실어증 및 편마비 환자의 의사소통 보조를 위한 CNN 기반 표정 인식 모델 연구

- 문제 인식

실어증 환자

언어상실증
뇌의 특정 영역이 손상된 후 언어에 대한 이해나 표현이 안 되는 장애

편마비 환자

몸의 한쪽이 마비되는 증상
표정을 짓을 때 얼굴 한쪽이 마비됨

환자의 의사소통 보조를 위한 표정 감정 인식이 필요함

- 목적

Eye-Tracking + CNN Emotion

개인 보고서에서 영감받았다고 구라
편마비 및 실어증 환자의 감정(Emotion)뿐만 아니라, **"무엇을 보면서"** 그런 감정을 느끼는지(Gaze) 아는 것은 매우 중요

최종 목표

편마비 + 실어증 강조

1. 얼굴 랜드마크(`dlib` 사용)에서 눈동자의 위치를 추정하여 시선 방향을 계산
2. "시선이 **간병인**을 향할 때" + "표정이 **'공포'** 일 때" => '도움 요청'으로 해석
3. "시선이 **음식**을 향할 때" + "표정이 **'혐오'** 일 때" => '다른 메뉴 원함'으로 해석
4. 이렇게 '감정'과 '시선'을 조합하여 훨씬 더 풍부하고 정확한 의사소통 보조 도구를 만들 수 있음

현재 목표

핑계 살짝
실제 편마비 환자 사진이 있었으면 좋겠지만, 없어서 대신
한쪽 얼굴 데이터로(마스킹 데이터 셋) CNN을 돌렸을 때 얼마정도의 Accuracy가 나오는지 비교

Model 1: Custom CNN

Model 2: Azure Face API

Accuracy 비교

1. Data Gathering

- **목표:** System 1(직접 학습)과 System 2(Azure API)에 **완벽히 동일한 입력 데이터**를 제공.
- **데이터 소스:** FER2013 원본 (48x48 픽셀 이미지).
- **통합 전처리 (마스킹 데이터셋 구축):**
  1. FER2013의 모든 48x48 이미지
  2. 이 이미지들의 **한쪽 절반(예: 오른쪽 24픽셀)을 검은색 사각형으로 칠함(마스킹).** (발표는 절반 오른쪽 절반 왼쪽으로 하고 모두 왼쪽으로 대칭이동 했다고 구라)
  3. 이 '마스킹된 48x48 이미지'를 **하나의 통일된 데이터셋**으로 만듦(Model 2 데이터셋에 맞추기 위해서)
  4. 이 데이터셋을 **학습(Train) / 검증(Validation) / 테스트(Test)** 용으로 분할

2. Model 1: Custom CNN

- **Data Preprocessing**
  - **정규화 (Normalization):** 위에서 만든 '마스킹 데이터셋'의 픽셀 값을 0~255에서 0~1 사이로 정규화합니다.
- **Training:**
  - **환경:** PyTorch
  - **모델:** 전이 학습(Transfer Learning) 기반 **ResNet-18**
  - **입력: 48x48 마스킹 이미지**를 그대로 사용. (모델 구조를 수정할 필요 없이, CNN이 스스로 검은 영역을 무시하도록 학습함)
  - **수정:** 마지막 FC Layer만 7가지 감정 클래스로 교체.
  - **손실 함수 (Loss):** `nn.CrossEntropyLoss`
  - **옵티마이저 (Optimizer):** `Adam`
- **Prediction Results Analysis:**
  - **'마스킹된 테스트(Test)셋'**으로 성능을 평가
  - **Accuracy (정확도)** 계산
  - **Confusion Matrix (혼동 행렬)** 분석 (모델이 어떤 감정을 헷갈려하는지)
  - **Softmax** 함수를 적용하여 '행복 90%' 같은 확률값 도출

3. Model 2: Azure Face API

- **Data (Test):**
  - **Model 1에서 사용한 것과 동일한 '마스킹된 테스트(Test)셋'**을 사용
- **Prediction (예측):**
  - 이 **48x48 마스킹 이미지**를 Azure Face API로 전송하여 감정 예측값
- **Prediction Results Analysis:**
  - API가 반환한 예측값과 실제 정답을 비교하여 **Accuracy (정확도)**를 계산

4. Final

- **Model 1**의 Accuracy와 **Model 2 (Azure API)**의 Accuracy를 **직접 비교**
- **결론 도출:** "동일하게 한쪽 얼굴이 가려진 조건에서, (A) 마스킹된 데이터로 직접 학습시킨 내 모델과 (B) 거대 데이터로 학습된 범용 Azure API 중 어떤 것이 편마비 환자의 표정을 더 잘 인식했는가?"
- 코딩 실패하면 구라

5. 향후 계획 및 고도화 방안

1. 실제 원본 사진 적용 및 얼굴 정렬 (Face Alignment)

- **필요성:** 지금은 FER2013이라는, 이미 정면을 보도록 잘린 48x48 픽셀의 '실험실용' 데이터를 사용. 하지만 실제 환경에서는 카메라의 각도, 사람의 고개 돌림, 얼굴 크기가 모두 다름.
- **실행 방안:**
  1. CK+, AffectNet 또는 직접 촬영한 **고해상도 원본 사진(Real-world data)**을 사용
  2. `dlib` 나 `MTCNN` 을 사용해 이미지에서 **얼굴을 먼저 검출(Detection)**
  3. 검출된 얼굴에서 **눈, 코, 입 등의 주요 랜드마크(Facial Landmarks)** 를 찾음
  4. 이 랜드마크(주로 두 눈)를 기준으로 이미지를 **회전시키고(rotate) 크기를 조절(scale)하여** 모든 얼굴이 동일한 구도(예: 정면)를 보도록 **표준화(Alignment)**
  5. 표준화된 얼굴 이미지를 일정한 크기로 잘라낸(Crop) 뒤, **여기에 '마스킹(Masking)' 처리를 적용**

2. 데이터 증강 (Data Augmentation)

- **필요성:** '한쪽이 가려진' 마스킹 데이터로만 학습. 데이터의 양이 부족하거나 다양성이 떨어지면, 모델이 학습 데이터에만 과적합(Overfitting)되어 처음 보는 사진에는 성능이 급격히 떨어짐
- **실행 방안:**
  - 학습 데이터를 모델에 넣기 직전에 **랜덤하게 변형**
  - **적용 기법:**
    - **밝기 및 대비 조절:** 다양한 조명 환경에 대응
    - **약간의 회전 및 이동:** 카메라가 살짝 삐뚤어지거나 얼굴이 중앙에 있지 않아도 잘 인식

3. 실시간 웹캠 적용 및 UI 구현

- **필요성:** 최종 목표는 실어증/편마비 환자분들을 돕는 것. 폴더의 사진을 분석하는 것을 넘어, **실시간 영상**에서 바로 표정을 읽어야 함
- **실행 방안:**
  1. `OpenCV` 라이브러리를 사용해 웹캠/카메라 영상을 프레임 단위로 불러옴
  2. **매 프레임마다** 위에서 언급한 `얼굴 검출` -> `정렬` -> `마스킹` -> `표정 예측` 과정을 실시간으로 수행합니다. or YOLO 이용. 그냥 다 쑤셔놓음

3. 환자의 마비된 쪽(왼쪽/오른쪽)을 선택할 수 있는 **간단한 UI**를 만듭니다. (선택에 따라 마스킹 위치가 결정됨)
4. 예측된 감정(예: '행복 90%')을 화면에 실시간으로 표시

4. 아이트래킹(시선 추적) 데이터와 융합

'감정'과 '시선'을 조합하여 훨씬 더 풍부하고 정확한 의사소통 보조 도구를 만들 수 있음