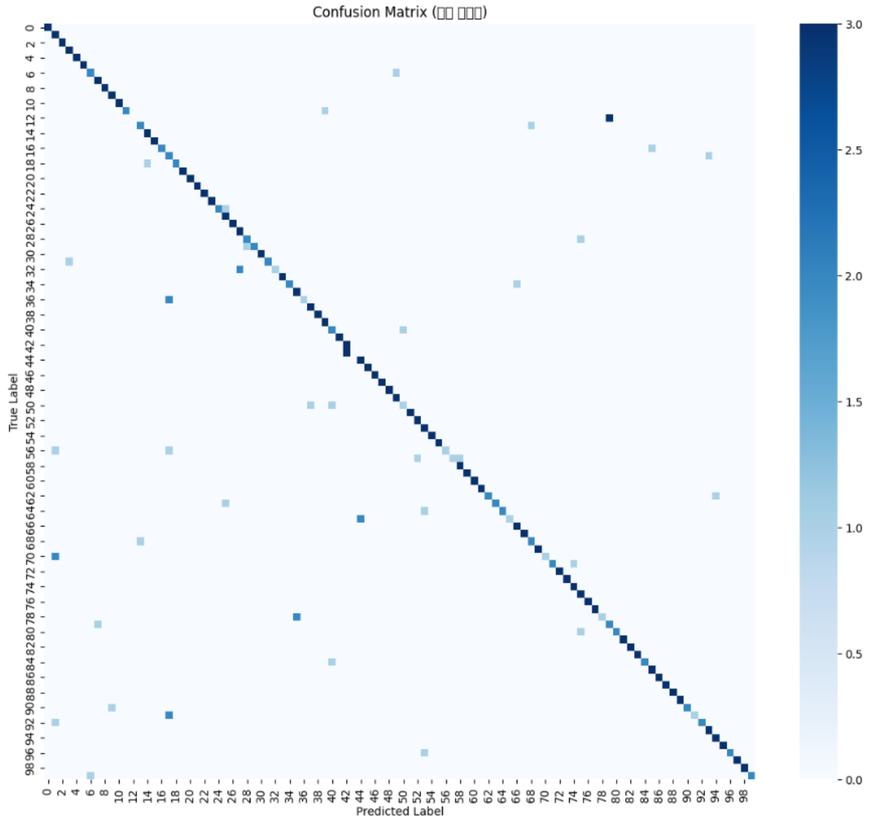
4. PCA 적용

5. 하이퍼파라미터 튜닝

```
input_shape = (60, 450)
model_input = Input(shape=input_shape)

x = Bidirectional(LSTM(units=best_hp.get('lstm_units'), return_sequences=True))(model_input)
x = BatchNormalization()(x)
x = Dropout(best_hp.get('lstm_dropout'))(x)

x = GRU(units=best_hp.get('gru_units'), return_sequences=True)(x)
x = BatchNormalization()(x)
x = Dropout(best_hp.get('gru_dropout'))(x)

x = Attention()([x, x])
x = GlobalAveragePooling1D()(x)

x = Dense(units=best_hp.get('dense_units'), activation='relu')(x)
x = Dropout(0.3)(x)
output = Dense(num_classes, activation='softmax')(x)

model = Model(model_input, output)
```

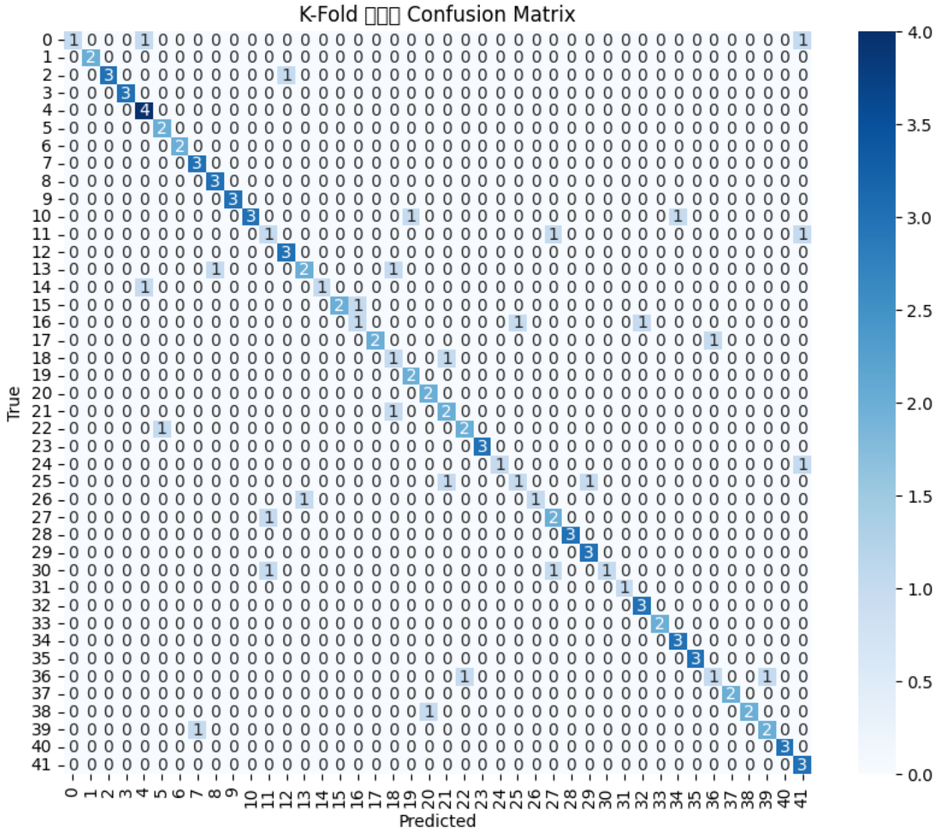
2 | LSTM + GRU

KETI



Test accuracy: 0.84

WLASL



Test accuracy: 0.69.

4 Transformer

성능 개선 시도	Test Accuracy
1. 기본 모델 초기 학습	20.84%
2. 시퀀스 길이 60 프레임 고정 + 손실 로깅	33.48%
3. 데이터 증강(노이즈, 반전, 마스킹, 회전) + 정규화 + 병합(merged)	64.33%
4. 특징 선택(LightGBM) + 프레임 균등화	82.00%

4 Transformer

성능 개선 시도	Test Accuracy
5.wlasl에 적용	53.85%
6. 차원 축소 최적화(PCA)	64.34%

4 Transformer

최종 모델

입력 차원: 500 (top-k 또는 PCA 적용 후)

TransformerEncoder 층 수: 2~4층 사용

시간 정보 통합 방식: Attention Pooling 사용

손실 함수: Label Smoothing이 적용된 CrossEntropyLoss

러닝레이트 스케줄러: CosineAnnealingLR 사용

조기 종료 조건: EarlyStopping (patience=25)

4 Transformer

최종 모델 - 하이퍼파라미터

KETI

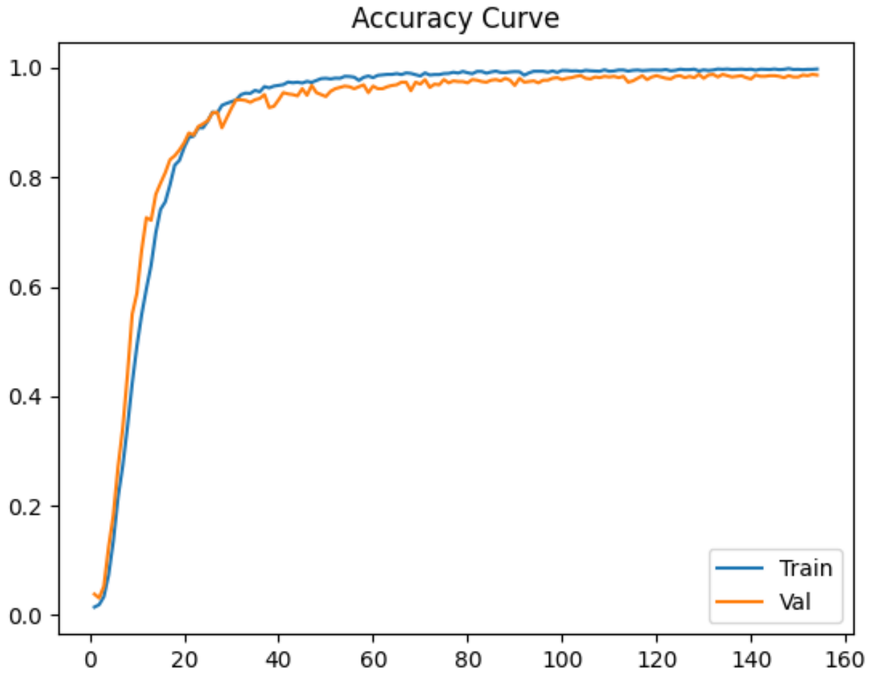
batch_size = 32
num_classes = 100
epochs = 250
input_dim = 500 # LightGBM 기반 top-500 특성
d_model = 256
nhead = 2
num_layers = 2
dropout = 0.3
label_smoothing = 0.1
lr = 1e-4
scheduler = CosineAnnealingLR
feature_selection = LightGBM top-k

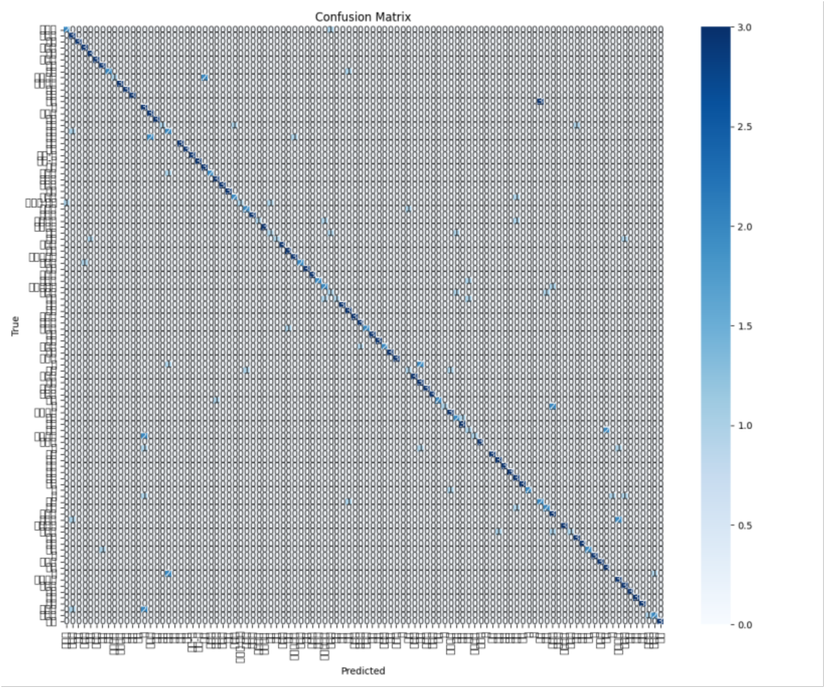
WLASL

batch_size = 동일
num_classes = 동일
epochs = 동일
input_dim = 500 # PCA 기반 500차원
d_model = 512
nhead = 8
num_layers = 4
dropout = 동일
label_smoothing = 동일
lr = 동일
scheduler = 동일
feature_selection = PCA

4 Transformer

학습 결과 - KETI 82%

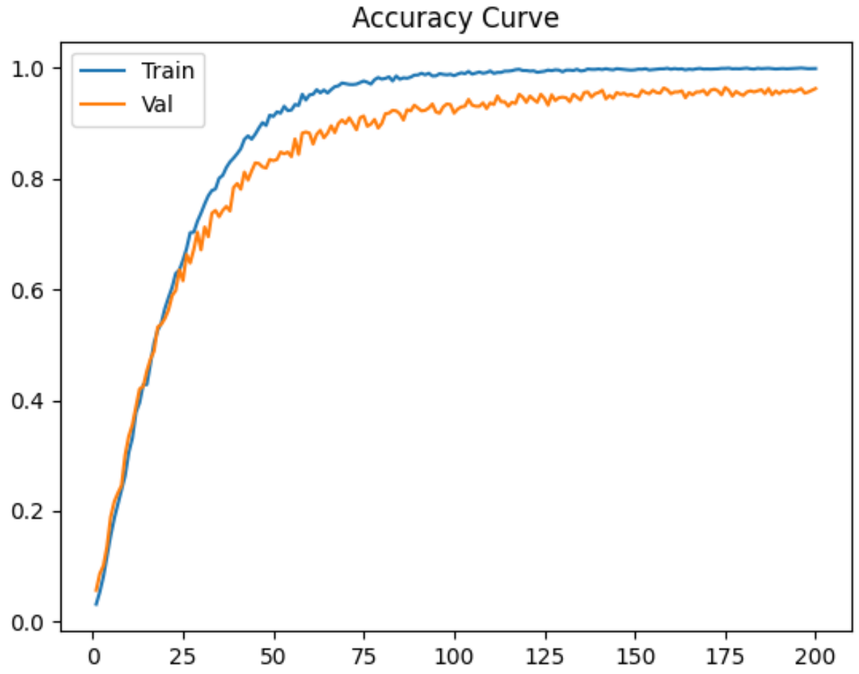


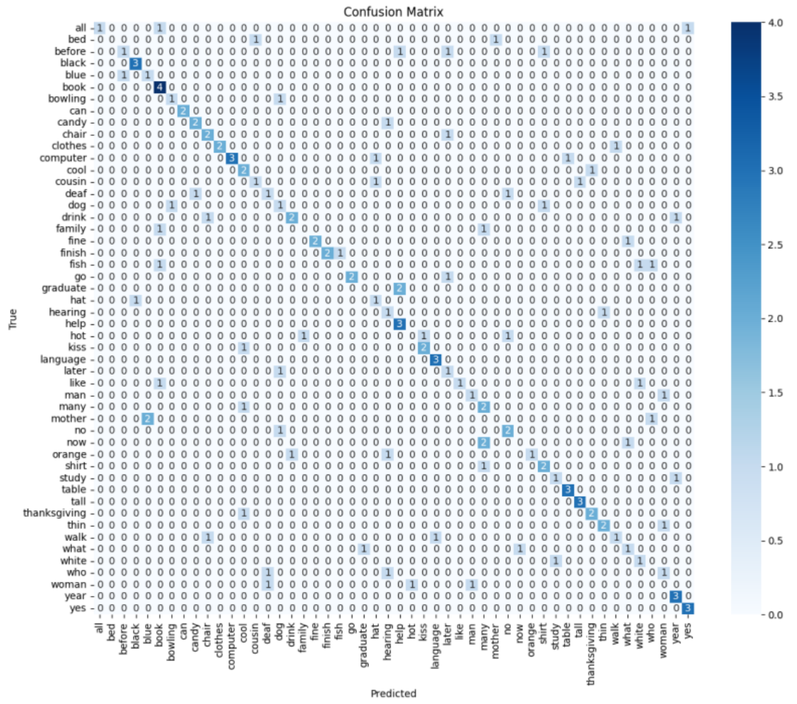


4 Transformer

학습 결과 - WLASL

64.34%





2 | 가상 환경 세팅 후 모델 성능 비교

가상 환경 세팅 목적

데이터셋

실행 환경

조건 통일

- 버전으로 인한 충돌을 방지
- 모델 간의 공정한 성능 비교

가상 환경 설정

GPU : NVIDIA Geforce RTX 4060
tensorflow-gpu : 2.10.0
python : 3.8.20
torch : 1.12 + cu113
mediapipe : 0.10.5

수행 결과

3

3 가상 환경에서의 모델별 성능 비교

- tensorflow, pytorch의 GPU 연동 및 mediapipe 와 버전충돌 문제를 해결 하여 하나의 가상환경 구축

	LSTM	GRU	LSTM+GRU	Transformer
KETI.	68%	61%	85%	82%
WLASL	60.87%	65%	68.64%	64.34%

3 실시간 수어인식 시스템 구현 : LSTM + GRU

LSTM + GRU

정확히 예측 성공한 사례

예측 확률
70% ~ 90%

'코', '곰', '화상', '홍수', 'book', 'go', 'yes'

- 프레임이 누적됨에 따라 예측되는 단어도 일부 존재
- 시퀀스 전체 흐름과 반복 패턴을 기반으로 동작 해석

유사 단어 혼동 사례

화재' -> "화상', '곰' -> '창백하다', ' shirt' -> 'clothes'

- 손 모양이 유사하고 움직임의 차이가 미세한 단어
- 손 모양의 변화보다는 위치나 방향성에 더 의존하는 수어의 경우 자주 발생
- 현재 사용 중인 좌표 기반 특징 벡터만으로는 미묘한 의미 차이를 판별하기 어려운 한계를 시사

3 실시간 수어인식 시스템 구현 : LSTM + GRU

LSTM + GRU

예측 실패 사례 — 예측 불가능

'파도', '파편', 'graduate', 'hearing' 등

- 해당 단어의 수어 동작이 상대적으로 빠르거나 짧아 충분한 프레임 누적이 이루어지지 못했을 가능성
- 수어 숙지의 미숙함으로 인해 수어 수행자의 동작이 학습 데이터에서의 평균적인 동작과 속도나 손 위치 등에서 크게 벗어났을 가능성
- 실시간 촬영 환경의 조명, 손 위치, 배경 등이 학습 데이터와 달라 특징 불일치가 발생했을 가능성

PCA 적용 여부에 따른 정확도 — 대부분 PCA 적용 > PCA 미적용

'화재', '화장실', '호흡곤란'

- PCA를 적용하지 않은 경우 예측되지 않거나 낮은 확률로 잘못 인식되었지만, PCA를 적용한 모델에서는 70~80% 수준의 정확도로 올바르게 인식
- PCA가 고차원 입력에서 불필요한 노이즈와 변동성을 제거하고, 핵심 정보만 보존함으로써 실시간 인식 환경에서도 모델의 일반화 성능을 향상시킨 것으로 해석
- 실시간 환경에서는 다양한 외부 요인이 정확도에 영향을 미치며, 정규화 및 PCA와 같은 전처리 기법의 유무가 모델 성능에 실질적인 영향을 준다

3 실시간 수어인식 시스템 구현 : Transformer

Transformer

특정 클래스 단어만 인식

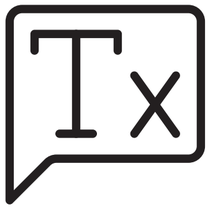
- 데이터 증강 방식의 간섭 가능성: 좌우 반전(flip_horizontal)은 수어의 의미를 변경할 수 있으며, 10도 회전을 적용한 증강(rotate_xy)은 MediaPipe의 정규화 좌표계 상에서 실제 측면 동작을 효과적으로 모사하지 못하고, 오히려 비자연적인 왜곡을 유발할 가능성이 있다.
- Top-k 특성 선택의 샘플 커버리지 부족: 현재의 특성 선택은 학습 데이터 중 상위 500개 샘플만을 기반으로 수행되었으며, 이로 인해 고빈도 클래스에 대한 과적합이 발생하고 저빈도 클래스에 대한 표현력이 저하되었을 가능성이 존재한다.
- 학습 데이터와 실제 카메라 입력 간 분포 불일치: 촬영 각도, 손의 거리, 배경 복잡도, 동작 속도, 조명 변화 등의 요인이 MediaPipe의 랜드마크 추출 품질에 영향을 미치며, 이로 인해 모델 입력의 통계적 분포가 달라져 전체적인 인식 정확도가 저하될 수 있다.

3 최종 결과물 주요 특징 및 설명

LSTM + GRU

TRANSFROMER

수어 동작 감지



텍스트 출력

- MediaPipe Holistic 기반 543개 랜드마크 추출
- PCA 기반 차원 축소로 학습 효율 향상
- Bidirectional LSTM + GRU 기반 시퀀스 분류 모델
- 데이터 증강 및 정규화 기법 활용
- 통합 가상환경 기반 테스트
- 실시간 수어 인식 및 텍스트 출력 UI 구현

기대 효과 및 활용 방안

4

04 | 기대 효과

1. 청각 장애인이 디지털 환경에서 능동적으로 의사소통에 참여할 수 있는 기반 마련

2. 손동작 입력 방식의 비접촉적 및 직관적 인터페이스

3. 비장애인 사용자에게도 새로운 커뮤니케이션 도구로 기능

4. 차세대 입력 방식으로서의 가능성

04 활용 방안

1. 전략 기반 멀티플레이어 게임에서 실시간 의사소통

2. 장애 유무와 무관한 공정한 협업 환경 제공

3. 다양한 플랫폼에서의 실버 케어, 헬스 케어 등의 분야

4. 인터랙션 방식의 혁신

결론 및 제언

5

05 | 결론 및 제언

	LSTM	GRU	LSTM + GRU	Transformer
KETI 정확도	68%	61%	85%	82%
WLASL정확도	60.87%	65%	68.64%	64.34%

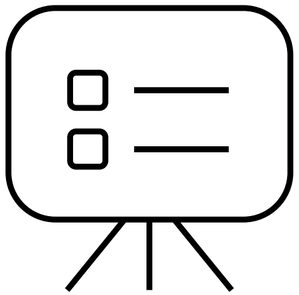
- 최종 LSTM + GRU 모델의 최고 정확도 기록

Transformer 모델 역시 강력한 self-attention 구조를 바탕으로 높은 정확도(82%)를 기록하였으나
학습 데이터가 상대적으로 적은 상황에서는 LSTM+GRU의 일반화 성능이 더욱 안정적
Bi-LSTM의 장기 의존성 처리 능력과 GRU의 경량화 특성을 결합하고,
Attention 메커니즘과 GlobalAveragePooling을 추가함으로써
시퀀스 내 중요한 프레임 정보에 집중할 수 있는 구조가 성능 향상에 크게 기여

- PCA 적용 모델의 성능 우수 확인
- 시퀀스 누적 기반 예측 방식의 효과 확인

05 | 결론 및 제언

Project



실시간 수어 인식 기반
텍스트 출력 시스템

1. 동적 시퀀스를 기반 실제 사용자와 상호작용 가능한 수준의 실시간 시스템으로 발전

2. 수어 기반 인터페이스의 실시간성·정확성·응용 가능성 입증

3. 향후 게임 산업을 포함한 다양한 산업 및 사회적 약자 지원 기술로의 확장 가능성

참고 자료

06 참고 자료

논문

[1] K. Navendu and V. Sahula, “Word Level Sign Language Recognition using MediaPipe and LSTM-GRU Network,” *iSeS 2024: International Symposium on Embedded Systems*, Jaipur, India, 2024.

[2] Vaishnavi Mishra, Tatheer Fatima, Shreshtha Srivastava, and Komal Asrani, “Advancement in Sign Language Recognition Technologies: A Comprehensive Review,” International Journal of Innovative Research in Computer Science & Technology (IJIRCST), vol. 12, special issue-1, pp. 40–44, Mar. 2024. doi: 10.55524/CSISTW.2024.12.1.7.

[3] D. Kim, Y. Jung, and H. Lee, “KSL-Guide: 안드로이드 기반 한국어 수어 학습 애플리케이션 개발,” in *Proceedings of the Korea Computer Congress (KCC)*, pp. 302–304, Jun. 2021.

[4] [3] J. Kim and J. H. Park, “미디어파이프와 장단기기억을 이용한 수화 동작인식 앱 개발,” *디지털콘텐츠학회논문지*, vol. 24, no. 1, pp. 111–117, Jan. 2023, doi: 10.9728/dcs.2023.24.1.111.

[5] Barathi Subramanian, Bekhzod Olimov, Shraddha M.Naik, Sangchul Kim, Kil-Houm Park and Jeonghong Kim, “An integrated mediapipe-optimized GRU model for Indian sign language recognition”

Git

[6]https://github.com/Ghoney99/hearing_impaired_helper_make_model/tree/main/sing_lang_trans/modules

[7] https://github.com/Dev1nW/Sign_language_recognition_final_project

THANK YOU