AI 기반 졸음 감지 예측모델 개발 및 시각화

CNN 딥러닝 모델과 LLM을 활용한
운전자 졸음 상태 실시간 감지 시스템

발표자: 김재혁

졸음운전 사고 예방을 위한 실시간 AI 모니터링의 필요성

운전 중 졸음은 전체 교통사고의 약 **20~30%**를 차지하는 치명적인 요인임. 단순한 주의 환기를 넘어 실시간으로 운전자의 상태를 파악하는 기술적 대응이 절실한 상황.

99%

목표 감지 정확도
실시간 얼굴 분석 시스템

본 프로젝트는 딥러닝을 통해 도로 위 안전을 확보하고 사고율을 획기적으로 낮추는 것을 목표로 합니다.

개발 프로세스

Step 01

데이터 수집 및 구축

Driver Drowsiness Dataset(DDD)을 활용한 6,000장 고해상도 이미지 확보

Step 02

데이터 전처리 및 증강

이미지 리사이징, 정규화 및 증강 기법 적용으로 모델 강인성 확보

Step 03

CNN 모델 설계 및 학습

4단계 합성곱 블록 기반 아키텍처와 Adam 옵티마이저로 학습

Step 04

평가 및 프로토타입 구현

정확도 분석 및 웹 기반 실시간 졸음 감지 인터페이스 시연

Dataset 구성

6,000
Total Images

1:1
Class Balance

200px
Input Resolution

Driver Drowsiness Dataset (DDD)

Drowsy와 Non Drowsy 클래스를 각각 3,000장씩 균등하게 배분되어 모델의 편향성을 최소화하고 학습의 안정성 확보.

명확한 시각적 특징 추출

눈의 개폐 정도, 고개의 각도 변화, 얼굴 근육의 이완 등 졸음 상태를 정의하는 핵심적인 시각적 지표들이 포함되어 CNN 학습에 최적화.

학습 최적화를 위한 데이터 처리 (코드)

이미지 샘플링

```python
# 데이터 샘플링
MAX_SAMPLES_PER_CLASS = 3000
import random
import shutil

if len(drowsy_files) > MAX_SAMPLES_PER_CLASS:
    random.seed(123)
    drowsy_files = random.sample(drowsy_files, MAX_SAMPLES_PER_CLASS)
    print(f"샘플링된 Drowsy 이미지 개수: {len(drowsy_files)}")

if len(non_drowsy_files) > MAX_SAMPLES_PER_CLASS:
    random.seed(123)
    non_drowsy_files = random.sample(non_drowsy_files, MAX_SAMPLES_PER_CLASS)
    print(f"샘플링된 Non Drowsy 이미지 개수: {len(non_drowsy_files)}")
```

데이터 증강 **(Augmentation)**

```python
# 데이터 증강 레이어 정의
data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.18),
    layers.RandomZoom(0.18),
    layers.RandomBrightness(0.15),
    layers.RandomContrast(0.1),
])
```

학습 최적화를 위한 데이터 처리

기본 데이터 전처리

이미지 리사이징 (Resizing)

모든 입력 이미지를 **200x200** 크기로 표준화하여 모델 연산 효율 최적화

픽셀 값 정규화 (Normalization)

0~255 범위의 픽셀 값을 **0~1** 사이로 스케일링하여 학습 수렴 속도 향상

데이터 분할 (Split)

학습 데이터(80%)와 검증 데이터(20%)로 분리하여 과적합 여부 실시간 모니터링

데이터 증강 (Augmentation)

기하학적 변환

Random Flip(좌우 반전) 및 Rotation(±15도)을 통해 다양한 운전자 자세 및 각도 학습

광학적 변환

Random Brightness 및 Contrast 조절을 통해 주간, 야간, 터널 등 다양한 조명 환경 대응

공간적 변환

Random Zoom(±5%) 적용으로 카메라와 운전자 사이의 거리 변화에 대한 강인성 확보

CNN 모델 구조

```python
model = tf.keras.Sequential([
    data_augmentation,
    layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),

    # 합성곱 블록
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(128, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(192, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.5),

    # Fully Connected Layer
    layers.Flatten(),
    layers.Dense(192, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.4),

    # 출력층 (이진 분류: 졸음/정상)
    layers.Dense(num_classes, activation='softmax')
```

Feature Extraction

4-Stage Conv Blocks

Conv2D와 MaxPooling이 결합된 4개의 블록을 통해 이미지의 저수준부터 고수준 특징까지 단계적으로 추출.

Classification

Dense Layers

Flatten 이후 192, 128 유닛의 Dense 레이어를 배치하여 추출된 특징을 기반으로 최종 졸음 여부 분류.

Regularization

Dropout Strategy

각 단계에 0.4~0.5의 Dropout을 적용하여 과적합을 방지하고, 실제 환경에서의 일반화 성능 극대화.

학습 전략 (컴파일)

```
# 모델 컴파일 (학습률 조정으로 과적합 방지)
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(),
    metrics=['accuracy']
)
```

Optimizer

Adam Optimizer

Learning Rate: 0.0001

낮은 학습률 설정을 통해 가중치를 정밀하게 업데이트하며 학습의 안정성 확보

Loss Function

Sparse Categorical Crossentropy

다중 클래스 분류 문제에 최적화된 손실 함수를 사용하여 예측값과 실제값 사이의 오차 최소화

학습 전략 (콜백 함수)

```python
model_dir = os.path.abspath('./model')
os.makedirs(model_dir, exist_ok=True)
model_path = os.path.join(model_dir, 'sleepy_detection_model_best.keras')

callbacks = [
    tf.keras.callbacks.EarlyStopping(
        monitor='val_loss',
        patience=3,
        restore_best_weights=True,
        verbose=1,
        min_delta=0.001  # 최소 개선량 설정
    ),
    tf.keras.callbacks.ModelCheckpoint(
        filepath=model_path,
        monitor='val_loss',
        save_best_only=True,  # 가장 좋은 모델만 저장
        mode='min',  # val_loss가 최소일 때 저장
        verbose=1  # 저장될 때 로그 출력
    )
]
```

Callback 02

Model Checkpoint

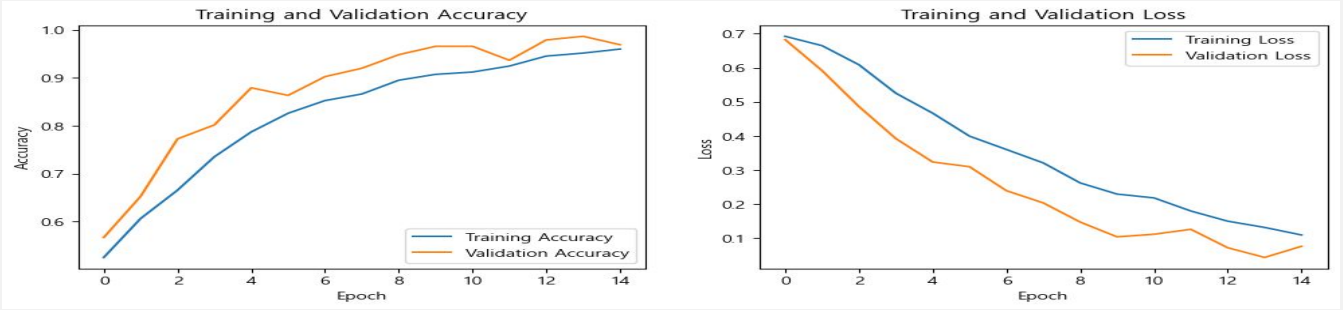
학습 과정 중 검증 손실이 가장 낮은 시점의 최적 가중치를 자동으로 저장하여 **최고 성능 모델** 확보

Callback 01

Early Stopping

검증 손실(val_loss)이 3 Epoch 동안 개선되지 않을 경우 학습을 조기 종료하여 **과적합 (Overfitting)** 방지

Accuracy & Loss



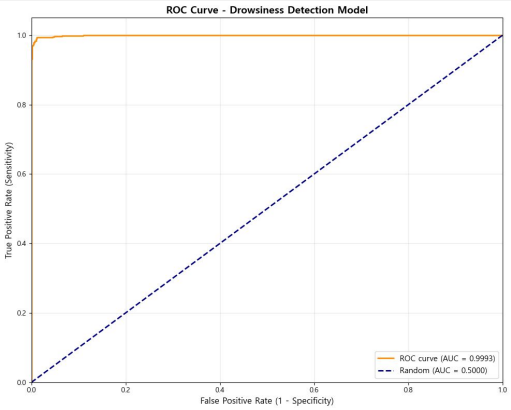
신속하고 안정적인 모델 수렴

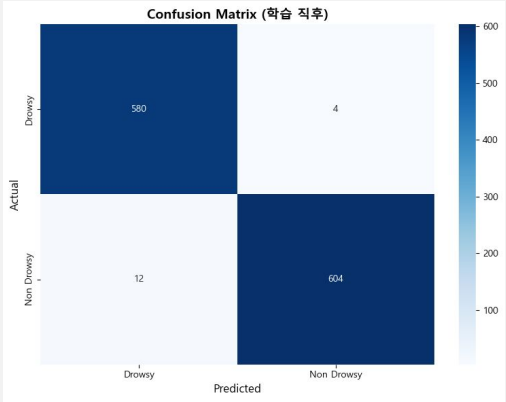
Adam 옵티마이저와 정밀한 학습률 설정을 통해 약 **15 Epoch** 내에 손실 함수가 안정적으로 최소값에 도달하며 수렴

과적합 없는 일반화 성능

Training과 Validation 정확도 곡선이 매우 밀접하게 일치하는 양상을 보이며, 모델이 학습 데이터에 매몰되지 않고 **핵심 특징을 정확히 학습**했음을 확인

ROC Curve & Confusion Matrix





ROC Curve 분석

AUC(Area Under the Curve) 값이 **0.9993**에 도달했다는 것은 모델이 졸음(Drowsy)과 정상(Non Drowsy) 상태를 구분하는 변별력이 거의 완벽함을 의미

Confusion Matrix 검증

1,200개의 테스트 샘플 중 오분류 사례가 **단 20건 미만**으로 나타났습니다. 이는 실제 주행 환경에서도 매우 낮은 오경보율과 높은 신뢰도를 보장할 수 있음을 시사

LLM 프롬프트

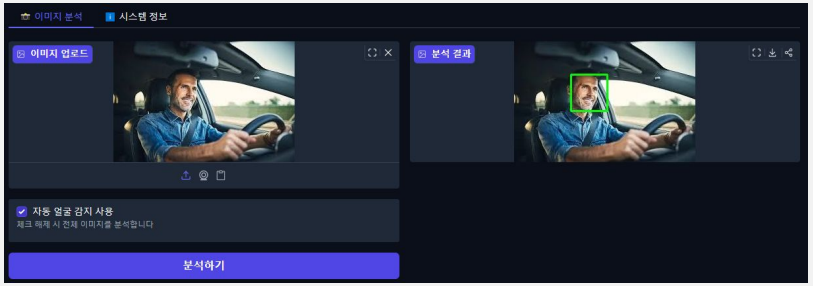
결과 설명

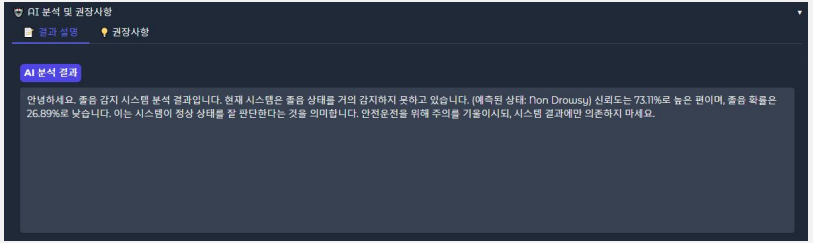
```
    prompt = f"""당신은 졸음 감지 시스템의 결과를 사용자에게 친절하게 설명하는 AI 어시스턴트입니다.

분석 결과:
- 예측된 상태: {predicted_class}
- 신뢰도: {confidence:.2f}%
- 졸음 확률: {drowsy_prob:.2f}%
- 정상 확률: {non_drowsy_prob:.2f}%

다음 내용을 포함하여 한국어로 간결하고 이해하기 쉽게 설명해주세요:
1. 분석 결과 요약 (2-3문장)
2. 신뢰도에 대한 평가
3. 결과의 의미

주의사항:
- 전문적이지만 이해하기 쉬운 언어 사용
- 200자 이내로 간결하게 작성
- 사용자에게 도움이 되는 정보 제공
"""
```

권장사항

```
    prompt = f"""당신은 운전자 안전 전문가입니다. 졸음 감지 시스템의 결과를 바탕으로 실용적인 권장사

분석 결과:
- 예측된 상태: {predicted_class}
- 신뢰도: {confidence:.2f}%
- 졸음 확률: {drowsy_prob:.2f}%

다음 내용을 포함하여 한국어로 작성해주세요:
1. 현재 상태에 대한 평가
2. 즉시 취해야 할 조치 (3-5개)
3. 장기적인 건강 관리 팁 (2-3개)

주의사항:
- 실용적이고 실행 가능한 조치 제시
- 안전을 최우선으로 고려
- 300자 이내로 작성
- 긍정적이고 격려하는 톤 사용
"""
```

프로토 타이핑





Input Process

실시간 이미지 업로드

사용자가 웹 인터페이스를 통해 얼굴 이미지를 업로드하면 시스템이 즉각적으로 데이터 수신

AI Inference

CNN 모델 실시간 추론

서버에 탑재된 최적화된 CNN 모델이 입력된 이미지를 분석하여 졸음 여부 판정

Output Result

직관적인 결과 시각화

'정상' 또는 '졸음' 판정 결과와 함께 예측 신뢰도(%)를 시각적으로 표시하여 사용자 편의성 상승

AI Report

LLM을 통한 설명 및 권장사항

현재 상태를 분석하고, 즉시 취해야할 조치에 대해 설명

기대 효과 및 활용 방안

Safety

교통 안전 향상 및 사고율 감소

첨단 운전자 보조 시스템(ADAS)에 통합되어 **졸음운전 사고를 사전에 예방**하고, 실시간 경고 시스템을 통해 운전자의 주의를 환기시켜 도로 위 인명 피해 최소화

Industry

물류 및 운송업계 모니터링 솔루션

장거리 운행이 빈번한 **화물차 및 버스 운전자**를 위한 전용 모니터링 시스템으로 활용 가능하며, 운전자의 피로도를 실시간 관리하여 기업의 안전 관리 효율 극대화

Scalability

기술 고도화 및 서비스 확장성

단순 졸음 감지를 넘어 **시선 추적, 스마트폰 사용 감지** 등 다양한 부주의 상태 분석으로 확장 가능하며, IoT 기기와의 연동을 통해 지능형 차량 제어 시스템으로의 발전 가능성 상승

개선 사항 및 소감

개선 사항

CNN + LSTM

LSTM과의 결합

차량 내부 영상인 Yawdd 사용을 통해 시계열 방식인 LSTM와 결합해 학습하여 더욱 정확하고 실용적인 실시간 졸음 감지

Multi Class

다중 클래스 확장

졸음 단계별 분류 및 다른 위험 상태(산만, 스마트폰 사용) 감지 추가하여 실용성 상승

Embedded

임베디드 기반 시스템

Rasberry Pi / Arduino 등을 이용하여 운전자가 실시간으로 졸음 여부를 판단하고 대처할 수 있도록 도움

소감

딥러닝에 대한 이해가 충분하지 않은 상태에서 프로젝트를 시작했기 때문에 초반에는 개념 하나하나가 어렵고 막막하게 느껴졌습니다. 모델 구조, 학습 과정과 같은 요소들이 단순히 코드가 아니라 어떤 의미를 가지는지 이해하는 데에도 많은 시간이 필요했습니다.
그럼에도 불구하고 사회문제와 연관된 주제를 선정하면서부터 프로젝트에 대한 흥미와 책임감이 생겼습니다.

비록 영상 분석과 같은 보다 정교한 기능까지 구현하지는 못했지만, 데이터 기반으로 문제를 정의하고 딥러닝 모델을 적용해 해결을 시도해 보았다는 경험 자체가 매우 값졌다고 생각합니다. 단순히 코드를 따라 치는 수준을 넘어, 전체적인 딥러닝 프로젝트의 구조와 흐름을 이해하게 된 것이 가장 큰 수확이라고 생각합니다.