종합설계 프로젝트 보고서

온디바이스 딥러닝 기반 실시간 안저 질환 분류 시스템

2025.05.26

한성대학교

전자트랙/시스템반도체트랙

학번 김주영

학번 박찬기

학번 이재강

종합설계 프로젝트 보고서

온디바이스 딥러닝 기반 실시간 안저 질환 분류 시스템

작성자: 김주영, 박찬기, 이재강
지도교수: 김준민

차 례

I. 프로젝트 개요

1. 개발 배경

나이관련 황반변성(AMD), 당뇨병성 망막병증(DR), 녹내장(Glaucoma) 등 주요 안구 질환은 조기에 발견하지 않으면 영구적인 시력 손실이나 실명으로 이어질 수 있어 정기적인 검진과 신속한 진단이 필수적이다. 특히 최근 고령화 사회로 진입하면서 이와 같은 안구 질환의 유병률이 빠르게 증가하고 있으며, 이에 따라 의료 접근성과 진단 효율성에 대한 사회적 요구도 높아지고 있다.
하지만 기존의 안과 진단은 고가의 전문 의료 장비와 숙련된 의료진의 판독을 필요로 하며, 특히 의료 인프라가 부족한 지역이나 1차 진료 환경에서는 활용에 한계가 있다. 또한 병원을 방문하지 않고 스스로 안구 건강 상태를 확인할 수 있는 방안이 부족하여 조기 진단의 기회를 놓치는 사례도 적지 않다.
이러한 배경 속에서 본 프로젝트는 인공지능 기술과 모바일 디바이스의 결합을 통해 문제를 해결하고자 하였다. 스마트폰을 활용해 사용자가 직접 안저(fundus) 이미지를 촬영하고, 이를 온디바이스(On-device) 환경에서 실시간 분석하여 안구 질환을 분류하는 경량형 딥러닝 시스템을 개발하였다. 본 시스템은 나이관련 황반변성(AMD), 당뇨병성 망막병증(DR), 녹내장(Glaucoma), 정상(Normal) 총 4가지 상태를 판별할 수 있으며, 클라우드 서버나 인터넷 연결 없이도 단독으로 진단이 가능하여 오프라인 환경에서도 유용하게 사용될 수 있다.
모델은 ResNet-18 기반의 분류 모델을 PyTorch Lite 형태로 변환하여 모바일 디바이스에서 직접 추론이 가능하도록 적용하였고, 이미지 전처리는 OpenCV 기반 알고리즘을 사용하여 모바일 환경에 최적화되었다. 또한 모델의 성능을 검증하기 위해 대조군으로 MobileNetV2 및 EfficientNet-B0 기반의 분류 모델과 비교 실험을 진행하였다. MobileNetV2와 EfficientNet-B0는 각각 대표적인 경량화 모델로, 정확도와 추론 속도 측면에서 ResNet-18과의 차이를 비교함으로써, 다양한 모바일 환경에서의 최적 모델 구성을 평가할 수 있도록 하였다. 이로써 고가 장비 없이도 높은 진단 정확도를 달성할 수 있으며, 특히 의료 사각지대나 응급의료 현장에서 효과적인 진단 도구로 활용될 가능성을 지닌다.

2. 기대 효과

본 프로젝트에서 개발한 스마트폰 기반 온 디바이스 안구 질환 분류 시스템은 고가의 전문 장비나 클라우드 서버에 의존하지 않고, 오직 스마트폰만으로도 안구 질환의 진단이 가능하다는 점에서 높은 실용성과 확장성을 지닌다. 특히 의료 인프라가 부족한 농어촌, 개발도상국, 응급 현장, 재난 지역 등과 같이 전문 의료 장비나 인력이 부족한 환경에서도 효과적으로 활용될 수 있어, 전 세계적으로 심화되고 있는 의료 접근성 문제 해결에 기여할 수 있다.

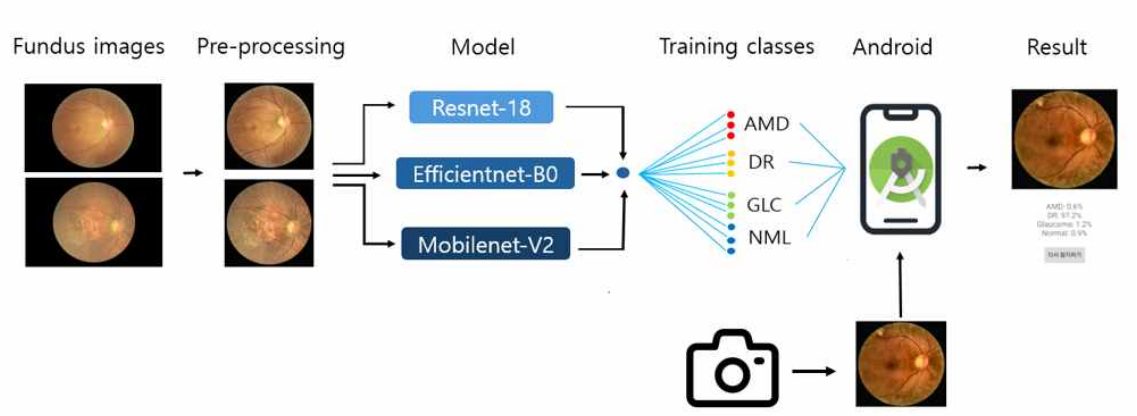
또한, 본 시스템은 인터넷 연결이 불가능한 오프라인 환경에서도 온전히 작동하며, 고정밀 안저 이미지를 실시간으로 분류할 수 있는 기능을 제공함으로써, 기존의 웹 기반 진단 시스템이나 병원 중심 진단 체계가 갖는 시간적·공간적 제약을 극복할 수 있다. 이러한 점은 특히 응급 상황이나 현장 중심의 1차 진료 체계에서 유용하며, 진단 대기 시간을 단축하고 조기 진단을 가능하게 함으로써 질병의 악화를 방지하는 데 실질적인 도움을 줄 수 있다.

기술적으로는 PyTorch 기반의 딥러닝 모델을 모바일 환경에 최적화한 형태(Pytorch Lite)로 변환하고, 이미지 전처리에 OpenCV 기반의 알고리즘을 적용함으로써, 모바일 디바이스 내에서 경량화된 고성능 AI 모델이 안정적으로 구동될 수 있음을 입증하였다. 이는 향후 피부암, 호흡기 질환, 구강 질환 등 다양한 의료 분야에서의 온디바이스 진단 시스템 개발로 이어질 수 있으며, 모바일 헬스케어 산업 전반의 기술적 발전을 견인할 수 있다.

또한 본 시스템은 자가 진단 도구로서도 활용 가능성이 높다. 사용자는 병원을 방문하지 않고 자신의 안구 상태를 1차적으로 점검할 수 있으며, 이상 징후가 발견될 경우 빠르게 전문 진료를 받을 수 있어, 조기 진단과 의료 자원 분산에 긍정적인 효과를 기대할 수 있다.

II. 시스템 구성

1. 전체 시스템 구성도



2. 시스템 구성 과정

(1) 데이터 셋

본 연구에서는 총 4,800장의 안저 이미지를 활용하여 안구 질환 분류 모델을 학습하였다. 이 중 약 2,800장은 공개 의료 영상 데이터셋인 ODIR-5K(Ocular Disease Intelligent Recognition, 2019)에서 수집하였으며, 나머지 약 2,000장은 스마트폰 카메라를 이용해 직접 촬영한 실제 환경 기반의 이미지로 구성하였다. ODIR-5K 데이터셋은 안과 전문 병원에서 수집된 안저 이미지로, 각 이미지에 대해 병리학적 라벨이 함께 제공되어 의료 AI 연구에 널리 활용되고 있다. 해당 데이터는 ODIR-5K 공식 웹사이트를 통해 제공되며, 본 프로젝트에서는 공개 라이선스에 따라 비상업적 연구 목적으로 사용되었다.
이미지 분류는 총 4개 클래스(AMD, Diabetic Retinopathy, Glaucoma, Normal)로 구성하였디. 클래스 간 데이터 분포는 균형 있게 구성하였으며, 학습 시 실제 환경을 반영하기 위해 스마트폰 촬영 이미지 비중을 40% 이상 포함시켜 모델의 일반화 성능을 확보하였다.

(2) 학습 모델 설계 및 변환

본 프로젝트에서는 안구 질환 분류를 위한 딥러닝 기반 모델로 ResNet-18을 사용하였다. PyTorch의 torchvision.models 라이브러리를 통해 ImageNet 사전학습 가중치가 적용된 ResNet-18 모델을 불러온 후, 분류기에 해당하는 fully connected layer를 커스터마이징하여 총 4개 클래스(AMD, DR, Glaucoma, Normal)에 맞도록 구조를 수정하였다. 마지막 분류 계층에는 Dropout, BatchNorm, ReLU 등을 적용하여 일반화 성능을 향상시켰다.
학습 데이터셋은 Google Colab 환경에서 구성하였으며, 안저 이미지에 대해 CLAHE 기반 명암 대비 강화, 원형 안저 검출 및 크롭, 정사각형 패딩, Resize(224×224), Normalize 등의 전처리 과정을 포함하였다. 추가적으로 ColorJitter, RandomRotation, GaussianBlur, Cutout 등의 다양한 데이터 증강 기법을 적용하여 학습 데이터 다양성을 높였다.
학습은 CrossEntropyLoss를 손실 함수로 사용하고, Adam 옵티마이저와 ReduceLROnPlateau 스케줄러를 통해 학습 안정성을 확보하였다. 학습은 최대 50 epoch 동안 수행되었으며, 검증 정확도가 가장 높은 epoch에서의 모델 가중치를 저장하였다.
학습이 완료된 모델은 PyTorch의 TorchScript 기능을 통해 .pt 파일로 변환되었고, 이후 optimize_for_mobile() 함수를 통해 PyTorch Lite 형식인 .ptl 파일로 경량화 및 최적화하였다. 이 과정을 통해 .pth, .pt, .ptl 파일이 모두 생성되었으며, 그 중 .ptl 파일은 Android 앱 내 온 디바이스 추론에 사용되었다.

(3) 안드로이드 앱 환경 구성

모바일 환경에서 학습된 모델을 온디바이스로 실행하기 위해 Android Studio를 기반으로 전용 앱을 개발하였다. 앱은 사용자가 직접 안저 이미지를 촬영하거나, 갤러리에서 이미지를 선택한 후, 해당 이미지를 딥러닝 모델로 분석하는 구조로 설계되었다. 전체 앱은 MainActivity1과 MainActivity2로 구성되며, 그 중 MainActivity1은 사용자의 입력(촬영 또는 이미지 선택)을 처리하고 다음 액티비티로 이미지를 전달하는 역할을 담당한다.
카메라 촬영 기능은 Android의 권한 요청 시스템을 활용하여 Manifest.permission.CAMERA 권한을 동적으로 확인하고, 사용자가 허용한 경우 내장 카메라를 실행하여 이미지를 촬영한다. 촬영된 이미지는 내부 저장소에 임시 파일로 저장되며, 이 경로는 FileProvider를 통해 다음 액티비티로 전달된다. 촬영 직후에는 이미지의 해상도 및 용량을 최적화하기 위해 BitmapFactory

를 통해 이미지를 불러온 뒤, EXIF 메타데이터를 분석하여 회전 상태를 보정하고, 가로 해상도가 1000픽셀을 초과할 경우 자동으로 리사이징 및 JPEG 압축을 수행한다. 이러한 최적화는 모델 추론의 정확성과 속도 모두에 기여한다.
또한 갤러리에서 이미지를 선택할 수 있는 기능도 함께 구현되었으며, 사용자는 Intent.ACTION_PICK을 통해 외부 저장소에서 원하는 이미지를 선택할 수 있다. 선택된 이미지는 InputStream으로 불러와 Bitmap 객체로 변환되고, 임시 파일로 저장된 후 MainActivity2로 전달된다. 이후 앱은 해당 이미지를 딥러닝 모델로 분석하여 결과를 시각적으로 출력한다.
OpenCV 라이브러리는 앱 내 이미지 전처리를 위해 초기화되며,
OpenCVLoader.initDebug()를 통해 OpenCV의 로딩 상태를 확인한다. 전체적인 UI는 activity_main.xml 파일에 정의된 버튼과 이미지 뷰로 구성되며, 기본적으로 로고 이미지가 배치되고, 진단 시작 및 갤러리 열기 버튼이 제공된다.
이와 같은 앱 구조는 사용자가 복잡한 절차 없이도 안구 질환 이미지를 손쉽게 입력하고, 모델이 예측한 결과를 실시간으로 확인할 수 있도록 설계되었으며,
전체 프로세스는 네트워크 연결 없이 온 디바이스 환경에서 단독으로 수행된다.

(4) 이미지 전처리 및 추론 기능 구현

MainActivity2는 촬영되거나 갤러리에서 선택된 안저 이미지를 딥러닝 모델에 입력하여 질환을 분류하고 결과를 사용자에게 시각적으로 출력하는 역할을 수행한다. 앱이 실행되면 assets 폴더에 저장된 PyTorch Lite 형식의 .ptl 모델을 LiteModuleLoader.load() 함수를 통해 로드한다. 모델 로드 후, MainActivity1에서 전달된 이미지 경로를 기반으로 이미지를 불러오고, EXIF 메타데이터를 제거한 뒤 cropFundusCircle() 함수를 통해 전처리를 수행한다.
전처리는 OpenCV 기반으로 구현되었으며, 입력 이미지를 Lab 색공간으로 변환한 뒤 CLAHE(Contrast Limited Adaptive Histogram Equalization)를 적용하여 명암 대비를 향상시킨다. 이후 그레이스케일 변환과 HoughCircles 검출을 통해 원형 안저 영역을 자동으로 탐지하고, 해당 영역을 중심으로 크롭한다. 크롭된 이미지는 정사각형 패딩과 Resize(224×224)를 거쳐 딥러닝 모델의 입력 크기에 맞게 변환된다.
이렇게 전처리된 이미지는 TensorImageUtils.bitmapToFloat32Tensor() 함수를 통해 FloatTensor 형식으로 변환되며, 이후 모델에 입력되어 추론이 수행된다. 추론 결과는 4개 클래스(AMD, Diabetic Retinopathy, Glaucoma, Normal)에 대한 소프트맥스 확률값으로 출력되며, 사용자는 이 결과를 UI 상의 TextView를

통해 확인할 수 있다. 추론 중에는 ProgressBar를 통해 분석 진행 상태가 표시되며, 결과 출력 후에는 재시작 버튼을 통해 새로운 이미지를 입력받을 수 있도록 구성되어 있다.
또한 디버깅 및 테스트 목적으로 전처리된 이미지를 내부 저장소에 저장하는 기능도 구현되었으며, OpenCV 기반 전처리 파이프라인의 출력 결과를 MediaStore를 통해 확인할 수 있도록 하였다.

(5) 사용자 인터페이스 구성

MainActivity1은 사용자로부터 입력을 받는 메인 진입 화면으로, 카메라를 통해 직접 안저 이미지를 촬영하거나, 갤러리에서 기존 이미지를 선택할 수 있는 두 가지 입력 방식을 제공한다. 해당 화면은 ConstraintLayout을 기반으로 구성되었으며, 직관적인 이미지 중심 UI와 함께 사용자 친화적인 버튼 배치로 설계되었다.
상단에는 ImageView를 통해 Retina 진단 시스템의 로고(retina_logo)가 표시되며, 이는 사용자에게 앱의 기능과 목적을 직관적으로 인지시키는 시각적 요소로 작용한다. 로고 하단에는 앱 설명 또는 진단 관련 안내 문구를 표시하기 위한 TextView가 배치되어 있으며, 향후 기능 확장을 고려하여 동적으로 텍스트를 설정할 수 있도록 구성되었다.
그 아래에는 두 개의 핵심 버튼이 배치되어 있다. start_detection_button은 사용자가 실시간으로 스마트폰 카메라를 통해 안저 사진을 직접 촬영할 수 있도록 하며, open_gallery_button은 사용자가 갤러리에서 기존의 이미지를 선택해 불러올 수 있도록 한다. 두 버튼 모두 앱 내에서 이미지 입력을 담당하는 핵심 인터페이스 요소로, 탐지 기능으로 연결되는 진입점이다.
화면 하단에는 프로젝트 식별 및 출처 표기를 위한 요소로써, 학교 로고와 모델명을 나타내는 TextView가 함께 배치되어 있다. 이 구역은 디자인 일관성과 보고서용 데모 환경을 고려해 구성되었으며, 프로젝트의 전문성과 신뢰성을 시각적으로 강조하는 역할을 한다.
안구 이미지의 분석 결과를 사용자에게 시각적으로 제공하기 위해, MainActivity2의 레이아웃은 직관적이고 간결한 UI로 구성되었다. 해당 화면은 ConstraintLayout을 기반으로 구성되었으며, 중심에는 진단에 사용된 안저 이미지, 그 아래로 진단 결과 텍스트, 진행 상태를 표시하는 로딩 스피너, 그리고 재시작 버튼이 배치되어 있다.
먼저, ImageView는 촬영 혹은 선택된 안저 이미지를 표시하며, 사용자가 어떤

이미지를 분석했는지를 직접 확인할 수 있도록 한다. 이 뷰는 centerCrop 속성을 통해 이미지 중심이 정확하게 보이도록 설정되었다.
ProgressBar는 이미지 분석이 진행되는 동안 로딩 상태를 나타내며, 분석 시작 시에는 보이고 분석 완료 후에는 자동으로 숨겨진다. 이 과정을 통해 사용자는 앱이 작동 중임을 인지할 수 있고, 분석 결과를 기다리는 동안 불필요한 혼란을 방지할 수 있다.
진단 결과는 TextView에 출력되며, 4개의 클래스(AMD, DR, Glaucoma, Normal)에 대한 확률이 퍼센트 형식으로 표시된다. 결과는 실시간으로 모델의 추론 결과와 연결되어 업데이트된다.
마지막으로 Button 요소는 "다시 탐지하기" 기능을 제공하며, 진단이 완료된 이후에만 활성화되어 사용자가 새로운 이미지를 입력할 수 있도록 유도한다. 이 버튼은 앱의 재사용성과 흐름 제어를 위한 핵심적인 인터페이스 요소로 작동한다.
이와 같이 구성된 사용자 인터페이스는 간단하면서도 사용자가 진단 과정을 직관적으로 이해할 수 있도록 설계되었으며, 앱의 목적에 부합하는 기능적이고 실용적인 구조를 갖춘다.

(6) 사용자 인터페이스 초기화 및 스플래시 화면 구성

앱의 사용자 경험을 향상시키기 위해, 본 프로젝트에서는 앱 실행 시 로딩 상태를 표시하고 팀 로고 이미지를 노출하는 스플래시 화면을 구현하였다. SplashActivity는 애플리케이션 실행 직후 최초로 호출되는 액티비티로, 일정 시간 동안 전체 화면 이미지를 표시한 후 자동으로 진단 화면(MainActivity1)으로 전환되도록 구성되어 있다.
스플래시 화면은 상단의 상태바 및 하단 내비게이션 바를 제거한 전체 화면 모드로 표시되며, View.SYSTEM_UI_FLAG_FULLSCREE 및 SYSTEM_UI_FLAG_HIDE_NAVIGATION 플래그를 설정하여 몰입형 사용자 경험을 제공한다. 또한 getSupportActionBar().hide()를 통해 액션바도 제거함으로써, 로고 이미지 또는 브랜드 요소가 화면 중앙에 집중되도록 설계하였다.
스플래시 화면은 약 3초간 유지되며, Handler를 사용해 일정 시간이 지난 후 자동으로 MainActivity1로 전환되도록 구현되었다. 사용자는 별도의 조작 없이도 자연스럽게 다음 화면으로 이동하게 되며, 앱 실행 과정의 심리적 대기 시간을 부드럽게 처리하는 역할을 한다.
이러한 스플래시 화면 구성은 모바일 앱 개발에서의 기본적인 UX 구성 요소로

서, 사용자의 몰입도 향상과 앱의 완성도 제고에 기여한다.

(7)스플래시 화면 레이아웃 구성

앱의 초기 실행 시 사용자에게 팀 로고의 아이덴티티를 명확히 전달하고, 로딩 과정을 자연스럽게 연결하기 위해 SplashActivity의 레이아웃은 단일 이미지 중심의 간결한 구조로 설계되었다. 해당 레이아웃은 RelativeLayout을 기반으로 하며, 전체 배경은 학교 공식 색상인 짙은 파란색 계열의 #005BAC으로 설정되었다.
화면 중앙에는 ImageView 요소가 배치되어 있으며, 앱의 로고 이미지가 표시된다. 로고는 중심 정렬을 통해 사용자 시선의 집중도를 높이고 앱의 정체성을 직관적으로 전달할 수 있도록 구성되었다.

(8) Android 앱 환경 설정

본 프로젝트의 안드로이드 애플리케이션은 PyTorch Lite 모델과 OpenCV를 활용하여 온디바이스에서 안구 질환 이미지를 실시간으로 분류할 수 있도록 개발되었으며, 이를 위해 build.gradle(:app) 파일 내에서 관련 환경 설정을 체계적으로 구성하였다.
애플리케이션의 최소 SDK 버전은 26(Android 8.0)으로 설정하였으며, 최신 디바이스와의 호환성을 고려하여 targetSdk는 35로 지정하였다. 또한 Java 11 버전을 명시적으로 설정함으로써 최신 문법 및 라이브러리 호환성을 확보하였다.
PyTorch 모델을 온디바이스에서 추론하기 위해 org .pytorch:pytorch_android_lite:1.13.1 및 pytorch_android_torchvision_lite:1.13.1 라이브러리를 추가하였다. 이 라이브러리는 PyTorch에서 학습된 .ptl 모델을 안드로이드 환경에서 경량 추론이 가능하도록 지원하며, TorchVision 기반 이미지 처리에도 필요한 연산 모듈을 포함하고 있다.
이미지 전처리에 필수적인 OpenCV 기능은 프로젝트 내 별도 모듈로 구성된 OpenCVLibrary411을 통해 연결되었으며, JNI를 통한 네이티브 코드 연동을 위해 jniLibs 디렉토리도 명시적으로 sourceSets에 포함되었다. 또한 안드로이드 기기의 이미지 EXIF 정보를 활용하여 자동 회전 처리를 수행하기 위해 androidx.exifinterface 라이브러리도 함께 추가되었다.

출시 버전에 대해서는 ProGuard를 통한 코드 난독화 및 리소스 압축 기능을 활성화하여 앱 용량을 최소화하고 보안성을 높였다. 리소스 인식 오류 방지를 위해 sourceSets 항목에서 Java, 리소스, 매니페스트, jniLibs의 경로를 수동으로 지정하여 R 클래스 오류를 방지하였다.
이와 같이 Gradle 설정은 앱이 안정적으로 작동할 수 있도록 모델 추론, 이미지 처리, UI 구성에 필요한 모든 외부 라이브러리 및 설정 요소를 통합하는 기반 역할을 수행하였다.

(9) OpenCV 라이브러리 통합 구성

안구 이미지의 원형 크롭, 대비 향상, 필터링 등 전처리 과정에 OpenCV의 다양한 컴퓨터 비전 기능을 활용하기 위해, 본 프로젝트에서는 OpenCV를 별도의 Android 라이브러리 모듈(OpenCVLibrary411)로 구성하여 앱에 통합하였다. 이를 통해 메인 앱 모듈과 명확히 분리된 구조에서 OpenCV 기능을 안정적으로 불러올 수 있도록 하였다.
OpenCVLibrary411 모듈의 build.gradle 파일은 com.android.library 플러그인을 적용하여 라이브러리 모듈로 선언되어 있으며, Android SDK 버전은 메인 앱과 동일하게 compileSdkVersion 35, targetSdkVersion 35로 설정하였다. 최소 지원 버전은 Android 5.0으로 지정되어 다양한 기기와의 호환성을 확보하였다.
소스셋은 OpenCV의 Java 코드, 리소스, 매니페스트 경로를 수동으로 명시함으로써 Android Studio가 관련 파일을 정확히 인식하고 빌드 과정에서 통합될 수 있도록 하였다. 이를 통해 메인 앱 모듈에서는 OpenCV 모듈을 직접 종속성으로 추가하여 OpenCV 기반의 이미지 처리 기능을 정상적으로 사용할 수 있다.

(10) AndroidManifest.xml 설정

본 프로젝트의 안드로이드 애플리케이션은 이미지 촬영, 모델 추론, 파일 접근 등의 기능을 포함하고 있으며, 이를 위해 AndroidManifest.xml 파일에서 필요한 권한 및 구성 요소들을 명시적으로 선언하였다.
우선 카메라 기능을 사용하기 위해 android.permission.CAMERA 권한을 선언하였으며, 사용자가 디바이스 내장 카메라를 사용할 수 있도록 uses-feature 항목을 통해 카메라 기능을 선택적으로 지정하였다. 또한, 카메라 앱을 호출할 수 있도록 queries 요소에 IMAGE_CAPTURE 인텐트 액션을 등록하여 외부 앱과의 상호작용이 가능하도록 하였다.

애플리케이션은 SplashActivity를 진입점으로 설정하였으며, 해당 액티비티에는 MAIN 액션과 LAUNCHER 카테고리를 지정하여 앱 실행 시 가장 먼저 로드되도록 하였다. 이후 사용자는 MainActivity1으로 자동 전환되며, 여기서 카메라 촬영 또는 갤러리 선택을 통해 안저 이미지를 입력할 수 있다. 진단 결과는 MainActivity2에서 추론 및 출력되며, parentActivityName 속성을 통해 이전 액티비티로의 네비게이션 흐름도 정의되어 있다.

(11) 성능 평가 및 실험 결과

학습된 안구 질환 분류 모델의 성능을 정량적으로 평가하기 위해, 총 4개 클래스(AMD, Diabetic Retinopathy, Glaucoma, Normal)로 구성된 검증용 이미지 960장을 활용하여 예측 실험을 수행하였다. 실험에 사용된 데이터는 ODIR-5K 및 직접 수집한 스마트폰 이미지로 구성된 데이터셋 중, 학습 시 사용하지 않은 검증용 디렉토리에 저장된 이미지이다.
PyTorch 기반으로 구성된 평가 파이프라인에서는 입력 이미지를 학습 시와 동일하게 Resize(224) → CenterCrop(224) → Normalize 방식으로 전처리한 후, 학습된 모델에 입력하였다. 모델은 GPU 또는 CPU에서 추론되었으며, 예측 결과는 Scikit-learn의 confusion_matrix 및 classification_report를 사용하여 시각화 및 수치로 분석하였다.

(12) Grad-CAM++ 기반 모델 시각화

모델의 예측 근거를 시각적으로 분석하고, 딥러닝 기반 진단 시스템의 신뢰성과 해석 가능성을 높이기 위해 Grad-CAM++ (Gradient-weighted Class Activation Mapping++) 기법을 적용하였다. 본 기법은 이미지 내에서 모델이 주목한 영역을 시각적으로 강조해줌으로써, 의사결정 과정에 대한 설명 가능성을 제공한다.
시각화 대상 모델은 학습이 완료된 ResNet-18 기반의 안구 질환 분류 모델이며, 최종 합성곱 계층을 타겟 레이어로 지정하여 Grad-CAM++ 맵을 추출하였다. CAM은 모델이 특정 질환을 예측할 때 어떤 시각적 패턴에 주로 의존하는지를 보여준다.

(13) 특징 벡터 시각화 (t-SNE)

모델이 각 클래스를 어떤 기준으로 분류하고 있는지를 특징 공간 상에서 시각적으로 이해하기 위해, 본 프로젝트에서는 t-SNE(t-distributed Stochastic Neighbor Embedding) 기법을 활용하였다.
t-SNE는 고차원 특징 벡터를 2차원 평면으로 축소하여, 클래스 간의 클러스터링 경향성과 분리 정도를 직관적으로 확인할 수 있도록 도와주는 비선형 차원 축소 기법이다.
이를 위해 먼저 학습이 완료된 ResNet-18 모델의 마지막 Fully Connected Layer 이전까지의 구조를 통해 총 512차원의 고차원 특징 벡터를 추출하였다. 각 클래스(AMD, DR, Glaucoma, Normal)에 대해 총 4,800장의 학습 이미지를 처리하여 해당 특징 벡터들을 수집한 뒤, StandardScaler를 이용해 정규화를 수행하였다.
정규화된 벡터들은 t-SNE 알고리즘을 통해 2차원 임베딩으로 변환되었으며, 최종적으로 matplotlib과 seaborn 라이브러리를 활용하여 각 클래스별 특징 분포를 시각화하였다. 이 시각화를 통해 모델이 각 질병 클래스에 대해 얼마나 구분 가능한 특징을 학습했는지 시각적으로 분석할 수 있고, 질병 간 시각적 유사성으로 인해 발생하는 중첩 현상도 확인할 수 있다.

(14) 3D 프린팅 기반 스마트폰 어댑터 설계

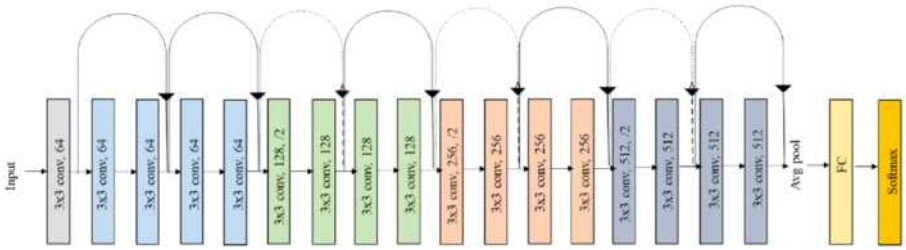
본 프로젝트에서는 안저 영상 촬영을 위한 스마트폰 기반 디바이스의 하드웨어 구현을 위해, Newopen source 3-dimensional printed smartphone fundus imaging adaptor (2019)에서 제안한 3D 프린팅 스마트폰 안저 촬영 어댑터 설계를 기반으로 하였다. 해당 논문에서는 렌즈 지지 구조물을 포함한 완전한 3D 프린팅 기반 어셈블리 구조를 공개하고 있으며, 추가적인 고정 장치나 별도 하드웨어 없이도 20D 렌즈만으로 간편한 안저 촬영이 가능한 장치를 제안하였다.
우리는 이 오픈소스 모델을 기본 골격으로 채택한 후, Samsung Galaxy S9의 후면 카메라 위치 및 크기에 맞도록 어댑터의 폰 케이스 부분을 직접 수정하여 최적화하였다. 구체적으로는 스마트폰 카메라 위치에 정확히 렌즈 축이 일치하도록 치수를 보정하고, 고정성이 높도록 폰 뒷면 구조를 보강하였다.
수정된 3D 모델은 TKINERCAD를 활용해 설계되었으며, 교내 3D 프린터를 이용해 PLA 필라멘트로 출력하였다. 이 구조물은 20D 렌즈 삽입부, 스마트폰 결착부, 손잡이 또는 지지대 구조 등으로 구성되며, 원활한 간접 안저 촬영이 가능하도록 설계되었다.

III. 각 모듈별 동작 원리

이름	이미지	동작원리
SAMSUNG Galaxy S9 (Exynos 9810)		스마트폰 플랫폼으로는 Exynos 9810 AP가 탑재된 Samsung Galaxy S9를 사용하였다. Exynos 9810은 4개의 고성능 Mongoose M3 코어와 4개의 Cortex-A55 코어로 구성되어 있으며, ARM Mali-G72 MP18 GPU, 10nm FinFET 공정 기반의 에너지 효율성과 처리 성능을 모두 갖춘 SoC이다. 이 디바이스에서는 학습된 PyTorch 모델을 TorchScript Lite(.ptl) 형태로 변환하여 앱 내에 탑재하고, 안드로이드 환경에서 OpenCV와 PyTorch Mobile을 함께 활용하여 추론을 수행하였다. 이미지 처리, 특징 추출, 분류 전 과정이 네트워크 연결 없이 단독으로 작동하도록 구성하였다.
20D FUNDUS LENS		20D 간접 검안용 렌즈(20D Fundus Lens)는 안과에서 실제로 널리 사용되는 렌즈로, 이 렌즈는 고굴절률의 볼록 렌즈로, 망막에서 반사된 빛을 집광하여 스마트폰 카메라가 초점을 맞출 수 있도록 해주는 광학 중계 장치 역할을 수행한다.
스마트폰 어댑터		스마트폰 어댑터는 스마트폰의 후면 카메라, 20D fundus lens, 피검자의 눈(안구) 사이의 광학적 정렬을 유지하고, 안정적인 안저 촬영을 가능하는 보조도구이다.

Ⅳ. 모듈별 설계

1. DEEP LEARNING AI MODEL



Resnet-18 Architecture

Google Colab 환경에서는 전체 데이터를 처리하고 딥러닝 모델을 학습하며, 최종적으로 모바일 환경에 최적화된 형태로 모델을 변환 및 저장하는 일련의 모듈들이 구성되어 있다.

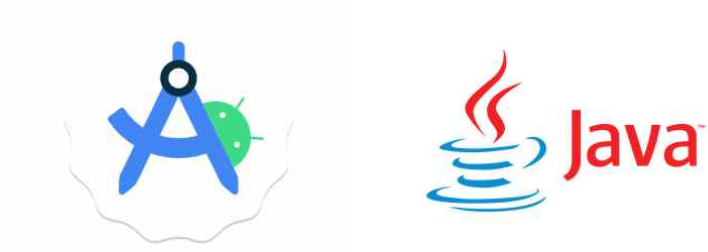
우선, 데이터 처리 모듈에서는 ODIR-5K 공개 데이터셋과 자체 촬영한 스마트폰 이미지를 포함한 총 4,800장의 안저 이미지를 사용하였으며, 모든 이미지는 CLAHE(Contrast Limited Adaptive Histogram Equalization) 적용 후 원형 크롭, 정사각형 패딩, 리사이징 등의 전처리를 거쳐 일관된 입력 형태로 변환되었다.

이후 데이터 증강 모듈에서는 학습 성능 향상과 과적합 방지를 위해 RandomRotation, ColorJitter, GaussianBlur, Cutout 등 다양한 augmentation 기법을 적용하였다.

모델 학습 모듈에서는 ImageNet 가중치로 사전 학습된 ResNet-18 구조를 기반으로, 마지막 fully connected layer를 변경하여 4클래스 분류가 가능하도록 재설계하였다. 중간에는 Dropout, BatchNorm, ReLU를 삽입하여 일반화 성능을 높였으며, CrossEntropyLoss를 기준으로 Adam optimizer와 ReduceLROnPla

teau scheduler를 적용해 최적화 과정을 수행하였다.
학습 완료 후에는 모델을 .pth 포맷으로 저장한 뒤, PyTorch의 torch.jit.trace()를 이용해 .pt 파일로 변환하고, optimize_for_mobile()을 통해 .ptl 파일로 경량화하여 Android에서 추론 가능한 형태로 변환하였다.
또한 성능 분석 모듈에서는 Confusion Matrix, Classification Report, F1-score, Grad-CAM++, t-SNE 등을 활용해 모델의 클래스 분별력, 시각적 근거, 내부 특징 공간 등을 분석하였다. 이를 통해 학습된 모델이 실제 진단에 사용될 수 있을 만큼 높은 신뢰성과 일관성을 가지는지를 정량적·정성적으로 평가하였다.

2. Android 기반 실행 모듈



스마트폰 앱에서는 MainActivity1과 MainActivity2를 중심으로, 사용자 인터페이스와 딥러닝 추론 시스템이 모듈화되어 구성되었다.
MainActivity1에서는 사용자가 직접 스마트폰으로 안저 사진을 촬영하거나 갤러리에서 이미지를 선택할 수 있는 인터페이스를 제공하며, 선택된 이미지 경로를 다음 액티비티로 전달한다. 카메라로 촬영한 이미지는 FileProvider를 통해 앱 내부에 저장되며, 필요 시 Exif 정보를 제거하고 리사이징 및 압축 처리를 수행한다.
MainActivity2는 실질적인 딥러닝 기반 추론이 수행되는 모듈이다. 여기에서는 PyTorch Lite를 통해 Colab에서 생성된 .ptl 모델을 불러온 후, OpenCV 기반의 전처리 함수인 cropFundusCircle()을 활용해 원형 영역 검출, CLAHE 적용, 정사각형 패딩, 리사이즈 및 정규화를 수행한다. 이 과정을 통해 얻어진 이미지 텐서는 PyTorch 모델에 입력되어 예측값을 출력하며, 결과는 Softmax 확률을 기반으로 TextView에 클래스별 확률 형태로 표시된다. 이때 예측 클래스는 AMD,

Diabetic Retinopathy, Glaucoma, Normal 중 하나로 출력된다.
앱은 전체적으로 PyTorch Mobile Lite, OpenCV Android, ExifInterface, ConstraintLayout 등 다양한 라이브러리를 포함하고 있으며, build.gradle과 AndroidManifest.xml에서 필요한 권한 및 종속성을 설정하였다. 특히 카메라 접근 권한, 이미지 압축 및 저장, assets 폴더로부터의 모델 로딩 경로 설정 등도 정확하게 구성하였다.

3. 하드웨어 모듈

Colab과 Android 앱 모두와 연계되어 작동하는 공통 모듈로는 20D fundus lens와 3D 프린팅 광학 어댑터가 있다.
20D 렌즈는 약 50mm의 초점거리와 46~50도의 시야각을 제공하는 간접 검안용 렌즈로, 안저 촬영 시 망막의 실제상을 형성하여 스마트폰 카메라가 초점을 맞추고 이미지를 캡처할 수 있게 한다. 이 렌즈는 황반, 시신경 유두 등 주요 구조를 넓은 시야로 포착하기에 적합하며, 스마트폰과 사용자의 눈 사이에 배치되어 광학적 초점을 안정적으로 유지한다.
이 렌즈를 고정하기 위한 구조물은 3D 프린터로 제작된 맞춤형 어댑터로, Galaxy S9의 후면 카메라 위치에 맞추어 설계되었다. 어댑터는 렌즈와 스마트폰 카메라가 동일한 광축 상에 위치하도록 하며, 사용자가 손으로 들고 조작하거나 피검자의 동공에 정확히 정렬시킬 수 있도록 돕는다. 이는 하드웨어 시스템 내에서 광학 모듈로 작동하며, 소프트웨어와의 결합을 통해 실시간 안저 진단이 가능하게 한다.

V. 전체 시스템 설계

본 프로젝트는 스마트폰을 활용하여 오프라인 환경에서도 실시간으로 안구 질환을 진단할 수 있는 온디바이스 인공지능 시스템을 개발하는 것을 목표로 하였다. 이를 위해 전체 시스템은 크게 세 가지 핵심 구성 요소로 설계되었다: 첫째, 학습 및 모델 최적화를 수행하는 Google Colab 기반의 클라우드 환경, 둘째, 추론과 사용자 인터페이스가 실행되는 Android 기반 모바일 앱 환경, 셋째, 고해상도 안저 촬영을 위한 광학 하드웨어 구성이다.

학습 환경에서는 ODIR-5K 데이터셋과 스마트폰으로 직접 촬영한 안저 이미지 총 4,800장을 활용하여 ResNet-18 기반의 분류 모델을 학습하였다. 이 과정에서 CLAHE, 중심 원 검출 기반 크롭, 정사각형 패딩, 224×224 크기 리사이징 및 정규화 등의 전처리 기법이 적용되었으며, Cutout과 ColorJitter 등의 증강 기법도 함께 사용되었다. 학습된 모델은 PyTorch에서 .pth 파일로 저장된 후, torch.jit.trace()를 통해 .pt 형태로 변환되고, 다시 optimize_for_mobile() 함수를 통해 .ptl 포맷으로 경량화되어 Android 앱에서 사용할 수 있도록 준비되었다.

모바일 앱 환경에서는 MainActivity1에서 카메라를 이용해 실시간으로 안저 이미지를 촬영하거나 갤러리에서 이미지를 불러올 수 있도록 구현하였다. 이후 MainActivity2에서는 OpenCV를 활용해 CLAHE 적용, 원형 크롭, 정사각형 패딩, 리사이징, 정규화를 포함한 전처리를 수행한 뒤, PyTorch Lite를 통해 로드된 .ptl 모델로 이미지를 추론한다. 추론 결과는 TextView를 통해 클래스별 확률 형태로 사용자에게 제공되며, "다시 진단하기" 버튼을 통해 반복 검사가 가능하도록 설계되었다.

하드웨어 측면에서는 삼성 Galaxy S9 기종에 맞춰 설계된 3D 프린팅 스마트폰 어댑터와 20D fundus lens가 사용되었다. 이 광학 장치는 스마트폰 카메라와 사용자 눈 사이의 정밀한 광축 정렬을 유지할 수 있도록 하며, 고가의 안저 촬영 장비 없이도 충분한 품질의 이미지를 확보할 수 있도록 해준다. 이러한 광학 구성은 안과 진단에서 가장 핵심적인 시신경 유두와 황반을 선명하게 포착할 수 있도록 설계되었으며, 결과적으로 하드웨어적인 초점 보정 없이도 안저 질환 진단에 필요한 입력 이미지를 안정적으로 확보할 수 있다.

전반적으로 본 시스템은 클라우드 없이도 스마트폰 단독으로 질병 분류를 수행할 수 있도록 구성되었으며, 모델 경량화 및 전처리 최적화를 통해 모바일 환경에서도 높은 정확도와 빠른 추론 속도를 확보할 수 있도록 설계되었다. 또한 사용자

인터페이스 측면에서도 앱 실행 초기의 Splash 화면부터 카메라 촬영, 결과 출력까지의 모든 흐름이 직관적으로 구성되어, 의료 지식이 없는 일반 사용자도 쉽게 사용할 수 있도록 개발되었다.

VI. 제작내용 (회로도, 소스코드 등 첨부)

1. Resnet-18 딥러닝 코드

```
# DL_Resnet-18
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
import torchvision.datasets as datasets
from torch.utils.data import Dataset, DataLoader
import torchvision.models as models
import os
from PIL import Image
from torch.utils.mobile_optimizer import optimize_for_mobile
import torchvision.transforms.functional as TF
from google.colab import drive
import random
import matplotlib.pyplot as plt
import numpy as np
import cv2

# 랜덤 시드 고정
def set_seed(seed=42):
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.deterministic = True
```

```
    torch.backends.cudnn.benchmark = False

set_seed(42)

# Google Drive 강제 마운트
drive_path = '/content/drive'
drive.mount(drive_path, force_remount=True)

# Colab 내부 경로 사용
data_dir = "/content/drive/MyDrive/seg_dataset2_Alpha"
train_dir = os.path.join(data_dir, "train")
valid_dir = os.path.join(data_dir, "valid")

def smart_crop_retina(pil_image):
    img = np.array(pil_image)

    # 1. CLAHE 적용 (Contrast Limited Adaptive Histogram Equalization)
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    clahe = cv2.createCLAHE(clipLimit=1.0, tileGridSize=(8, 8))
    gray = clahe.apply(gray)

    # 2. 원 검출을 위한 블러 적용
    gray = cv2.medianBlur(gray, 5)

    # 3. Hough Circle로 원 검출
    height, width = gray.shape
    min_r = int(min(height, width) * 0.25)
    max_r = int(min(height, width) * 0.6)
    circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, dp=1.2,
minDist=100,
                               param1=60, param2=45,
minRadius=min_r, maxRadius=max_r)

    # 4. 원이 감지되면 중심을 기준으로 crop
```

```
    if circles is not None:
        circles = np.round(circles[0, :]).astype("int")
        x, y, r = circles[0]
        r = int(r * 0.96)
        crop_size = int(r * 2.2)
        x1 = max(0, x - crop_size // 2)
        y1 = max(0, y - crop_size // 2)
        x2 = min(width, x + crop_size // 2)
        y2 = min(height, y + crop_size // 2)
        cropped_img = img[y1:y2, x1:x2]
    else:
        cropped_img = img  # 원 검출 실패 시 원본 사용

    return Image.fromarray(cropped_img)


def pad_to_square(image):
    w, h = image.size
    max_side = max(w, h)
    padding = (
        (max_side - w) // 2,
        (max_side - h) // 2,
        (max_side - w + 1) // 2,
        (max_side - h + 1) // 2
    )
    return TF.pad(image, padding, fill=0, padding_mode='constant')

class Cutout(object):
    def __init__(self, n_holes, length):
        self.n_holes = n_holes
        self.length = length
    def __call__(self, img):
        h = img.size(1)
        w = img.size(2)
```

```
            mask = torch.ones((1, h, w), dtype=torch.float32)
            for _ in range(self.n_holes):
                y = random.randint(0, h - self.length)
                x = random.randint(0, w - self.length)
                mask[:, y:y+self.length, x:x+self.length] = 0.
            mask = mask.expand_as(img)
            return img * mask

# left 안구 사진만 좌우반전 적용
class LeftFlipDataset(Dataset):
    def __init__(self, root_dir, transform=None, apply_left_flip=False):
        self.root_dir = root_dir
        self.transform = transform
        self.image_paths = []
        self.labels = []
        self.classes = sorted(os.listdir(root_dir))
        self.apply_left_flip = apply_left_flip
        for i, class_name in enumerate(self.classes):
            class_path = os.path.join(root_dir, class_name)
            if os.path.isdir(class_path):
                for img_name in os.listdir(class_path):
                    self.image_paths.append(os.path.join(class_path,
img_name))
                    self.labels.append(i)
    def __len__(self):
        return len(self.image_paths)
    def __getitem__(self, idx):
        img_path = self.image_paths[idx]
        label = self.labels[idx]
        image = Image.open(img_path).convert('RGB')
        if self.apply_left_flip and "left" in os.path.basename(img_path):
            image = TF.hflip(image)
        if self.transform:
            image = self.transform(image)
```

```
            return image, label

# 학습용 transform
train_transform = transforms.Compose([
    transforms.Lambda(smart_crop_retina),
    transforms.Lambda(pad_to_square),
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.RandomRotation(degrees=(-2, 2)),
    transforms.ColorJitter(brightness=0.5, contrast=0.3, saturation=0.3,
hue=0.04),
    transforms.RandomAffine(degrees=0, translate=(0.1, 0.1)),
    transforms.GaussianBlur(kernel_size=3),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225]),
    Cutout(n_holes=2, length=24)
])

# 검증용 transform
valid_transform = transforms.Compose([
    transforms.Lambda(smart_crop_retina),
    transforms.Lambda(pad_to_square),
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# 데이터 로더
train_dataset = LeftFlipDataset(train_dir, train_transform,
apply_left_flip=True)
valid_dataset = LeftFlipDataset(valid_dir, valid_transform,
```

```
apply_left_flip=True)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True, num_workers=2)
valid_loader = DataLoader(valid_dataset, batch_size=32, shuffle=False, num_workers=2)

# 모델 정의
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = models.resnet18(weights='IMAGENET1K_V1')
num_ftrs = model.fc.in_features
num_classes = len(train_dataset.classes)
model.fc = nn.Sequential(
    nn.Dropout(0.6),
    nn.Linear(num_ftrs, 256),
    nn.BatchNorm1d(256),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(256, num_classes)
)
model = model.to(device)

# GPU 사용 가능 여부 확인 및 출력
if torch.cuda.is_available():
    print("CUDA is available!")
    print(f"Device name: {torch.cuda.get_device_name(0)}")
    print(f"Number of GPUs: {torch.cuda.device_count()}")
else:
    print("CUDA is NOT available. Training will be on CPU.")

# 손실 함수, 옵티마이저, 스케줄러
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001, weight_decay=1e-4)
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer,
```

```
mode='max', factor=0.1, patience=5, threshold=1e-4, verbose=True, min_lr=1e-6)

# 학습 함수
def train(model, train_loader, valid_loader, criterion, optimizer, scheduler, epochs=50):
    best_valid_acc = 0.0
    best_model_state = None
    best_epoch = -1
    train_accuracies = []
    valid_accuracies = []

    for epoch in range(epochs):
        model.train()
        total_loss, correct, total = 0, 0, 0
        print(f"₩n Epoch {epoch+1}/{epochs} 시작...")

        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            if torch.isnan(loss):
                print("Loss is Nan")
                return None, train_accuracies, valid_accuracies, best_epoch
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
            _, predicted = outputs.max(1)
            correct += (predicted == labels).sum().item()
            total += labels.size(0)

        train_acc = 100 * correct / total
```

```
        train_accuracies.append(train_acc)
        print(f" Epoch [{epoch+1}/{epochs}] 완료 - Loss: {total_loss/len(train_loader):.4f}, Accuracy: {train_acc:.2f}%")

        # Validation
        model.eval()
        val_loss, correct, total = 0, 0, 0
        with torch.no_grad():
            for images, labels in valid_loader:
                images, labels = images.to(device), labels.to(device)
                outputs = model(images)
                loss = criterion(outputs, labels)
                val_loss += loss.item()
                _, predicted = outputs.max(1)
                correct += (predicted == labels).sum().item()
                total += labels.size(0)

        val_loss /= len(valid_loader)
        valid_acc = 100 * correct / total
        valid_accuracies.append(valid_acc)
        print(f" Validation Loss: {val_loss:.4f}, Accuracy: {valid_acc:.2f}%")

        # Best model 저장
        if valid_acc > best_valid_acc:
            best_valid_acc = valid_acc
            best_model_state = model.state_dict()
            best_epoch = epoch + 1

        scheduler.step(val_loss)

    return best_model_state, train_accuracies, valid_accuracies, best_epoch
```

```
# 학습 및 저장
best_model, train_acc_history, valid_acc_history, best_epoch = train(
    model, train_loader, valid_loader, criterion, optimizer, scheduler,
epochs=50)

if best_model is not None:
    model.load_state_dict(best_model)
    model.eval()
    model.to(torch.device("cpu"))

    # 저장 경로 설정
    save_base = "/content/drive/MyDrive"
    pth_path = os.path.join(save_base, "resnet18_mobile_3.1.pth")
    pt_path = os.path.join(save_base, "resnet18_mobile_3.1.pt")
    ptl_path = os.path.join(save_base, "resnet18_mobile_3.1.ptl")

    # .pth 저장
    torch.save(model.state_dict(), pth_path)

    try:
        torch.manual_seed(42)
        example_input = torch.randn(1, 3, 224, 224,
dtype=torch.float32)
        traced_model = torch.jit.trace(model, example_input)
        traced_model = torch.jit.freeze(traced_model)

        # .pt 저장
        torch.jit.save(traced_model, pt_path)

        # .ptl 저장 (Lite용)
        optimized_model = optimize_for_mobile(traced_model)
        optimized_model._save_for_lite_interpreter(ptl_path)

        print(f"모델 저장 완료 (.pth, .pt, .ptl) | 최적 에폭: {best_epoch}")
```

```
    except Exception as e:
        print(f"모델 저장 중 오류 발생: {e}")
else:
    print("학습 실패 - 저장된 모델 없음.")

# 정확도 시각화
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(train_acc_history) + 1), train_acc_history, label='Train Accuracy')
plt.plot(range(1, len(valid_acc_history) + 1), valid_acc_history, label='Valid Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy (%)')
plt.title('Train and Validation Accuracy over Epochs')
plt.legend()
plt.grid(True)
plt.show()
```

2. Efficientent-B0 딥러닝 코드

```
//DL_Efficientnet_B0
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
import torchvision.datasets as datasets
from torch.utils.data import Dataset, DataLoader
import torchvision.models as models
import os
from PIL import Image
from torch.utils.mobile_optimizer import optimize_for_mobile
from torchvision.models import efficientnet_b0, EfficientNet_B0_Weights
import torchvision.transforms.functional as TF
```

```
from google.colab import drive
import random
import matplotlib.pyplot as plt
import numpy as np
import cv2

# 랜덤 시드 고정
def set_seed(seed=42):
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False

set_seed(42)

# Google Drive 강제 마운트
drive_path = '/content/drive'
drive.mount(drive_path, force_remount=True)

# Colab 내부 경로 사용
data_dir = "/content/drive/MyDrive/seg_dataset2_Alpha"
train_dir = os.path.join(data_dir, "train")
valid_dir = os.path.join(data_dir, "valid")

# 전처리 과정
def smart_crop_retina(pil_image):
    img = np.array(pil_image)

    # 1. CLAHE 적용 (Contrast Limited Adaptive Histogram Equalization)
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    clahe = cv2.createCLAHE(clipLimit=1.0, tileGridSize=(8, 8))
    gray = clahe.apply(gray)
```

```
    # 2. 원 검출을 위한 블러 적용
    gray = cv2.medianBlur(gray, 5)

    # 3. Hough Circle로 원 검출
    height, width = gray.shape
    min_r = int(min(height, width) * 0.25)
    max_r = int(min(height, width) * 0.6)
    circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, dp=1.2,
minDist=100,
                               param1=60, param2=45,
minRadius=min_r, maxRadius=max_r)

    # 4. 원이 감지되면 중심을 기준으로 crop
    if circles is not None:
        circles = np.round(circles[0, :]).astype("int")
        x, y, r = circles[0]
        r = int(r * 0.96)
        crop_size = int(r * 2.2)
        x1 = max(0, x - crop_size // 2)
        y1 = max(0, y - crop_size // 2)
        x2 = min(width, x + crop_size // 2)
        y2 = min(height, y + crop_size // 2)
        cropped_img = img[y1:y2, x1:x2]
    else:
        cropped_img = img  # 원 검출 실패 시 원본 사용

    return Image.fromarray(cropped_img)


def pad_to_square(image):
    w, h = image.size
    max_side = max(w, h)
    padding = (
```

```
            (max_side - w) // 2,
            (max_side - h) // 2,
            (max_side - w + 1) // 2,
            (max_side - h + 1) // 2
        )
        return TF.pad(image, padding, fill=0, padding_mode='constant')

class Cutout(object):
    def __init__(self, n_holes, length):
        self.n_holes = n_holes
        self.length = length
    def __call__(self, img):
        h = img.size(1)
        w = img.size(2)
        mask = torch.ones((1, h, w), dtype=torch.float32)
        for _ in range(self.n_holes):
            y = random.randint(0, h - self.length)
            x = random.randint(0, w - self.length)
            mask[:, y:y+self.length, x:x+self.length] = 0.
        mask = mask.expand_as(img)
        return img * mask

# 커스텀 Dataset
class LeftFlipDataset(Dataset):
    def __init__(self, root_dir, transform=None, apply_left_flip=False):
        self.root_dir = root_dir
        self.transform = transform
        self.image_paths = []
        self.labels = []
        self.classes = sorted(os.listdir(root_dir))
        self.apply_left_flip = apply_left_flip
        for i, class_name in enumerate(self.classes):
            class_path = os.path.join(root_dir, class_name)
            if os.path.isdir(class_path):
```

```
                for img_name in os.listdir(class_path):
                    self.image_paths.append(os.path.join(class_path,
img_name))
                    self.labels.append(i)
    def __len__(self):
        return len(self.image_paths)
    def __getitem__(self, idx):
        img_path = self.image_paths[idx]
        label = self.labels[idx]
        image = Image.open(img_path).convert('RGB')
        if self.apply_left_flip and "left" in os.path.basename(img_path):
            image = TF.hflip(image)
        if self.transform:
            image = self.transform(image)
        return image, label

# 학습용 transform
train_transform = transforms.Compose([
    # transforms.Lambda(smart_crop_retina),
    # transforms.Lambda(pad_to_square),
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.RandomRotation(degrees=(-2, 2)),
    transforms.ColorJitter(brightness=0.5, contrast=0.3, saturation=0.3,
hue=0.04),
    transforms.RandomAffine(degrees=0, translate=(0.1, 0.1)),
    transforms.GaussianBlur(kernel_size=3),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225]),
    Cutout(n_holes=2, length=24)
])

# 검증용 transform
```

```
valid_transform = transforms.Compose([
    # transforms.Lambda(smart_crop_retina),
    # transforms.Lambda(pad_to_square),
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# 데이터 로더
train_dataset = LeftFlipDataset(train_dir, train_transform, apply_left_flip=True)
valid_dataset = LeftFlipDataset(valid_dir, valid_transform, apply_left_flip=True)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True, num_workers=2)
valid_loader = DataLoader(valid_dataset, batch_size=32, shuffle=False, num_workers=2)

# 모델 정의
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
weights = EfficientNet_B0_Weights.DEFAULT
model = efficientnet_b0(weights=weights)

num_ftrs = model.classifier[1].in_features
num_classes = len(train_dataset.classes)
model.classifier = nn.Sequential(
    nn.Dropout(0.6),
    nn.Linear(num_ftrs, 256),
    nn.BatchNorm1d(256),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(256, num_classes)
```

```
)
model = model.to(device)

# GPU 사용 가능 여부 확인 및 출력
if torch.cuda.is_available():
    print("CUDA is available!")
    print(f"Device name: {torch.cuda.get_device_name(0)}")
    print(f"Number of GPUs: {torch.cuda.device_count()}")
else:
    print("CUDA is NOT available. Training will be on CPU.")

# 손실 함수, 옵티마이저, 스케줄러
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001,
weight_decay=1e-4)
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer,
mode='max', factor=0.1, patience=5,
                                                 threshold=1e-4,
verbose=True, min_lr=1e-6)

# 학습 함수
def train(model, train_loader, valid_loader, criterion, optimizer, scheduler,
epochs=50):
    best_valid_acc = 0.0
    best_model_state = None
    best_epoch = -1
    train_accuracies = []
    valid_accuracies = []

    for epoch in range(epochs):
        model.train()
        total_loss, correct, total = 0, 0, 0
        print(f"₩n Epoch {epoch+1}/{epochs} 시작...")
```

```
        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            if torch.isnan(loss):
                print("Loss is Nan")
                return None, train_accuracies, valid_accuracies,
best_epoch
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
            _, predicted = outputs.max(1)
            correct += (predicted == labels).sum().item()
            total += labels.size(0)

        train_acc = 100 * correct / total
        train_accuracies.append(train_acc)
        print(f" Epoch [{epoch+1}/{epochs}] 완료 - Loss:
{total_loss/len(train_loader):.4f}, Accuracy: {train_acc:.2f}%")

        # Validation
        model.eval()
        val_loss, correct, total = 0, 0, 0
        with torch.no_grad():
            for images, labels in valid_loader:
                images, labels = images.to(device), labels.to(device)
                outputs = model(images)
                loss = criterion(outputs, labels)
                val_loss += loss.item()
                _, predicted = outputs.max(1)
                correct += (predicted == labels).sum().item()
                total += labels.size(0)
```

```
            val_loss /= len(valid_loader)
            valid_acc = 100 * correct / total
            valid_accuracies.append(valid_acc)
            print(f" Validation Loss: {val_loss:.4f}, Accuracy: {valid_acc:.2f}%")

            # Best model 저장
            if valid_acc > best_valid_acc:
                best_valid_acc = valid_acc
                best_model_state = model.state_dict()
                best_epoch = epoch + 1

            scheduler.step(val_loss)

    return best_model_state, train_accuracies, valid_accuracies, best_epoch

# 학습 및 저장
best_model, train_acc_history, valid_acc_history, best_epoch = train(
    model, train_loader, valid_loader, criterion, optimizer, scheduler, epochs=50)

if best_model is not None:
    model.load_state_dict(best_model)
    model.eval()
    model.to(torch.device("cpu"))

    # 저장 경로 설정
    save_base = "/content/drive/MyDrive"
    pth_path = os.path.join(save_base, "efficientnetB0_mobile_1.0.pth")
    pt_path = os.path.join(save_base, "efficientnetB0_mobile_1.0.pt")
    ptl_path = os.path.join(save_base, "efficientnetB0_mobile_1.0.ptl")

    # .pth 저장
```

```
        torch.save(model.state_dict(), pth_path)

        try:
            torch.manual_seed(42)
            example_input = torch.randn(1, 3, 224, 224,
dtype=torch.float32)
            traced_model = torch.jit.trace(model, example_input)
            traced_model = torch.jit.freeze(traced_model)

            # .pt 저장
            torch.jit.save(traced_model, pt_path)

            # .ptl 저장 (Lite용)
            optimized_model = optimize_for_mobile(traced_model)
            optimized_model._save_for_lite_interpreter(ptl_path)

            print(f"모델 저장 완료 (.pth, .pt, .ptl) | 최적 에폭: {best_epoch}")
        except Exception as e:
            print(f"모델 저장 중 오류 발생: {e}")
else:
        print("학습 실패 - 저장된 모델 없음.")

# 정확도 시각화
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(train_acc_history) + 1), train_acc_history,
label='Train Accuracy')
plt.plot(range(1, len(valid_acc_history) + 1), valid_acc_history,
label='Valid Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy (%)')
plt.title('Train and Validation Accuracy over Epochs')
plt.legend()
plt.grid(True)
plt.show()
```

3. Mobilenet-V2 딥러닝 코드

```
//DL_Mobilenet_V2
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
import torchvision.datasets as datasets
from torch.utils.data import Dataset, DataLoader
import torchvision.models as models
import os
from PIL import Image
from torch.utils.mobile_optimizer import optimize_for_mobile
from torchvision.models import mobilenet_v2, MobileNet_V2_Weights
import torchvision.transforms.functional as TF
from google.colab import drive
import random
import matplotlib.pyplot as plt
import numpy as np
import cv2

# 랜덤 시드 고정
def set_seed(seed=42):
    random.seed(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False

set_seed(42)
```

```
# Google Drive 강제 마운트
drive_path = '/content/drive'
drive.mount(drive_path, force_remount=True)

# Colab 내부 경로 사용
data_dir = "/content/drive/MyDrive/seg_dataset2_Alpha"
train_dir = os.path.join(data_dir, "train")
valid_dir = os.path.join(data_dir, "valid")

# 전처리 과정
def smart_crop_retina(pil_image):
    img = np.array(pil_image)

    # 1. CLAHE 적용 (Contrast Limited Adaptive Histogram Equalization)
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    clahe = cv2.createCLAHE(clipLimit=1.0, tileGridSize=(8, 8))
    gray = clahe.apply(gray)

    # 2. 원 검출을 위한 블러 적용
    gray = cv2.medianBlur(gray, 5)

    # 3. Hough Circle로 원 검출
    height, width = gray.shape
    min_r = int(min(height, width) * 0.25)
    max_r = int(min(height, width) * 0.6)
    circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, dp=1.2,
minDist=100,
                               param1=60, param2=45,
minRadius=min_r, maxRadius=max_r)

    # 4. 원이 감지되면 중심을 기준으로 crop
    if circles is not None:
        circles = np.round(circles[0, :]).astype("int")
        x, y, r = circles[0]
```

```
            r = int(r * 0.96)
            crop_size = int(r * 2.2)
            x1 = max(0, x - crop_size // 2)
            y1 = max(0, y - crop_size // 2)
            x2 = min(width, x + crop_size // 2)
            y2 = min(height, y + crop_size // 2)
            cropped_img = img[y1:y2, x1:x2]
        else:
            cropped_img = img  # 원 검출 실패 시 원본 사용

        return Image.fromarray(cropped_img)


def pad_to_square(image):
    w, h = image.size
    max_side = max(w, h)
    padding = (
        (max_side - w) // 2,
        (max_side - h) // 2,
        (max_side - w + 1) // 2,
        (max_side - h + 1) // 2
    )
    return TF.pad(image, padding, fill=0, padding_mode='constant')

class Cutout(object):
    def __init__(self, n_holes, length):
        self.n_holes = n_holes
        self.length = length
    def __call__(self, img):
        h = img.size(1)
        w = img.size(2)
        mask = torch.ones((1, h, w), dtype=torch.float32)
        for _ in range(self.n_holes):
            y = random.randint(0, h - self.length)
```

```
            x = random.randint(0, w - self.length)
            mask[:, y:y+self.length, x:x+self.length] = 0.
        mask = mask.expand_as(img)
        return img * mask

# 커스텀 Dataset
class LeftFlipDataset(Dataset):
    def __init__(self, root_dir, transform=None, apply_left_flip=False):
        self.root_dir = root_dir
        self.transform = transform
        self.image_paths = []
        self.labels = []
        self.classes = sorted(os.listdir(root_dir))
        self.apply_left_flip = apply_left_flip
        for i, class_name in enumerate(self.classes):
            class_path = os.path.join(root_dir, class_name)
            if os.path.isdir(class_path):
                for img_name in os.listdir(class_path):
                    self.image_paths.append(os.path.join(class_path,
img_name))
                    self.labels.append(i)
    def __len__(self):
        return len(self.image_paths)
    def __getitem__(self, idx):
        img_path = self.image_paths[idx]
        label = self.labels[idx]
        image = Image.open(img_path).convert('RGB')
        if self.apply_left_flip and "left" in os.path.basename(img_path):
            image = TF.hflip(image)
        if self.transform:
            image = self.transform(image)
        return image, label

# 학습용 transform
```

```
train_transform = transforms.Compose([
    # transforms.Lambda(smart_crop_retina),
    # transforms.Lambda(pad_to_square),
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.RandomRotation(degrees=(-2, 2)),
    transforms.ColorJitter(brightness=0.5, contrast=0.3, saturation=0.3,
hue=0.04),
    transforms.RandomAffine(degrees=0, translate=(0.1, 0.1)),
    transforms.GaussianBlur(kernel_size=3),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225]),
    Cutout(n_holes=2, length=24)
])

# 검증용 transform
valid_transform = transforms.Compose([
    # transforms.Lambda(smart_crop_retina),
    # transforms.Lambda(pad_to_square),
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# 데이터 로더
train_dataset = LeftFlipDataset(train_dir, train_transform,
apply_left_flip=True)
valid_dataset = LeftFlipDataset(valid_dir, valid_transform,
apply_left_flip=True)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True,
num_workers=2)
```

```
valid_loader = DataLoader(valid_dataset, batch_size=32, shuffle=False, num_workers=2)

# 모델 정의
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
weights = MobileNet_V2_Weights.DEFAULT
model = mobilenet_v2(weights=weights)

num_ftrs = model.classifier[1].in_features
num_classes = len(train_dataset.classes)
model.classifier = nn.Sequential(
    nn.Dropout(0.6),
    nn.Linear(num_ftrs, 256),
    nn.BatchNorm1d(256),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(256, num_classes)
)
model = model.to(device)

# GPU 사용 가능 여부 확인 및 출력
if torch.cuda.is_available():
    print("CUDA is available!")
    print(f"Device name: {torch.cuda.get_device_name(0)}")
    print(f"Number of GPUs: {torch.cuda.device_count()}")
else:
    print("CUDA is NOT available. Training will be on CPU.")

# 손실 함수, 옵티마이저, 스케줄러
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001, weight_decay=1e-4)
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='max', factor=0.1, patience=5,
```

```
                                                    threshold=1e-4,
verbose=True, min_lr=1e-6)

# 학습 함수
def train(model, train_loader, valid_loader, criterion, optimizer, scheduler,
epochs=50):
    best_valid_acc = 0.0
    best_model_state = None
    best_epoch = -1
    train_accuracies = []
    valid_accuracies = []

    for epoch in range(epochs):
        model.train()
        total_loss, correct, total = 0, 0, 0
        print(f"₩n Epoch {epoch+1}/{epochs} 시작...")

        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            if torch.isnan(loss):
                print("Loss is Nan")
                return None, train_accuracies, valid_accuracies,
best_epoch
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
            _, predicted = outputs.max(1)
            correct += (predicted == labels).sum().item()
            total += labels.size(0)

        train_acc = 100 * correct / total
```

```
        train_accuracies.append(train_acc)
        print(f" Epoch [{epoch+1}/{epochs}] 완료 - Loss: {total_loss/len(train_loader):.4f}, Accuracy: {train_acc:.2f}%")

        # Validation
        model.eval()
        val_loss, correct, total = 0, 0, 0
        with torch.no_grad():
            for images, labels in valid_loader:
                images, labels = images.to(device), labels.to(device)
                outputs = model(images)
                loss = criterion(outputs, labels)
                val_loss += loss.item()
                _, predicted = outputs.max(1)
                correct += (predicted == labels).sum().item()
                total += labels.size(0)

        val_loss /= len(valid_loader)
        valid_acc = 100 * correct / total
        valid_accuracies.append(valid_acc)
        print(f" Validation Loss: {val_loss:.4f}, Accuracy: {valid_acc:.2f}%")

        # Best model 저장
        if valid_acc > best_valid_acc:
            best_valid_acc = valid_acc
            best_model_state = model.state_dict()
            best_epoch = epoch + 1

        scheduler.step(val_loss)

    return best_model_state, train_accuracies, valid_accuracies, best_epoch
```

```
# 학습 및 저장
best_model, train_acc_history, valid_acc_history, best_epoch = train(
    model, train_loader, valid_loader, criterion, optimizer, scheduler,
epochs=50)

if best_model is not None:
    model.load_state_dict(best_model)
    model.eval()
    model.to(torch.device("cpu"))

    # 저장 경로 설정
    save_base = "/content/drive/MyDrive"
    pth_path = os.path.join(save_base, "mobilenetv2_mobile_1.0.pth")
    pt_path = os.path.join(save_base, "mobilenetv2_mobile_1.0.pt")
    ptl_path = os.path.join(save_base, "mobilenetv2_mobile_1.0.ptl")

    # .pth 저장
    torch.save(model.state_dict(), pth_path)

    try:
        torch.manual_seed(42)
        example_input = torch.randn(1, 3, 224, 224,
dtype=torch.float32)
        traced_model = torch.jit.trace(model, example_input)
        traced_model = torch.jit.freeze(traced_model)

        # .pt 저장
        torch.jit.save(traced_model, pt_path)

        # .ptl 저장 (Lite용)
        optimized_model = optimize_for_mobile(traced_model)
        optimized_model._save_for_lite_interpreter(ptl_path)

        print(f"모델 저장 완료 (.pth, .pt, .ptl) | 최적 에폭: {best_epoch}")
```

```
    except Exception as e:
        print(f"모델 저장 중 오류 발생: {e}")
else:
    print("학습 실패 - 저장된 모델 없음.")

# 정확도 시각화
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(train_acc_history) + 1), train_acc_history, label='Train Accuracy')
plt.plot(range(1, len(valid_acc_history) + 1), valid_acc_history, label='Valid Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy (%)')
plt.title('Train and Validation Accuracy over Epochs')
plt.legend()
plt.grid(True)
plt.show()
```

4. MainActivity1.java

```
//MainActivity1.java
package eye.application;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
```

```
import android.util.Log;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.core.content.FileProvider;
import androidx.exifinterface.media.ExifInterface;

import org.opencv.android.OpenCVLoader;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

public class MainActivity1 extends AppCompatActivity {

    private static final int CAMERA_PERMISSION_REQUEST_CODE = 100;
    private ActivityResultLauncher<Intent> cameraLauncher;
    private ActivityResultLauncher<Intent> galleryLauncher;

    private String currentPhotoPath;
    private ImageView imageView;

    static {
        if (!OpenCVLoader.initDebug()) {
            Log.d("OpenCV", "OpenCV initialization failed");
        } else {
```

```
            Log.d("OpenCV", "OpenCV initialized successfully");
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Log.d("MainActivity1", " onCreate 실행됨");

        setContentView(R.layout.activity_main);
        Log.d("MainActivity1", " setContentView 완료");

        if (getSupportActionBar() != null) getSupportActionBar().hide();

        Button cameraButton = findViewById(R.id.start_detection_button);
        Button galleryButton = findViewById(R.id.open_gallery_button);
        imageView = findViewById(R.id.image_view);
        imageView.setImageResource(R.drawable.retina_logo);

        // 카메라 버튼
        cameraButton.setOnClickListener(v -> {
            if (ContextCompat.checkSelfPermission(this,
Manifest.permission.CAMERA) ==
PackageManager.PERMISSION_GRANTED) {
                launchCameraIntent();
            } else {
                ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.CAMERA},
CAMERA_PERMISSION_REQUEST_CODE);
            }
        });

        // 카메라 결과 처리
```

```
        cameraLauncher = registerForActivityResult(new
ActivityResultContracts.StartActivityForResult(), result -> {
            if (result.getResultCode() == RESULT_OK) {
                File imageFile = new File(currentPhotoPath);
                if (imageFile.exists()) {
                    compressImageFile(currentPhotoPath, 85);
                    Intent intent = new Intent(MainActivity1.this,
MainActivity2.class);
                    intent.putExtra("image_path", currentPhotoPath);
                    startActivity(intent);
                    finish();
                } else {
                    Toast.makeText(this, "사진 촬영 실패",
Toast.LENGTH_SHORT).show();
                }
            }
        });

        // 갤러리 버튼
        galleryButton.setOnClickListener(v -> {
            Intent intent = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
            intent.setType("image/*");
            galleryLauncher.launch(intent);
        });

        // 갤러리 결과 처리
        galleryLauncher = registerForActivityResult(new
ActivityResultContracts.StartActivityForResult(), result -> {
            if (result.getResultCode() == RESULT_OK &&
result.getData() != null) {
                Uri imageUri = result.getData().getData();
                if (imageUri != null) {
                    try {
```

```
                    InputStream imageStream = getContentResolver().openInputStream(imageUri);
                    Bitmap bitmap = BitmapFactory.decodeStream(imageStream);

                    File tempFile = new File(getCacheDir(), "gallery_input.jpg");
                    FileOutputStream out = new FileOutputStream(tempFile);
                    bitmap.compress(Bitmap.CompressFormat.JPEG, 100, out);
                    out.close();

                    Intent intent = new Intent(MainActivity1.this, MainActivity2.class);
                    intent.putExtra("image_path", tempFile.getAbsolutePath());
                    startActivity(intent);
                    finish();

                } catch (IOException e) {
                    e.printStackTrace();
                    Toast.makeText(this, "이미지 불러오기 실패", Toast.LENGTH_SHORT).show();
                }
            }
        }
    });
}

private void launchCameraIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) !=
```

```
null) {
                File photoFile = createImageFile();
                if (photoFile != null) {
                    Uri photoUri = FileProvider.getUriForFile(this,
"eye.application.fileprovider", photoFile);
                    takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT,
photoUri);
                    cameraLauncher.launch(takePictureIntent);
                }
            }
        }

        private File createImageFile() {
            File storageDir =
getExternalFilesDir(Environment.DIRECTORY_PICTURES);
            try {
                File image = File.createTempFile("captured_image_", ".jpg",
storageDir);
                currentPhotoPath = image.getAbsolutePath();
                return image;
            } catch (IOException e) {
                e.printStackTrace();
                return null;
            }
        }

        private void compressImageFile(String inputPath, int quality) {
            try {
                Bitmap bitmap = BitmapFactory.decodeFile(inputPath);

                ExifInterface exif = new ExifInterface(inputPath);
                int orientation =
exif.getAttributeInt(ExifInterface.TAG_ORIENTATION,
ExifInterface.ORIENTATION_NORMAL);
```

```
            Matrix matrix = new Matrix();
            switch (orientation) {
                case ExifInterface.ORIENTATION_ROTATE_90:
matrix.postRotate(90); break;
                case ExifInterface.ORIENTATION_ROTATE_180:
matrix.postRotate(180); break;
                case ExifInterface.ORIENTATION_ROTATE_270:
matrix.postRotate(270); break;
            }
            if (orientation != ExifInterface.ORIENTATION_NORMAL) {
                bitmap = Bitmap.createBitmap(bitmap, 0, 0,
bitmap.getWidth(), bitmap.getHeight(), matrix, true);
            }

             int width = bitmap.getWidth();
            int height = bitmap.getHeight();
            if (width > 1000) {
                float scale = 1000f / width;
                int newHeight = (int) (height * scale);
                bitmap = Bitmap.createScaledBitmap(bitmap, 1000,
newHeight, true);
            }

            FileOutputStream out = new FileOutputStream(inputPath);
            bitmap.compress(Bitmap.CompressFormat.JPEG, quality,
out);
            out.flush();
            out.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if (requestCode == CAMERA_PERMISSION_REQUEST_CODE) {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                launchCameraIntent();
            } else {
                Toast.makeText(this, "카메라 권한이 필요합니다",
Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

5. MainActivity2.java

```
//MainActivity2.java
package eye.application;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.ProgressBar;
```

```
import android.view.View;

import androidx.appcompat.app.AppCompatActivity;

import org.opencv.android.OpenCVLoader;
import org.opencv.android.Utils;
import org.opencv.core.Mat;
import org.opencv.core.Rect;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;
import org.opencv.imgproc.CLAHE;
import org.opencv.core.Core;
import org.opencv.core.Scalar;

import org.pytorch.IValue;
import org.pytorch.LiteModuleLoader;
import org.pytorch.Module;
import org.pytorch.Tensor;
import org.pytorch.torchvision.TensorImageUtils;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Locale;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.ArrayList;
import java.util.List;

import android.content.Context;
import android.net.Uri;
import android.content.ContentResolver;
```

```
import android.content.ContentValues;
import android.provider.MediaStore;

public class MainActivity2 extends AppCompatActivity {
    static {
        System.loadLibrary("pytorch_jni");
        System.loadLibrary("pytorch_vision_jni");
    }

    static {
        if (!OpenCVLoader.initDebug()) {
            Log.d("OpenCV", "OpenCV initialization failed");
        } else {
            Log.d("OpenCV", "OpenCV initialized successfully");
        }
    }

    private Module model;
    private ProgressBar loadingSpinner;
    private TextView resultText;
    private ImageView imageView;
    private Button retryButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity2_main);

        if (getSupportActionBar() != null) getSupportActionBar().hide();

        resultText = findViewById(R.id.result_text);
        loadingSpinner = findViewById(R.id.loading_spinner);
        imageView = findViewById(R.id.image_view);
        retryButton = findViewById(R.id.retry_button);
```

```
        try {
            String modelPath = assetFilePath(this,
"resnet18_mobile_3.1.ptl");

            model = LiteModuleLoader.load(modelPath);
        } catch (IOException e) {
            e.printStackTrace();
            resultText.setText("모델 로드 실패");
            return;
        }

        String imagePath = getIntent().getStringExtra("image_path");
        if (imagePath != null) {
            File imgFile = new File(imagePath);
            if (imgFile.exists()) {
                String cleanedPath =
removeExifFromImage(imgFile.getAbsolutePath());
                Bitmap bitmap = BitmapFactory.decodeFile(cleanedPath);
                Bitmap cropped = cropFundusCircle(bitmap);
                imageView.setImageBitmap(cropped);
                saveBitmapForDebug(cropped, "debug_output.jpg");
                classifyEyeDisease(cropped);
            }
        }

        retryButton.setOnClickListener(v -> {
            startActivity(new Intent(MainActivity2.this,
MainActivity1.class));
            finish();
        });
    }

    private void classifyEyeDisease(Bitmap bitmap) {
```

```
        resultText.setText("분석 중입니다...₩n잠시만 기다려주세요");
        retryButton.setVisibility(View.GONE);
        loadingSpinner.setVisibility(View.VISIBLE);

        ExecutorService executor = Executors.newSingleThreadExecutor();
        executor.execute(() -> {
            Tensor inputTensor = TensorImageUtils.bitmapToFloat32Tensor(
                    bitmap,
                    TensorImageUtils.TORCHVISION_NORM_MEAN_RGB,
                    TensorImageUtils.TORCHVISION_NORM_STD_RGB
            );

            // tensor[0,:,0,0] 값 출력 (R, G, B 순서)
            float[] inputData = inputTensor.getDataAsFloatArray();
            int H = 224, W = 224;
            float r0 = inputData[0];
            float g0 = inputData[H * W];
            float b0 = inputData[2 * H * W];
            Log.d("TENSOR_DEBUG", String.format("Tensor[0,:,0,0] → R: %.4f, G: %.4f, B: %.4f", r0, g0, b0));

            Tensor outputTensor = model.forward(IValue.from(inputTensor)).toTensor();
            float[] scores = outputTensor.getDataAsFloatArray();
            float[] probabilities = softmax(scores);

            String[] classes = {"AMD", "Diabetic Retinopathy", "Glaucoma", "Normal"};
            String result = String.format(Locale.US,
                    "AMD: %.1f%%₩nDR: %.1f%%₩nGlaucoma: %.1f%%₩nNormal: %.1f%%",
                    probabilities[0] * 100, probabilities[1] * 100,
```

```
                    probabilities[2] * 100, probabilities[3] * 100);

            runOnUiThread(() -> {
                resultText.setText(result);
                retryButton.setVisibility(View.VISIBLE);
                loadingSpinner.setVisibility(View.GONE);
            });
        });
    }

    private float[] softmax(float[] input) {
        float[] output = new float[input.length];
        float sum = 0;
        for (float val : input) sum += Math.exp(val);
        for (int i = 0; i < input.length; i++) {
            output[i] = (float) (Math.exp(input[i]) / sum);
        }
        return output;
    }

    private Bitmap padToSquare(Bitmap input) {
        int width = input.getWidth();
        int height = input.getHeight();
        int size = Math.max(width, height);
        Bitmap padded = Bitmap.createBitmap(size, size,
input.getConfig());
        Canvas canvas = new Canvas(padded);
        canvas.drawColor(Color.BLACK);
        int left = (size - width) / 2;
        int top = (size - height) / 2;
        canvas.drawBitmap(input, left, top, null);
        return padded;
    }
```

```
public Bitmap cropFundusCircle(Bitmap inputBitmap) {
    Mat src = new Mat();
    Utils.bitmapToMat(inputBitmap, src);
    Imgproc.cvtColor(src, src, Imgproc.COLOR_RGBA2RGB);

    Mat lab = new Mat();
    Imgproc.cvtColor(src, lab, Imgproc.COLOR_RGB2Lab);
    List<Mat> labChannels = new ArrayList<>();
    Core.split(lab, labChannels);

    CLAHE clahe = Imgproc.createCLAHE(1.0, new Size(8, 8));
    clahe.apply(labChannels.get(0), labChannels.get(0));
    Core.merge(labChannels, lab);

    Mat claheRgb = new Mat();
    Imgproc.cvtColor(lab, claheRgb, Imgproc.COLOR_Lab2RGB);

    Mat gray = new Mat();
    Imgproc.cvtColor(claheRgb, gray, Imgproc.COLOR_RGB2GRAY);
    Imgproc.medianBlur(gray, gray, 5);

    int height = gray.rows();
    int width = gray.cols();
    int minR = (int) (Math.min(height, width) * 0.25);
    int maxR = (int) (Math.min(height, width) * 0.6);

    Mat circles = new Mat();
    Imgproc.HoughCircles(gray, circles, Imgproc.HOUGH_GRADIENT,
1.2, 100, 60, 45, minR, maxR);

    Rect roi;
    if (circles.cols() > 0) {
        double[] c = circles.get(0, 0);
        int x = (int) Math.round(c[0]);
```

```
            int y = (int) Math.round(c[1]);
            int r = (int) Math.round(c[2] * 0.96);
            int cropSize = (int) (r * 2.2);
            int x1 = Math.max(0, x - cropSize / 2);
            int y1 = Math.max(0, y - cropSize / 2);
            int x2 = Math.min(width, x + cropSize / 2);
            int y2 = Math.min(height, y + cropSize / 2);
            roi = new Rect(x1, y1, x2 - x1, y2 - y1);
        } else {
            roi = new Rect(0, 0, width, height);
        }

        Mat cropped = new Mat(claheRgb, roi);
        int w = cropped.cols();
        int h = cropped.rows();
        int maxSide = Math.max(w, h);
        int top = (maxSide - h) / 2;
        int bottom = (maxSide - h + 1) / 2;
        int left = (maxSide - w) / 2;
        int right = (maxSide - w + 1) / 2;

        Mat padded = new Mat();
        Core.copyMakeBorder(cropped, padded, top, bottom, left, right,
Core.BORDER_CONSTANT, new Scalar(0, 0, 0));

        Mat resized = new Mat();
        Imgproc.resize(padded, resized, new Size(224, 224));

        Bitmap output = Bitmap.createBitmap(224, 224,
Bitmap.Config.ARGB_8888);
        Utils.matToBitmap(resized, output);
        return output;
    }
```

```
    public static String assetFilePath(Context context, String assetName)
throws IOException {
        File file = new File(context.getFilesDir(), assetName);
        if (file.exists() && file.length() > 0) {
            return file.getAbsolutePath();
        }

        try (InputStream is = context.getAssets().open(assetName);
             OutputStream os = new FileOutputStream(file)) {
            byte[] buffer = new byte[4096];
            int read;
            while ((read = is.read(buffer)) != -1) {
                os.write(buffer, 0, read);
            }
            os.flush();
        }

        return file.getAbsolutePath();
    }

    private String removeExifFromImage(String inputPath) {
        Bitmap bitmap = BitmapFactory.decodeFile(inputPath);
        File tempFile = new File(getCacheDir(), "cleaned_image.jpg");
        try (FileOutputStream out = new FileOutputStream(tempFile)) {
            bitmap.compress(Bitmap.CompressFormat.JPEG, 100, out);
            out.flush();
            return tempFile.getAbsolutePath();
        } catch (IOException e) {
            e.printStackTrace();
            return inputPath;
        }
    }

    private void saveBitmapForDebug(Bitmap bitmap, String filename) {
```

```
        // filename_output
        String finalFilename;
        if (filename.toLowerCase().endsWith(".jpg") ||
filename.toLowerCase().endsWith(".jpeg")) {
            finalFilename = filename.substring(0,
filename.lastIndexOf(".")) + "_output.jpg";
        } else {
            finalFilename = filename + "_output.jpg";
        }

        ContentValues values = new ContentValues();
        values.put(MediaStore.Images.Media.DISPLAY_NAME,
finalFilename);
        values.put(MediaStore.Images.Media.MIME_TYPE, "image/jpeg");
        values.put(MediaStore.Images.Media.RELATIVE_PATH,
Environment.DIRECTORY_PICTURES + "/EyeDebug");

        ContentResolver resolver = getContentResolver();
        Uri uri =
resolver.insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
values);

        if (uri != null) {
            try (OutputStream out = resolver.openOutputStream(uri)) {
                bitmap.compress(Bitmap.CompressFormat.JPEG, 100,
out);
                out.flush();
                Log.d("DEBUG", "이미지 저장 완료 (MediaStore): " +
uri.toString());
            } catch (IOException e) {
                e.printStackTrace();
            }
        } else {
            Log.e("DEBUG", "MediaStore에 URI 생성 실패");
```

```
        }
    }
}
```

6. activity_main.xml (MainActivity1의 UI)

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="eye.application.MainActivity1">


    <ImageView
        android:id="@+id/image_view"
        android:layout_width="226dp"
        android:layout_height="218dp"
        android:layout_marginTop="60dp"
        android:scaleType="fitCenter"
        android:src="@drawable/retina_logo"
        app:layout_constraintBottom_toTopOf="@+id/result_text"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.448"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/result_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="44dp"
```

```
            android:textSize="16sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@id/image_view" />

        <Button
            android:id="@+id/open_gallery_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="갤러리에서 이미지 선택"
            android:fontFamily="@font/pretendard_std_variable"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.498"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@id/result_text" />

        <Button
            android:id="@+id/start_detection_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="56dp"
            android:text="사진을 직접 찍기"
            android:fontFamily="@font/pretendard_std_variable"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.498"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/open_gallery_button"
/>

        <ImageView
            android:id="@+id/univ_logo"
            android:layout_width="120dp"
            android:layout_height="wrap_content"
```

```
        android:layout_marginBottom="4dp"
        android:scaleType="fitCenter"
        android:src="@drawable/univ_logo"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.017"
        app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/univ_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:fontFamily="@font/pretendard_std_variable"
        android:text="Resnet18_mobile_3.0"
        android:textColor="#888888"
        android:textSize="12sp"
        android:textStyle="bold|italic"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.854"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

7. activity2_main.xml (MainActivity2의 UI)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
android:id="@+id/main"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="eye.application.MainActivity2">

<ProgressBar
    android:id="@+id/loading_spinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:visibility="gone"
    android:layout_marginTop="20dp"
    app:layout_constraintTop_toBottomOf="@+id/image_view"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    style="?android:attr/progressBarStyleLarge"
    />

<ImageView
    android:id="@+id/image_view"
    android:layout_width="244dp"
    android:layout_height="237dp"
    android:layout_marginTop="164dp"
    android:scaleType="centerCrop"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.497"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"  />

<TextView
    android:id="@+id/result_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="28dp"
    android:gravity="center"
```

```
        android:text="질병 결과"
        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/image_view" />

    <Button
        android:id="@+id/retry_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="다시 탐지하기"
        android:visibility="gone"
        app:layout_constraintTop_toBottomOf="@+id/result_text"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="16dp" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

8. SplashActivity.java (Splash Screen)

```
//SplashActivity.java
package eye.application;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import androidx.appcompat.app.AppCompatActivity;

public class SplashActivity extends AppCompatActivity {
    private static final int SPLASH_DURATION = 3000;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 상단 상태바 + 하단 내비게이션 바 제거
        getWindow().getDecorView().setSystemUiVisibility(
                View.SYSTEM_UI_FLAG_FULLSCREEN
                        | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
        );

        // 액션바 제거
        if (getSupportActionBar() != null) {
            getSupportActionBar().hide();
        }

        setContentView(R.layout.activity_splash);

        new Handler().postDelayed(() -> {
            Intent intent = new Intent(SplashActivity.this,
MainActivity1.class);
            startActivity(intent);
            finish();
        }, SPLASH_DURATION);
    }
}
```

9. activity_splash.java (Splash Screen의 UI)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:background="#005BAC"
    android:gravity="center">

    <ImageView
        android:id="@+id/logo"
        android:layout_width="212dp"
        android:layout_height="181dp"
        android:contentDescription="앱 로고"
        android:src="@drawable/white_logo" />
</RelativeLayout>
```

10. build.gradle(:app)

```
//build.gradle(:app)
plugins {
    alias(libs.plugins.android.application)
}

android {
    namespace = "eye.application"
    compileSdk = 35

    defaultConfig {
        applicationId = "eye.application"
        minSdk = 26
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"
        testInstrumentationRunner =
"androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
```

```
            minifyEnabled = true
            shrinkResources = true
            proguardFiles(

getDefaultProguardFile("proguard-android-optimize.txt"),
                    "proguard-rules.pro"
            )
        }
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }

    // R 파일 인식 안 되는 문제 해결
    sourceSets {
        getByName("main") {
            java.srcDirs("src/main/java")
            res.srcDirs("src/main/res")
            manifest.srcFile("src/main/AndroidManifest.xml")
            jniLibs.srcDirs("src/main/jniLibs")
        }
    }
}

dependencies {

    implementation 'org.pytorch:pytorch_android_lite:1.13.1'
    implementation 'org.pytorch:pytorch_android_torchvision_lite:1.13.1'

    implementation(libs.appcompat)
    implementation(libs.material)
    implementation(libs.activity)
```

```
    implementation(libs.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
    implementation(project(":OpenCVLibrary411"))

    implementation("androidx.exifinterface:exifinterface:1.4.1")
}
```

11. build.gradle(:OpenCVLibrary411)

```
//build.gradle(:OpenCVLibrary411)
apply plugin: 'com.android.library'

android {
    namespace 'org.opencv'
    compileSdkVersion 35

    defaultConfig {
        minSdkVersion 21
        targetSdkVersion 35
    }

    sourceSets {
        main {
            java.srcDirs = ['src/main/java']
            res.srcDirs = ['src/main/res']
            manifest.srcFile 'src/main/AndroidManifest.xml'
        }
    }
}

dependencies {
```

```
    implementation libs.constraintlayout
}
```

12. string.xml (apk 이름)

```
<resources>
    <string name="app_name">RetinaScan_V1</string>
</resources>
```

13. AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="eye.application">

    <uses-feature
        android:name="android.hardware.camera"
        android:required="false" />

    <uses-permission android:name="android.permission.CAMERA" />

    <queries>
        <intent>
            <action android:name="android.media.action.IMAGE_CAPTURE"
/>
        </intent>
    </queries>

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
```

```
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.EyeDiagnosisApp"
        tools:targetApi="31">

        <!-- 파일 접근 권한 설정 -->
        <provider
            android:name="androidx.core.content.FileProvider"
            android:authorities="eye.application.fileprovider"
            android:grantUriPermissions="true"
            android:exported="false">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/file_paths" />
        </provider>

        <!--  앱 시작 화면 (Splash) -->
        <activity
            android:name="eye.application.SplashActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!-- 기존 메인 액티비티 -->
        <activity
            android:name="eye.application.MainActivity1"
            android:exported="true" />
```

```
        〈activity
            android:name="eye.application.MainActivity2"
            android:exported="true"
            android:parentActivityName="eye.application.MainActivity1" /〉

    〈/application〉

〈/manifest〉
```

14. GradCam ++

```
# 1. Google Drive 마운트
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# 2. torchcam 설치 (Grad-CAM++ 지원용)
#!pip uninstall -y numpy
#!pip install numpy==1.23.5
!pip install -q torchcam

import numpy as np
print(np.__version__)

# 3. 라이브러리 임포트
import torch
import torch.nn as nn
from torchvision import models, transforms
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
import cv2
from torchcam.methods import GradCAMpp

# 4. 이미지 전처리 정의
```

```
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# 5. 모델 정의 (ResNet18 + Custom FC)
model = models.resnet18()
model.fc = nn.Sequential(
    nn.Dropout(0.6),
    nn.Linear(512, 256),
    nn.BatchNorm1d(256),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(256, 4)  # 클래스 수: 4
)

# 6. 학습된 모델 가중치 로드 (.pth → state_dict)
model.load_state_dict(torch.load("/content/drive/MyDrive/resnet18_mobile_
3.1.pth", map_location='cpu'))
model.eval()

# 7. Grad-CAM++ 객체 생성
cam_extractor = GradCAMpp(model, target_layer="layer4.1.conv2")

# 8. 테스트 이미지 불러오기
image_path                                                              =
"/content/drive/MyDrive/seg_dataset2_Alpha/train/amd/614_right.jpg"
image_pil = Image.open(image_path).convert("RGB")
input_tensor = transform(image_pil).unsqueeze(0)

# OpenCV용 시각화 이미지 (BGR + uint8 + 3채널 보장)
image_np = np.array(image_pil.resize((224, 224)).convert("RGB"))
```

```
image_np = cv2.cvtColor(image_np, cv2.COLOR_RGB2BGR).astype(np.uint8)

# 9. 모델 추론 + CAM 추출
output = model(input_tensor)
pred_class = output.argmax(dim=1).item()
cam = cam_extractor(pred_class, output)[0]  # 첫 번째 CAM만 추출

# 10. CAM 후처리 (정규화 + 업샘플링)
if isinstance(cam, torch.Tensor):
    cam = cam.detach().cpu().numpy()

cam = (cam - cam.min()) / (cam.max() - cam.min() + 1e-8)  # [0,1]
cam = np.uint8(cam * 255)  # [0,255]
if cam.ndim == 3:
    cam = cam.squeeze()

# 업샘플링: 7x7 → 224x224로 크기 맞추기
cam = cv2.resize(cam, (224, 224), interpolation=cv2.INTER_LINEAR)

# 11. Heatmap 및 Overlay 생성
heatmap = cv2.applyColorMap(cam, cv2.COLORMAP_JET)
heatmap = cv2.convertScaleAbs(heatmap, alpha=1.5)
overlay = cv2.addWeighted(image_np, 0.45, heatmap, 0.55, 0)

# 12. 시각화 출력
plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)
plt.imshow(cv2.cvtColor(image_np, cv2.COLOR_BGR2RGB))
plt.title("Original Image")
plt.axis('off')

plt.subplot(1, 3, 2)
```

```
plt.imshow(cam, cmap='jet')
plt.title("Grad-CAM++")
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(overlay, cv2.COLOR_BGR2RGB))
plt.title("Overlay")
plt.axis('off')

plt.tight_layout()
plt.show()
```

15. Confusion_Matrix(Resnet18)

```
//Confusion_Matrix(Resnet18)
import os
import torch
import numpy as np
import matplotlib.pyplot as plt
from torchvision import transforms, models
from PIL import Image
from tqdm import tqdm
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
classification_report
import torch.nn as nn
from google.colab import drive

# Google Drive 마운트
drive.mount('/content/drive', force_remount=True)

# 경로 설정
TEST_DIR = "/content/drive/MyDrive/seg_dataset2_Alpha/valid"
MODEL_PATH = "/content/drive/MyDrive/resnet18_mobile_3.1.pth"
```

```
# 폴더 이름 기반 클래스 매핑
folder_names = sorted(os.listdir(TEST_DIR))
label_map = {
    'amd': 0,
    'diabetic_retinopathy': 1,
    'glaucoma': 2,
    'normal': 3
}
class_names = ['AMD', 'DR', 'GLC', 'NML']
print("[INFO] label_map 생성됨:", label_map)
print("[DEBUG] 실제 폴더명:", folder_names)

# 폴더 존재 여부 및 이미지 수 디버깅
for folder_name in folder_names:
    folder_path = os.path.join(TEST_DIR, folder_name)
    exists = os.path.exists(folder_path)
    files = os.listdir(folder_path) if exists else []
    print(f"[CHECK] {folder_name} → exists: {exists}, images: {len(files)}")

# 이미지 전처리 (학습과 동일)
transform = transforms.Compose([
    transforms.Resize(224),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# 디바이스 설정
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"[INFO] 사용 장치: {device}")

# 모델 구성 및 가중치 로딩
```

```
model = models.resnet18(weights=None)
model.fc = nn.Sequential(
    nn.Dropout(0.6),
    nn.Linear(512, 256),
    nn.BatchNorm1d(256),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(256, len(class_names))
)
model.load_state_dict(torch.load(MODEL_PATH, map_location=device))
model.to(device)
model.eval()

# 예측 수행
y_true, y_pred = [], []

with torch.no_grad():
    for folder_name, label_idx in label_map.items():
        folder_path = os.path.join(TEST_DIR, folder_name)
        image_files = [f for f in os.listdir(folder_path) if
f.lower().endswith(('.jpg', '.jpeg', '.png'))]

        print(f"[DEBUG] folder: {folder_name}, label: {label_idx}, images:
{len(image_files)}")

        for img_file in tqdm(image_files, desc=folder_name, unit="img"):
            img_path = os.path.join(folder_path, img_file)

            try:
                image = Image.open(img_path).convert("RGB")
                input_tensor                                                =
transform(image).unsqueeze(0).to(device)

                output = model(input_tensor)
```

```
                pred = torch.argmax(output, dim=1).item()

                y_true.append(label_idx)
                y_pred.append(pred)
            except Exception as e:
                print(f"[ERROR] {folder_name}/{img_file} → {e}")
                continue

# Confusion Matrix 출력
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=class_names)
plt.figure(figsize=(6, 6))
disp.plot(cmap='Blues', values_format='d')
plt.title("Confusion Matrix:Resnet_18")
plt.show()

# Classification Report 출력
print("₩nClassification Report:Resnet_18")
print(classification_report(y_true, y_pred, target_names=class_names,
digits=4))
```

16. Confusion_Matrix(Efficientnet_B0)

```
//Confusion_Matrix(Efficientnet_B0)
import os
import torch
import numpy as np
import matplotlib.pyplot as plt
from torchvision import transforms
from torchvision.models import efficientnet_b0, EfficientNet_B0_Weights
from PIL import Image
from tqdm import tqdm
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
import torch.nn as nn
from google.colab import drive

# Google Drive 마운트
drive.mount('/content/drive', force_remount=True)

# 경로 설정
TEST_DIR = "/content/drive/MyDrive/seg_dataset2_Alpha/valid"
MODEL_PATH = "/content/drive/MyDrive/efficientnetB0_mobile_1.0.pth"

# 클래스 매핑
label_map = {
    'amd': 0,
    'diabetic_retinopathy': 1,
    'glaucoma': 2,
    'normal': 3
}
class_names = ['AMD', 'DR', 'GLC', 'NML']

# 폴더 확인
folder_names = sorted(os.listdir(TEST_DIR))
print("[INFO] label_map 생성됨:", label_map)
print("[DEBUG] 실제 폴더명:", folder_names)
for folder_name in folder_names:
    folder_path = os.path.join(TEST_DIR, folder_name)
    exists = os.path.exists(folder_path)
    files = os.listdir(folder_path) if exists else []
    print(f"[CHECK] {folder_name} → exists: {exists}, images: {len(files)}")

# 이미지 전처리
transform = transforms.Compose([
```

```
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# 디바이스 설정
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"[INFO] 사용 장치: {device}")

# EfficientNet-B0 모델 정의 및 가중치 로딩
weights = EfficientNet_B0_Weights.DEFAULT
model = efficientnet_b0(weights=weights)  # pretrained 구조

# classifier 구조 재정의 (학습 시와 동일하게!)
num_ftrs = model.classifier[1].in_features
model.classifier = nn.Sequential(
    nn.Dropout(0.6),
    nn.Linear(num_ftrs, 256),
    nn.BatchNorm1d(256),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(256, len(class_names))  # 4-class
)

# 모델 가중치 로딩
model.load_state_dict(torch.load(MODEL_PATH, map_location=device))
model.to(device)
model.eval()

# 예측 수행
y_true, y_pred = [], []
```

```
with torch.no_grad():
    for folder_name, label_idx in label_map.items():
        folder_path = os.path.join(TEST_DIR, folder_name)
        image_files = [f for f in os.listdir(folder_path) if f.lower().endswith(('.jpg', '.jpeg', '.png'))]

        print(f"[DEBUG] folder: {folder_name}, label: {label_idx}, images: {len(image_files)}")

        for img_file in tqdm(image_files, desc=folder_name, unit="img"):
            img_path = os.path.join(folder_path, img_file)

            try:
                image = Image.open(img_path).convert("RGB")
                input_tensor = transform(image).unsqueeze(0).to(device)

                output = model(input_tensor)
                pred = torch.argmax(output, dim=1).item()

                y_true.append(label_idx)
                y_pred.append(pred)
            except Exception as e:
                print(f"[ERROR] {folder_name}/{img_file} → {e}")
                continue

# Confusion Matrix 출력
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
plt.figure(figsize=(6, 6))
disp.plot(cmap='Blues', values_format='d')
plt.title("Confusion Matrix:Efficientnet_B0")
plt.show()
```

```
# Classification Report 출력
print("₩nClassification Report:Efficientnet_B0")
print(classification_report(y_true, y_pred, target_names=class_names, digits=4))
```

17. Confusion_Matrix(Mobilenet_V2)

```
#Confusion_Matrix(Mobilenet_V2)
import os
import torch
import numpy as np
import matplotlib.pyplot as plt
from torchvision import transforms
from torchvision.models import mobilenet_v2, MobileNet_V2_Weights
from PIL import Image
from tqdm import tqdm
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
import torch.nn as nn
from google.colab import drive

# Google Drive 마운트
drive.mount('/content/drive', force_remount=True)

# 경로 설정
TEST_DIR = "/content/drive/MyDrive/seg_dataset2_Alpha/valid"
MODEL_PATH = "/content/drive/MyDrive/mobilenetv2_mobile_1.0.pth"  # MobileNetV2 가중치 경로

# 클래스 매핑
label_map = {
    'amd': 0,
    'diabetic_retinopathy': 1,
```

```
    'glaucoma': 2,
    'normal': 3
}
class_names = ['AMD', 'DR', 'GLC', 'NML']

# 폴더 확인
folder_names = sorted(os.listdir(TEST_DIR))
print("[INFO] label_map 생성됨:", label_map)
print("[DEBUG] 실제 폴더명:", folder_names)
for folder_name in folder_names:
    folder_path = os.path.join(TEST_DIR, folder_name)
    exists = os.path.exists(folder_path)
    files = os.listdir(folder_path) if exists else []
    print(f"[CHECK] {folder_name} → exists: {exists}, images: {len(files)}")

# 이미지 전처리
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# 디바이스 설정
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"[INFO] 사용 장치: {device}")

# MobileNetV2 모델 정의 및 가중치 로딩
weights = MobileNet_V2_Weights.DEFAULT
model = mobilenet_v2(weights=weights)

# classifier 구조 재정의 (학습 시와 동일하게!)
```

```
num_ftrs = model.classifier[1].in_features
model.classifier = nn.Sequential(
    nn.Dropout(0.6),
    nn.Linear(num_ftrs, 256),
    nn.BatchNorm1d(256),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(256, len(class_names))  # 4-class
)

# 모델 가중치 로딩
model.load_state_dict(torch.load(MODEL_PATH, map_location=device))
model.to(device)
model.eval()

# 예측 수행
y_true, y_pred = [], []

with torch.no_grad():
    for folder_name, label_idx in label_map.items():
        folder_path = os.path.join(TEST_DIR, folder_name)
        image_files = [f for f in os.listdir(folder_path) if
f.lower().endswith(('.jpg', '.jpeg', '.png'))]

        print(f"[DEBUG] folder: {folder_name}, label: {label_idx}, images:
{len(image_files)}")

        for img_file in tqdm(image_files, desc=folder_name, unit="img"):
            img_path = os.path.join(folder_path, img_file)

            try:
                image = Image.open(img_path).convert("RGB")
                input_tensor                                              =
transform(image).unsqueeze(0).to(device)
```

```
                output = model(input_tensor)
                pred = torch.argmax(output, dim=1).item()

                y_true.append(label_idx)
                y_pred.append(pred)
            except Exception as e:
                print(f"[ERROR] {folder_name}/{img_file} → {e}")
                continue

# Confusion Matrix 출력
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=class_names)
plt.figure(figsize=(6, 6))
disp.plot(cmap='Blues', values_format='d')
plt.title("Confusion Matrix:Mobilenet_V2")
plt.show()

# Classification Report 출력
print("₩nClassification Report:Mobilenet_V2")
print(classification_report(y_true, y_pred, target_names=class_names,
digits=4))
```

18. AUC-ROC (Resnet-18)

```
# AUC-ROC_Resnet-18
import os
import torch
import numpy as np
import matplotlib.pyplot as plt
from torchvision import transforms, models
from PIL import Image
from tqdm import tqdm
```

```
import torch.nn as nn
from google.colab import drive
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize

# Google Drive 마운트
drive.mount('/content/drive', force_remount=True)

# 경로 설정
TEST_DIR = "/content/drive/MyDrive/seg_dataset2_Alpha/train"
MODEL_PATH = "/content/drive/MyDrive/resnet18_mobile_3.1.pth"
SAVE_PATH = "/content/drive/MyDrive/resnet18_roc_train.png"

# 클래스 정의
label_map = {
    'amd': 0,
    'diabetic_retinopathy': 1,
    'glaucoma': 2,
    'normal': 3
}
class_names = ['AMD', 'DR', 'GLC', 'NML']

# 이미지 전처리 (학습과 동일하게 유지)
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# 디바이스 설정
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"[INFO] 사용 장치: {device}")
```

```
# ResNet18 모델 정의 및 가중치 로딩
model = models.resnet18(weights=None)
model.fc = nn.Sequential(
    nn.Dropout(0.6),
    nn.Linear(512, 256),
    nn.BatchNorm1d(256),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(256, len(class_names))
)
model.load_state_dict(torch.load(MODEL_PATH, map_location=device))
model.to(device)
model.eval()

# 예측 수행
y_true = []
y_score = []

with torch.no_grad():
    for folder_name, label_idx in label_map.items():
        folder_path = os.path.join(TEST_DIR, folder_name)
        image_files = [f for f in os.listdir(folder_path) if
f.lower().endswith(('.jpg', '.jpeg', '.png'))]

        print(f"[DEBUG] folder: {folder_name}, label: {label_idx}, images:
{len(image_files)}")

        for img_file in tqdm(image_files, desc=folder_name, unit="img"):
            img_path = os.path.join(folder_path, img_file)

            try:
                image = Image.open(img_path).convert("RGB")
                input_tensor                                    =
```

```
transform(image).unsqueeze(0).to(device)

                output = model(input_tensor)
                prob = torch.softmax(output, dim=1).cpu().numpy()[0]

                y_true.append(label_idx)
                y_score.append(prob)
            except Exception as e:
                print(f"[ERROR] {folder_name}/{img_file} → {e}")
                continue

# ROC Curve 계산
y_true_bin = label_binarize(y_true, classes=[0, 1, 2, 3])
y_score = np.array(y_score)

# 시각화 및 저장
plt.figure(figsize=(8, 6))
colors = ['red', 'blue', 'green', 'orange']

for i in range(n_classes):
    fpr, tpr, _ = roc_curve(y_true_bin[:, i], y_score[:, i])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, color=colors[i], lw=2,
             label=f"{class_names[i]} (AUC = {roc_auc:.4f})")

plt.plot([0, 1], [0, 1], 'k--', lw=1)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve (ResNet18)")
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
```

```
# ROC Curve 이미지 저장
plt.savefig(SAVE_PATH)
plt.show()
print(f"[SAVED] ROC Curve saved to: {SAVE_PATH}")
```

19. AUC-ROC (Efficientnet-B0)

```
#AUC-ROC_Efficientnet-B0
import os
import torch
import numpy as np
import matplotlib.pyplot as plt
from torchvision import transforms
from torchvision.models import efficientnet_b0, EfficientNet_B0_Weights
from PIL import Image
from tqdm import tqdm
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_curve, auc
import torch.nn as nn
from google.colab import drive

# Google Drive 마운트
drive.mount('/content/drive', force_remount=True)

# 경로 설정
TEST_DIR = "/content/drive/MyDrive/seg_dataset2_Alpha/train"
MODEL_PATH = "/content/drive/MyDrive/efficientnetB0_mobile_1.0.pth"
SAVE_PATH = "/content/drive/MyDrive/efficientnet_roc_train.png"

# 클래스 정의
label_map = {
    'amd': 0,
```

```
    'diabetic_retinopathy': 1,
    'glaucoma': 2,
    'normal': 3
}
class_names = ['AMD', 'DR', 'GLC', 'NML']
n_classes = len(class_names)

# 이미지 전처리
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# 디바이스 설정
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"[INFO] 사용 장치: {device}")

# 모델 정의 및 가중치 로딩
weights = EfficientNet_B0_Weights.DEFAULT
model = efficientnet_b0(weights=weights)
num_ftrs = model.classifier[1].in_features
model.classifier = nn.Sequential(
    nn.Dropout(0.6),
    nn.Linear(num_ftrs, 256),
    nn.BatchNorm1d(256),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(256, n_classes)
)
model.load_state_dict(torch.load(MODEL_PATH, map_location=device))
model.to(device)
```

```
model.eval()

# 예측 수행
y_true = []
y_score = []

with torch.no_grad():
    for folder_name, label_idx in label_map.items():
        folder_path = os.path.join(TEST_DIR, folder_name)
        image_files = [f for f in os.listdir(folder_path) if f.lower().endswith(('.jpg', '.jpeg', '.png'))]

        for img_file in tqdm(image_files, desc=f"[ROC] {folder_name}", unit="img"):
            img_path = os.path.join(folder_path, img_file)
            try:
                image = Image.open(img_path).convert("RGB")
                input_tensor = transform(image).unsqueeze(0).to(device)

                output = model(input_tensor)
                prob = torch.softmax(output, dim=1).cpu().numpy()[0]

                y_true.append(label_idx)
                y_score.append(prob)
            except Exception as e:
                print(f"[ERROR ] {folder_name}/{img_file} → {e}")
                continue

# ROC Curve 계산
y_true_bin = label_binarize(y_true, classes=list(label_map.values()))
y_score = np.array(y_score)

# 시각화 및 저장
```

```
plt.figure(figsize=(8, 6))
colors = ['red', 'blue', 'green', 'orange']

for i in range(n_classes):
    fpr, tpr, _ = roc_curve(y_true_bin[:, i], y_score[:, i])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, color=colors[i], lw=2,
             label=f"{class_names[i]} (AUC = {roc_auc:.4f})")

plt.plot([0, 1], [0, 1], 'k--', lw=1)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve (EfficientNetB0)")
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()

# ROC Curve 이미지 저장
plt.savefig(SAVE_PATH)
plt.show()
print(f"[SAVED] ROC Curve saved to: {SAVE_PATH}")
```

20. AUC-ROC(Mobilenet-V2)

```
#AUC-ROC_Mobilenet-V2
import os
import torch
import numpy as np
import matplotlib.pyplot as plt
from torchvision import transforms
from torchvision.models import mobilenet_v2, MobileNet_V2_Weights
from PIL import Image
```

```
from tqdm import tqdm
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_curve, auc
import torch.nn as nn
from google.colab import drive

# Google Drive 마운트
drive.mount('/content/drive', force_remount=True)

# 경로 설정
TEST_DIR = "/content/drive/MyDrive/seg_dataset2_Alpha/train"
MODEL_PATH = "/content/drive/MyDrive/mobilenetv2_mobile_1.0.pth"
SAVE_PATH = "/content/drive/MyDrive/mobilenetv2_roc_train.png"

# 클래스 정의
label_map = {
    'amd': 0,
    'diabetic_retinopathy': 1,
    'glaucoma': 2,
    'normal': 3
}
class_names = ['AMD', 'DR', 'GLC', 'NML']
n_classes = len(class_names)

# 이미지 전처리
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# 디바이스 설정
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"[INFO] 사용 장치: {device}")

# 모델 정의 및 가중치 로딩
weights = MobileNet_V2_Weights.DEFAULT
model = mobilenet_v2(weights=weights)
num_ftrs = model.classifier[1].in_features
model.classifier = nn.Sequential(
    nn.Dropout(0.6),
    nn.Linear(num_ftrs, 256),
    nn.BatchNorm1d(256),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(256, n_classes)
)
model.load_state_dict(torch.load(MODEL_PATH, map_location=device))
model.to(device)
model.eval()

# 예측 수행
y_true = []
y_scores = []

with torch.no_grad():
    for folder_name, label_idx in label_map.items():
        folder_path = os.path.join(TEST_DIR, folder_name)
        image_files = [f for f in os.listdir(folder_path) if
f.lower().endswith(('.jpg', '.jpeg', '.png'))]

        for img_file in tqdm(image_files, desc=folder_name, unit="img"):
            img_path = os.path.join(folder_path, img_file)
            try:
                image = Image.open(img_path).convert("RGB")
                input_tensor                                    =
```

```
transform(image).unsqueeze(0).to(device)

                output = model(input_tensor)
                prob = torch.softmax(output, dim=1).cpu().numpy()[0]

                y_true.append(label_idx)
                y_scores.append(prob)
            except Exception as e:
                print(f"[ERROR] {folder_name}/{img_file} → {e}")
                continue

# ROC Curve 계산
y_true_bin = label_binarize(y_true, classes=[0, 1, 2, 3])
y_scores = np.array(y_scores)

# 시각화 및 저장
plt.figure(figsize=(8, 6))
colors = ['red', 'blue', 'green', 'orange']

for i in range(n_classes):
    fpr, tpr, _ = roc_curve(y_true_bin[:, i], y_scores[:, i])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, color=colors[i], lw=2,
             label=f"{class_names[i]} (AUC = {roc_auc:.4f})")

plt.plot([0, 1], [0, 1], 'k--', lw=1)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve (MobileNetV2)")
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
```

```
# ROC Curve 이미지 저장
plt.savefig(SAVE_PATH)
plt.show()
print(f"[SAVED] ROC Curve saved to: {SAVE_PATH}")
```

21. t-SNE

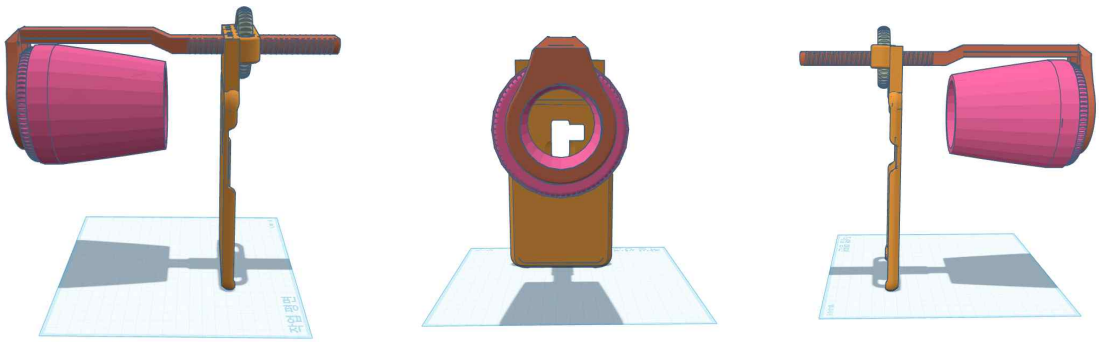
```
# t-SNE
import os
import torch
import numpy as np
import matplotlib.pyplot as plt
from torchvision import transforms, models
from PIL import Image
from tqdm import tqdm
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import pandas as pd
import torch.nn as nn
from google.colab import drive

# Google Drive 마운트
drive.mount('/content/drive', force_remount=True)

# 데이터셋 및 모델 경로 설정
TEST_DIR = "/content/drive/MyDrive/seg_dataset2_Alpha/train"
MODEL_PATH = "/content/drive/MyDrive/resnet18_mobile_3.1.pth"

# 클래스 이름 및 라벨 매핑
label_map = {
    'amd': 0,
```

```
    'diabetic_retinopathy': 1,
    'glaucoma': 2,
    'normal': 3
}
class_names = ['AMD', 'DR', 'GLC', 'NML']

# 이미지 전처리 (학습과 동일한 방식 유지)
transform = transforms.Compose([
    transforms.Resize(224),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# 디바이스 설정
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"[INFO] 사용 디바이스: {device}")

# ResNet18 기반 커스텀 모델 구성
model = models.resnet18(weights=None)
model.fc = nn.Sequential(
    nn.Dropout(0.6),
    nn.Linear(512, 256),
    nn.BatchNorm1d(256),
    nn.ReLU(),
    nn.Dropout(0.4),
    nn.Linear(256, len(class_names))
)
model.load_state_dict(torch.load(MODEL_PATH, map_location=device))
model.to(device)
model.eval()

# 특징 추출기 구성 (FC 레이어 전까지)
```

```
feature_extractor = torch.nn.Sequential(*list(model.children())[:-1])
feature_extractor.to(device)
feature_extractor.eval()

# 특징 벡터 및 레이블 수집
features = []
labels = []

with torch.no_grad():
    for folder_name, label_idx in label_map.items():
        folder_path = os.path.join(TEST_DIR, folder_name)
        image_files = [f for f in os.listdir(folder_path) if
f.lower().endswith(('.jpg', '.jpeg', '.png'))]

        for img_file in tqdm(image_files, desc=f"{folder_name}",
unit="img"):  # 전체 train 이미지
            img_path = os.path.join(folder_path, img_file)
            try:
                image = Image.open(img_path).convert("RGB")
                input_tensor                                    =
transform(image).unsqueeze(0).to(device)
                feature = feature_extractor(input_tensor)
                feature          =          feature.view(feature.size(0),
-1).cpu().numpy()  # (1, 512)
                features.append(feature[0])
                labels.append(label_idx)
            except Exception as e:
                print(f"[ERROR] {folder_name}/{img_file} → {e}")
                continue

# t-SNE로 차원 축소 (2D)
features = StandardScaler().fit_transform(features)
tsne = TSNE(
    n_components=2,
```

```
        perplexity=30,
        learning_rate=1500,
        early_exaggeration=100,
        n_iter=1500,
        init='pca',
        random_state=42
)
features_2d = tsne.fit_transform(features)

# 시각화용 DataFrame 생성
df = pd.DataFrame({
        'x': features_2d[:, 0],
        'y': features_2d[:, 1],
        'label': [class_names[i] for i in labels]
})

# 2D t-SNE 시각화 출력
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='x', y='y', hue='label', palette='Set2', s=60,
edgecolor='black')
plt.title("t-SNE Visualization of Feature Embeddings (ResNet-18)",
fontsize=14)
plt.xlabel("Component 1")
plt.ylabel("Component 2")
plt.legend(title="Class", loc='best')
plt.grid(True)
plt.tight_layout()
plt.show()
```

22. 3D 프린팅 스마트폰 어댑터

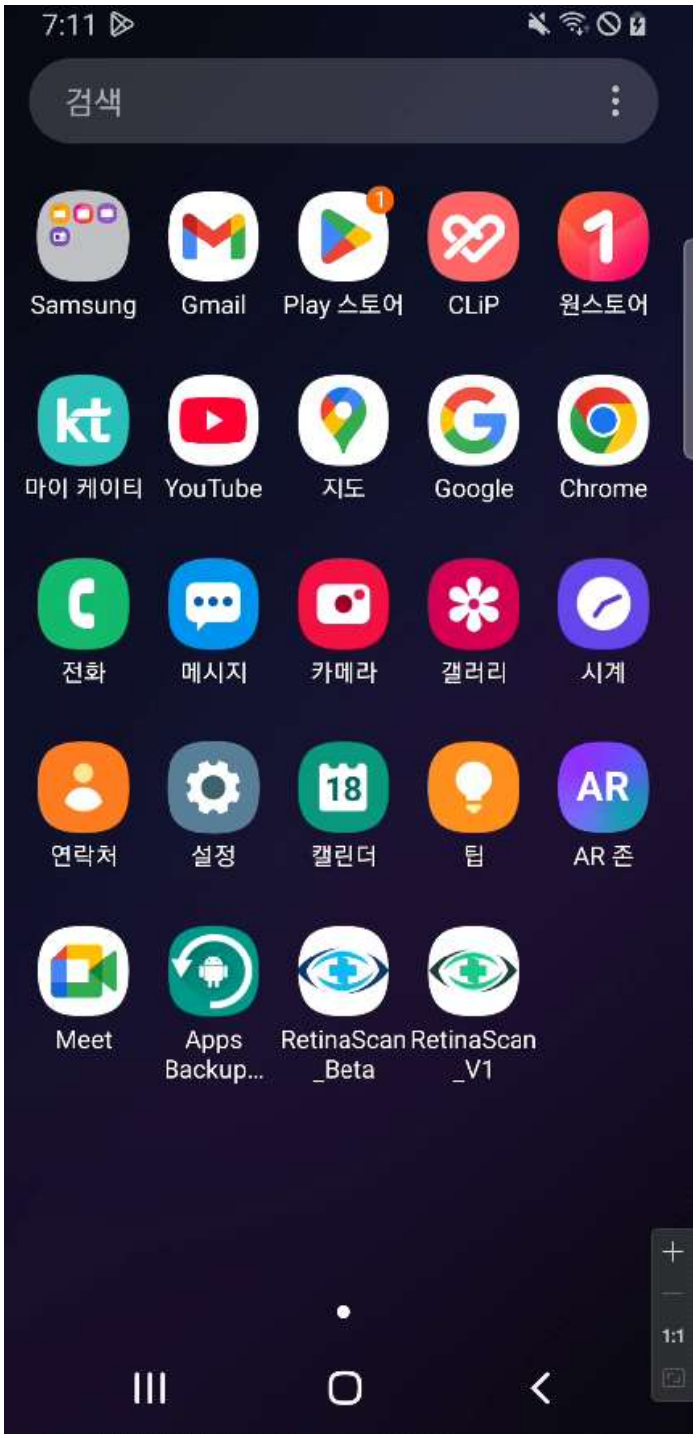


VII. 결과물 설명 (결과물 관련 이미지(사진) 첨부)

1. 하드웨어

하드웨어는 3D 프린팅을 한 모델로, Phone case, Screwknob, Lenscover, Centralbeam 총 4가지로 구성되어 있다. 각각의 3D 프린팅 한 모델은 조립식으로 쉽게 조립이 가능하며, Lenscover와 Screwknob 사이에 20D Fundus lens가 들어간다.

2. Smartphone Application

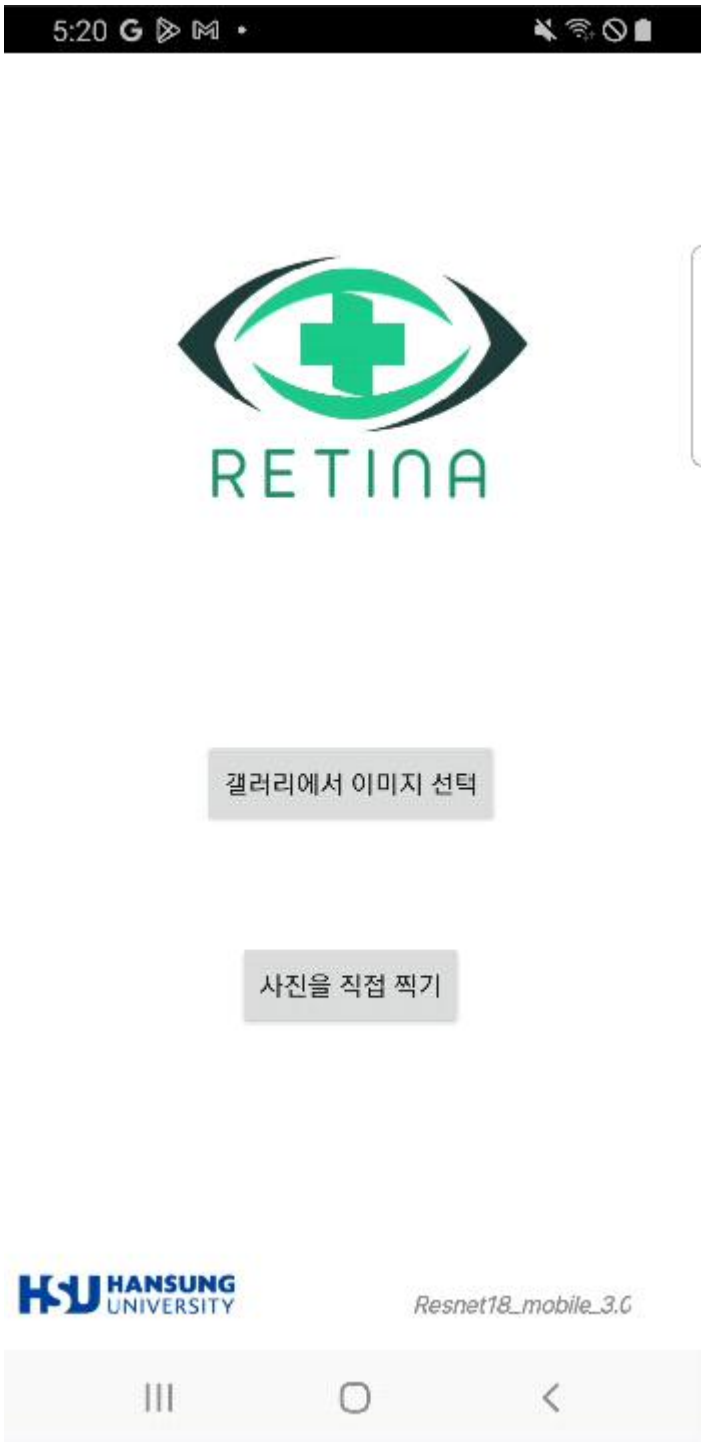


다음은 실제 만들어진 application으로, 팀 로고가 application의 대표 이미지이다. 좌측의 Beta 버전은 우리가 적용한 전처리를 넣기 전의 apk 파일이다.

3. Smartphone Application의 Splash Screen

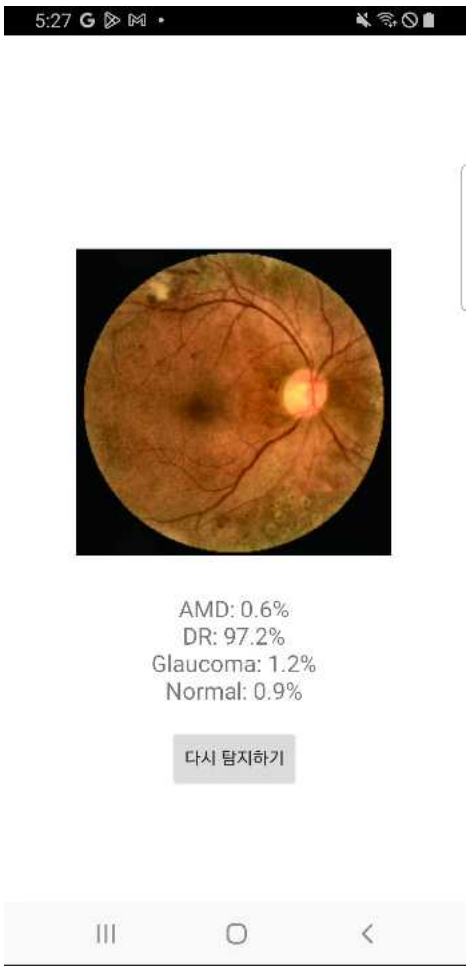
앱 실행시 나오는 Splash Screen으로 앱의 정체성과 팀 로고의 이미지를 강조되어 보여진다.

4. Smartphone Application의 MainActivity1 화면



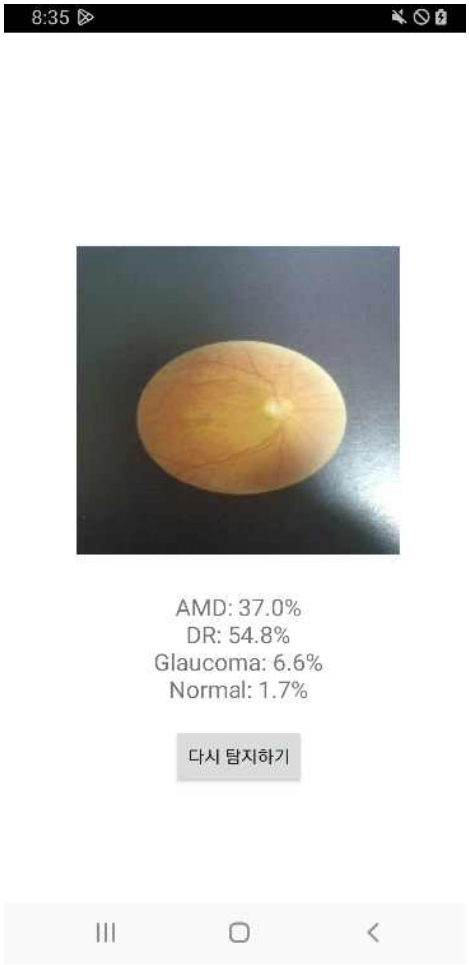
앱 실행시 나오는 MainActivity1 화면으로, 사용자는 '갤러리에서 이미지 선택' 혹은 '사진을 직접 찍기' 버튼을 눌러 원하는 동작을 실행시킬 수 있다.

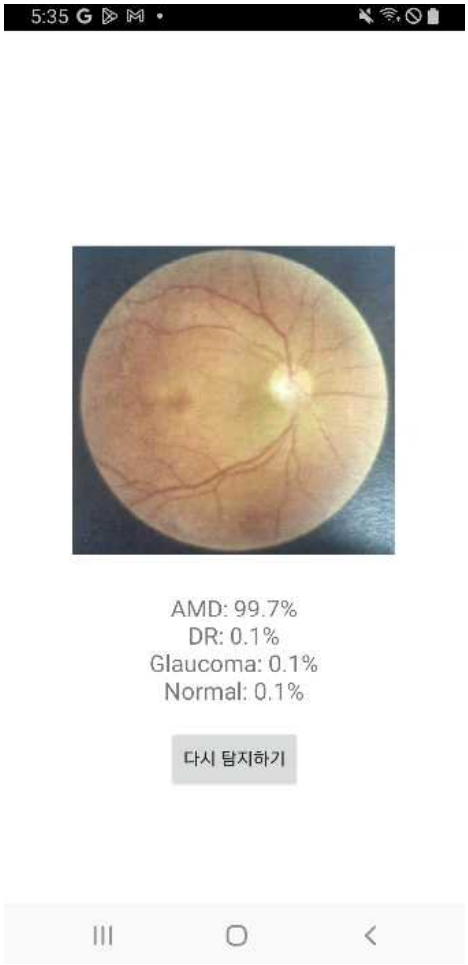
4-1. 갤러리에서 이미지 선택시의 MainActivity2 화면



사용자가 '갤러리에서 이미지 선택'을 누르면, 갤러리에서 원하는 안저사진을 선택할 수 있으며, 사진 선택 후 결과 값을 잠시 뒤에 확인할 수 있다. '다시 탐지하기' 버튼을 누르면 다시 MainActivity1으로 회귀한다.

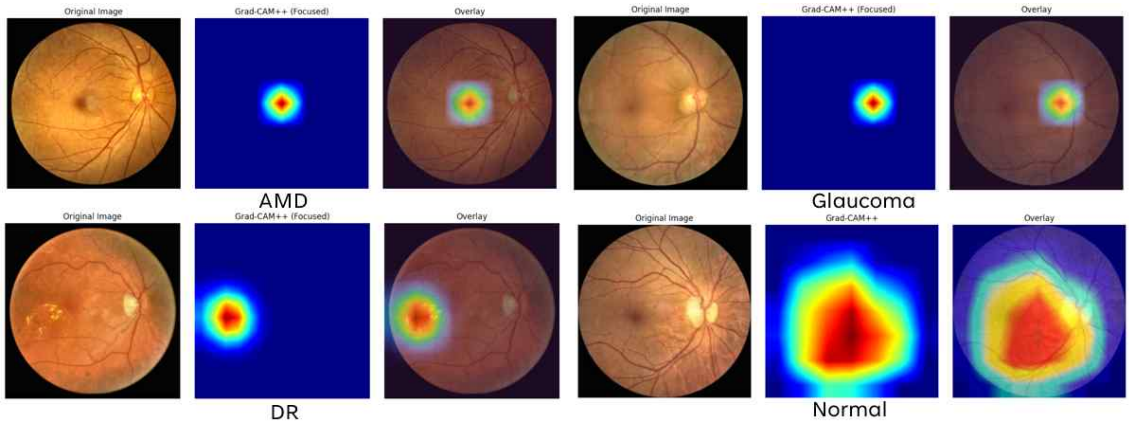
4-2. 사진을 직접 찍을 시의 MainActivity2 화면





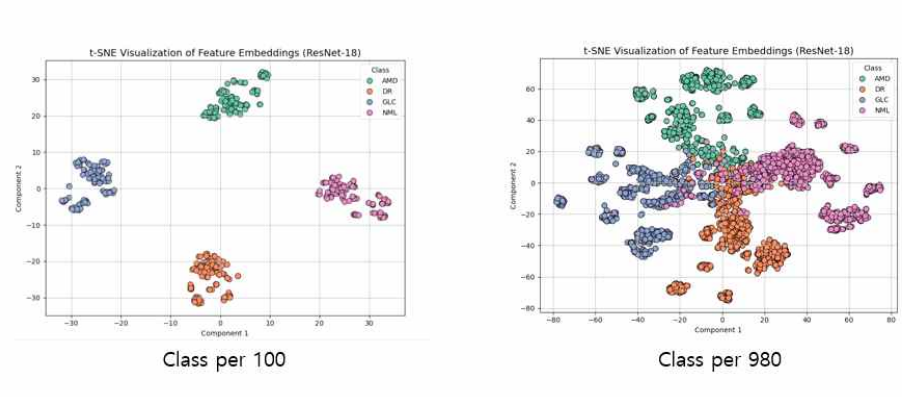
사용자가 '사진을 직접 찍기'를 누르면, 카메라 기능이 작동되며, 결과 값을 잠시 뒤에 확인할 수 있다. '다시 탐지하기' 버튼을 누르면 다시 MainActivity1으로 회귀한다. 위의 사진은 AMD 사진으로, 동일한 조건에서 사진을 촬영하였다. 왼쪽의 사진은 우리가 만든 전처리를 이용하기 전 모습으로, 제대로 Crop이 안 될 뿐만이 아니라 병명조차도 틀리는 모습을 보여준다.

5. Grad-Cam ++ 시각화 (Colab)



Grad-CAM++은 딥러닝된 모델이 해당 질병별로 집중하는 구역을 시각화한다. 이는 모델의 투명성과 신뢰성을 향상시키며, 임상적으로 중요한 영역을 더욱 정밀하게 강조해줌으로써 안저 영상을 활용한 안구 질환 분류와 같은 의료 영상 분석에 특히 적합하다.

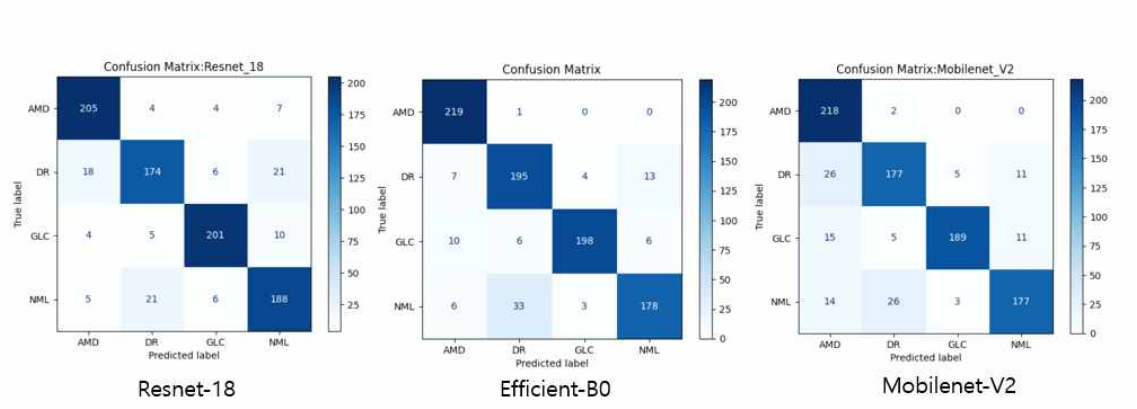
6. t-SNE (Colab)



왼쪽 그림은 클래스당 100장의 이미지를 사용할 경우 명확하게 분리된 클래스 클러스터를 보여주는 반면, 오른쪽 그림은 클래스당 980장의 이미지를 사용할 경우 데이터 다양성이 증가하면서 경계가 더 겹치는 양상을 나타낸다. 이러한 시각화는 데이터 규모에 따라 특징의 응집도와 클래스 간 분리도가 어떻게 변화하

는지를 보여준다.

7. Confusion Matrix (Colab)



각 Confusion Matrix는 네 가지 안구 질환 클래스에 대한 분류 성능을 보여준다. EfficientNet-B0는 정상을 제외한 모든 클래스에 대해 더 강한 구분 능력을 나타낸 반면, MobileNet-V2는 AMD를 제외한 다른 클래스 간의 오분류가 다소 더 높은 것으로 나타났다.

VIII. 프로젝트 수행 결과 분석

1. 재학중 취득한 기초지식의 활용 내용

교과목	활동 내용
자료구조	알고리즘 효율성에 대한 기초 역량
생활속의 빅데이터	데이터 활용에 대한 기초 역량
문제해결을 위한 코딩 첫 걸음	Python을 이용한 기초 알고리즘 설계 능력
객체지향언어	프로그램 작성시 효율적인 방안 학습

2. 재학중 취득한 실험지식의 활용 내용

교과목	활동 내용
자바 프로그래밍	Java을 이용한 Android Studio에서의 코드 작성
디지털영상처리	Deep Learning AI의 기본 개념 이해 및 심화 학습

3. 본 프로젝트 수행과정에서의 설계 능력 향상 내용

본 프로젝트를 수행하면서 단순한 기능 구현을 넘어, 사용성과 목적에 최적화된 시스템 구조와 설계 방법론의 중요성을 체계적으로 이해하고 실전 적용할 수 있게 되었다. 특히 전체 시스템을 하드웨어, 소프트웨어, 데이터 처리로 구분하여 모듈 단위로 기능을 설계하고 연결하는 구조적 사고 능력이 크게 향상되었다.
우선, 딥러닝 모델의 학습 구조를 설계할 때, 단순히 모델을 선택하는 데 그치지 않고, 학습 데이터의 도메인 차이와 추론 환경의 제약 조건까지 고려하여 전처리 방식과 모델 아키텍처를 직접 구성하였다. 이를 통해 경량화 및 모바일 최적화에 필요한 요소들을 명확히 분석하고 설계하는 능력을 기를 수 있었다.
또한, OpenCV를 이용한 이미지 전처리 기능, PyTorch Lite 모델과의 연동 방식, 사용자 인터페이스 흐름 등을 Android 앱의 구조 안에 기능 단위로 나누어

설계함으로써, 사용자 친화성과 실시간성이라는 목표를 달성하기 위한 UI/UX 중심의 기능 설계 능력도 크게 향상되었다.
하드웨어 측면에서는 3D 프린팅을 통한 어댑터 설계 및 20D fundus 렌즈와의 광축 정렬을 고려한 구조 설계를 수행함으로써, 물리적 설계(기구 설계)와 소프트웨어 로직 간의 통합 설계 역량도 경험할 수 있었다. 특히 Galaxy S9의 실제 치수와 카메라 위치에 기반하여 커스터마이징한 3D 모델을 제작하면서, 실사용자를 고려한 제품 설계 사고력을 발전시킬 수 있었다.
종합적으로, 본 프로젝트는 단순 구현이 아닌 문제 해결을 위한 구조적 사고와 기능적 분해, 그리고 시스템 통합 설계 능력 향상에 중점을 두었으며, 결과적으로 실제 사용자 환경에 적용 가능한 실용적 시스템을 설계·구현하는 경험을 쌓을 수 있었다.

4. 본 프로젝트 수행과정에서의 문제 해결 내용

1. 안드로이드 환경에서의 모델 동작 에러

Colab에서 학습된 모델은 .pt 파일로 저장된 상태였으며, 이를 Android 환경에서 사용하기 위해서는 PyTorch Lite 형식(.ptl)으로 변환이 필요했다. 변환 과정에서 torch.jit.trace와 optimize_for_mobile()을 적용하였으나, 초기에는 모델 로딩 시 오류가 발생하거나 추론 결과가 비정상적으로 출력되었다. 이 문제를 해결하기 위해 Android 앱에서 사용하는 이미지 전처리 방식이 Colab 학습 시의 전처리와 일치하는지에 대해 비교 분석하였으며, CLAHE 적용, 중심 크롭, 패딩, Resize, Normalize 순서를 통일시킨 결과, 모델이 스마트폰 환경에서도 정상적인 성능으로 추론되기 시작하였다.

2. Resnet-18의 성능 향상화를 위한 Customized

ResNet-18의 기본 구조로는 소량의 학습 데이터에 대한 과적합 우려가 있었고, 분류 성능에도 한계가 있었다. 이를 개선하기 위해 Fully Connected Layer를 Dropout, BatchNorm, ReLU로 커스터마이징하고, parameter를 직접 하나하나 수정해가면서 일반화 성능을 강화하였고, 이 구조 변경은 실제로 Validation Accuracy 향상에 기여하였다.

3. 학습을 위한 데이터 셋 전처리 및 분류

모델 학습에 사용된 데이터셋 또한 성능에 큰 영향을 미쳤기 때문에, 분석에 적합한 데이터셋을 찾기 위해 여러 공공 의료 데이터베이스를 검토하였다. 그 결과, ODIR-5K 안저 이미지 데이터셋을 확보하였고, 클래스별 균형, 이미지 품질, 질병 레이블의 신뢰성 측면에서 높은 분석 가치를 가진다고 판단되어 최종 데이터셋으로 선정하였다. 이후 이 데이터셋을 기반으로 실험을 반복하여, 성능 분석 및 시각화까지의 전체 파이프라인을 구축하였다.

4. 안드로이드 환경에서의 계산처리속도 최적화

마지막으로, 모바일 앱의 실시간 추론 속도를 개선하기 위해 다양한 최적화를 수행하였다. 고해상도 이미지를 그대로 사용하는 경우 연산 시간이 증가하고 앱이 느려지는 현상이 있어, 이미지 크기 축소 및 압축, 불필요한 로깅 제거, Exif 데이터 제거 후 처리 등의 방식을 적용하였다. 특히 사진을 찍었을 시의 .ptl 모델의 추론 속도는 .pt 대비 약 90% 이상 개선되었으며, 스마트폰에서도 3초 이내에 결과를 출력할 수 있도록 최적화가 완료되었다.

5. 본 프로젝트 수행과정에서의 실무 능력 향상 내용

본 프로젝트는 단순한 기술 구현을 넘어, 실제 서비스 환경을 고려한 전 주기적 시스템 개발 경험을 제공함으로써 다양한 실무 능력을 향상시킬 수 있는 계기가 되었다.
우선, 딥러닝 모델 학습과 최적화 과정에서 실무에 필요한 데이터 전처리 파이프라인 구성, 모델 구조 튜닝, 하이퍼파라미터 조정 등의 과정을 직접 수행하면서, 모델 성능을 분석하고 개선하는 능력을 체계적으로 키울 수 있었다. 특히 PyTorch 프레임워크 기반의 학습 코드를 구성하고, torch.jit.trace 및 optimize_for_mobile을 통해 실서비스용 모델을 경량화하며, 실제 디바이스에서 작동하는 모델로 전환하는 실전 경험을 축적하였다.
또한, Android Studio 환경에서의 앱 개발 및 배포 과정에서는 Java 기반으로 액티비티 구성, 카메라 촬영 처리, 이미지 압축 및 저장, OpenCV 연동, PyTorch Lite 모델 로딩 등 실제 모바일 앱에서 사용되는 주요 기술들을 실습을 통해 익혔으며, build.gradle, AndroidManifest.xml 등 구성 파일의 설정과 권

한 처리 방식에 대한 이해도 함께 심화되었다.
이미지 처리 측면에서는 OpenCV를 활용한 원형 검출, CLAHE 적용, 정규화 처리 등 실시간 전처리 알고리즘을 모바일 환경에 최적화하는 과정을 통해, 모바일과 서버 환경 간 성능 차이를 고려한 알고리즘 구현 역량을 높일 수 있었다. 특히, 학습과 추론 환경의 전처리 일치를 유지하며 모델 오동작을 방지하는 등 실무에서 자주 발생하는 데이터 도메인 차이 문제에 대한 대응 역량을 키웠다.
추가로, UI/UX 설계 능력도 함께 향상되었다. 사용자 중심의 앱 구성과 인터페이스 흐름을 직접 설계하고 구현하면서, 병원 외 일반 사용자나 의료 취약 지역의 사용자를 고려한 직관적인 사용자 경험 설계 능력을 확보하였다.
하드웨어 및 시스템 통합 측면에서도, 3D 프린팅을 활용한 광학 어댑터 제작, Galaxy S9 기종에 맞춘 맞춤 설계, 20D fundus lens와의 정렬 최적화 등 실물 기기를 기반으로 한 하드웨어-소프트웨어 연동 경험을 느낄 수 있었다.

6. 본 프로젝트 수행과정에서의 팀원간 협동 내용

본 프로젝트는 기획부터 개발, 테스트 및 발표까지 전 과정에 걸쳐 팀원 간의 긴밀한 협업을 기반으로 수행되었다. 팀은 총 3명으로 구성되었으며, 각자의 전공 역량과 관심 분야에 따라 역할을 분담하고 상호 피드백을 주고받으며 공동의 목표를 달성해 나갔다.
또한, 프로젝트 중 발생한 문제 상황에 대해 정기적인 온라인/오프라인 회의를 통해 빠르게 원인을 분석하고 대안을 도출하며 팀워크를 강화하였다. 서로의 전문성을 존중하고 지식을 공유하면서, 각자의 작업 영역을 넘어서서 도와주는 분위기를 유지한 덕분에 기한 내에 고품질의 결과물을 도출할 수 있었다.

7. 개발된 결과물에 대한 전시 방법 계획

개발된 스마트폰 기반 온디바이스 안구 질환 진단 시스템은 졸업 작품전 및 캡스톤 전시회에서 사용자가 직접 체험할 수 있는 형태로 전시할 예정이다. 부스 중앙에는 Galaxy S9에 3D 프린팅 어댑터와 20D 렌즈를 장착한 실물 데모 장비를 배치하여, 모형 안구 또는 출력된 안저 이미지를 직접 촬영해볼 수 있도록 한다. 관람객이 촬영 버튼을 누르면 앱이 즉시 이미지를 전처리(CLAHE, 원형 크롭, 정규화)하고 ResNet-18 Lite 모델로 실시간 분류를 수행한 뒤, 4개 질환(AMD, DR, Glaucoma, Normal)에 대한 확률과 함께 Grad-CAM++ 히트맵을 화면에

시각적으로 표시한다. 옆편에는 A1 크기 포스터와 태블릿을 활용해 데이터 파이프라인, t-SNE 및 Confusion Matrix 분석 결과, 추론 속도 비교 차트 등을 설명자료로 전시하며, 사용자가 학습 과정과 성능 지표를 직관적으로 이해할 수 있도록 구성한다.

〈참고 문헌〉

[1] L. Dai et al., "A deep learning system for detecting diabetic retinopathy across the disease spectrum," Nature Communications, vol. 12, p. 3242, 2021. doi: 10.1038/s41467-021-23458-5.

[2] Y. Jeong, Y.-J. Hong, and J.-H. Han, "Review of machine learning applications using retinal fundus images," Diagnostics, vol. 12, no. 1, p. 134, 2022. doi: 10.3390/diagnostics12010134.

[3] A. Neto, J. Camara, and A. Cunha, "Evaluations of deep learning approaches for glaucoma screening using retinal images from mobile device," Sensors, vol. 22, no. 4, p. 1449, 2022. doi: 10.3390/s22041449.

[4] S. B. Patiland B. P. Patil, "Retinal fundus image enhancement using adaptive CLAHE methods,"
SeyboldReport, vol. 15, no. 9, pp. 3476–3484, 2020.

[5] S. Anitha and S. Priyanka, "Smart phone based automated diabetic retinopathy detection system," Measurement: Sensors, vol. 31, p. 100957, 2024.

[6] 김성민 외,"데이터 효율적 이미지 분류를 통한 안질환 진단,"
Journalof Internet Computingand Services, vol. 25, no. 3, pp. 19–25, 2024. doi: 10.7472/jksii.2024.25.3.19

[7] P. Jiang, Q. Dou, and L. Shi, "Ophthalmologist-level classification of fundus disease with deep neural networks," Transl.Vis. Sci. Technol., vol.9, no2, p.39,2020 doi: 10.1167/tvst.9.2.39.

[8] A. Hu and K. F. Damji, "New open source 3-dimensional printed smartphone fundus imaging adaptor," Can. J. Ophthalmol., vol. 54, no. 3, pp. 399–400, 2019, doi: 10.1016/j.jcjo.2018.10.017.

<참고 도서>

천인국,「안드로이드 프로그래밍 개정 7판」, 생능출판사, 2024

엘리 스티븐슨 외,「파이토치 딥러닝 마스터」, 책만, 2020

〈종합설계프로젝트 수행 후기〉

학번 **김주영**

작년 5월 말, 졸업 프로젝트 주제를 고민하던 중 우연히 딥러닝을 활용하여 피부암을 진단하는 프로젝트를 진행한 선배님들의 결과물을 보게 되었습니다. 당시 저는 딥러닝에 대한 구체적인 이해가 부족했지만, 막연하게나마 “나도 저런 프로젝트를 한번 해보고 싶다”는 열망이 생겼습니다. 과연 내가 할 수 있을까 하는 두려움도 있었지만, 동시에 도전해보고 싶다는 마음이 컸습니다. 이후 교수님과의 상담을 통해 방향을 잡아나갔고, 결국 9월부터 본격적으로 딥러닝 기반의 프로젝트를 수행하기로 결심하였습니다.

프로젝트를 시작할 당시에는 “데이터를 잘 수집하고, 코드가 잘 돌아가면 원하는 결과가 자연스럽게 나올 것”이라고 생각했지만, 실제로 딥러닝과 안드로이드 개발을 병행하면서 예상보다 훨씬 복잡하고 어려운 문제들이 끊임없이 발생했습니다. 저는 안드로이드 관련 과목을 수강한 적이 없었기 때문에, 개발에 필요한 지식들을 기초부터 전부 독학으로 익혀야 했습니다. 화면 구성, 이벤트 처리, 외부 라이브러리 연동, PyTorch 모델 적용 등 모든 단계가 처음 접하는 내용이었고, 공식 문서나 해외 포럼, 오픈소스 예제를 하나하나 읽고 이해하면서 개발을 진행해야 했습니다. 특히, 딥러닝 모델을 안드로이드 환경에 이식하고 실제 스마트폰에서 동작시키는 과정은 반복적인 시행착오의 연속이었습니다. 코드가 잘 작동하지 않거나 결과가 예상과 다르게 나왔을 때는 좌절감도 컸으며, ‘지금이라도 방향을 바꿔야 하는 것이 아닐까’ 하는 고민도 수없이 했습니다.

그럼에도 불구하고 제가 여기까지 올 수 있었던 이유는 “이 문제만 해결하면 분명히 동작할 것이다”, “이번 시도에서 성공할 수 있을 것이다”라는 희망을 놓지 않았기 때문입니다. 코드가 동작하지 않아도 포기하지 않고, 가능한 모든 방법을 시도해보며 문제의 원인을 찾고자 노력했습니다. 때로는 실패의 연속으로 인해 지치기도 했지만, 그래도 다시 마음을 다잡고 한 줄씩 코드를 고쳐나가면서 결국 원하는 결과에 가까워질 수 있었습니다.

특히 어려움이 극심했던 시점은 안드로이드 환경에서 이미지 입력값을 모델에 전달하는 부분이었습니다. 학습 환경에서는 정확하게 분류되던 이미지가, 실제 앱에서는 제대로 결과가 나오지 않아 많은 혼란을 겪었습니다. 여러 번 전처리 과정을 점검하고, 도메인 차이를 극복하기 위해 추가 데이터를 수집하여 모델을 재학습하는 등 수많은 시도를 반복하며 해결의 실마리를 찾아야 했습니다. 그러한 과정에서 체력적으로나 정신적으로 힘들었지만, 그럼에도 불구하고 끝까지 포기하지 않고 해결책을 찾을 수 있었습니다.

프로젝트의 후반부에는 팀원들과 함께 몇 주간 밤을 새우며 구현을 마무리하였고, 팀의 리더로서 팀원들이 지치지 않도록 격려하고, 방향을 잡아주기 위해 노력했습니다. 사실은 저 또한 해결되지 않을까봐 불안하고 두려운 마음이 있었지만, 겉으로는 최대한 침착하게 행동하려 애썼습니다. 다행히도 팀원들 모두가 끝까지 책임감 있게 참여해주었고, 서로를 믿고 의지하는 분위기 속에서 프로젝트를 무사히 완수할 수 있었습니다.

무엇보다 매주 바쁘신 와중에도 지속적으로 조언과 피드백을 주신 김준민 교수님께 진심으로 감사드립니다. 교수님의 방향 제시와 격려가 없었다면 프로젝트를 이만큼 끌고 오기는 어려웠을 것입니다. 또한, 힘든 순간에도 끝까지 저를 믿고 함께 해준 팀원들에게 깊은 감사를 전하

고 싶습니다. 제가 무리한 요청을 드렸을 때도 반발하지 않고 묵묵히 따라와 준 팀원들 덕분에 저도 끝까지 포기하지 않고 이 프로젝트를 마무리할 수 있었다고 생각합니다.
마지막으로, 프로젝트를 준비하던 작년 이맘때쯤, 제가 선배님들의 결과물을 보며 큰 영감을 받았던 것처럼, 이번 저희 팀의 프로젝트도 후배님들에게 하나의 참고가 되고, 나아가 도움이 될 수 있기를 진심으로 바랍니다. 직접 문제를 해결해가며 쌓은 경험과 지식들이 단순히 저희 팀에만 머무는 것이 아니라, 후배님들이 졸업 작품이나 관련된 주제를 고민할 때 실질적인 방향을 잡는 데에도 작은 이정표가 되었으면 좋겠습니다. 후배님들이 '나도 해볼 수 있겠다'는 용기를 얻는 계기가 된다면, 그 자체로도 이 프로젝트의 의미는 더욱 커질 것이라 생각합니다.

학번 **박찬기**

저희 조는 이번 프로젝트의 주된 기술로 인공지능과 머신러닝, 그 중 딥러닝 분야를 선택했습니다. 저는 이전에 디지털 영상처리 과목을 선행학습 했음에도 불구하고, 실제로 사용이 가능한 수준의 프로젝트 결과물을 만들기에는 부족하다고 판단하여, 따로 정보를 찾아보며 추가적으로 학습을 했습니다.
처음에는 무엇을 어떻게 만들어야 하는지 이해하지 못해 방황했던 시간도 있었습니다. 초반에 딥러닝 훈련을 시킨 모델과 안드로이드 스튜디오에서의 사용하는 언어가 달라 이를 서로 호환시키기 위해 라이브러리에서 네이티브 파일을 따로 받아와야 했는데, 이 부분이 쉽게 해결되지 않아 며칠 동안 프로젝트에 진전이 없던 시기에는 포기하고 싶다는 생각도 들었습니다.
그러나 팀원들과 같이 상의하며 협동하는 과정을 통해 이와 같은 문제가 해결되고 프로젝트가 한 단계씩 나아가는 모습을 보일 때는 그에 따른 성취감도 느낄 수 있었습니다. 제가 혼자서 작업을 할 때는 찾아내지 못했던 부분도 팀원끼리 상의하며 도움을 받아 해결한 부분도 많았고, 반대로 제가 팀원이 찾아내지 못했던 부분을 찾아내 직접 도움도 주는 이른바 상호협동 관계가 가장 빛났던 팀이라고 생각합니다.
프로젝트가 진행되며 자연스럽게 몰랐던 부분도 직접 찾아보며 학습하고, 무엇보다 어떤 문제가 발생했을 시 이를 어떻게 대처하고 해결할지를 추론하는 문제해결 능력을 키우는데 있어, 가장 큰 도움이 되었습니다.
또한, 이번 졸업 프로젝트는 나중에 겪게 될 엔지니어로서의 실무환경에 대해서도 선행학습을 해보는 기회가 되었다고 생각합니다. 여러 명의 인원이 모여 생기는 장점은 분명 혼자 작업을 수행하는 것보다 큰 도움이 되지만, 그에 따른 단점도 분명 존재합니다. 팀원들 간에 있어 서로의 의견이 맞지 않아 충돌할 수도 있고, 서로 의사소통에 있어 전하려는 뜻이 정확히 전달되지 않는 어려움을 겪을 수도 있습니다. 저희 팀도 이러한 크고 작은 마찰을 프로젝트 수행 중 필연적으로 겪었고, 오히려 이를 극복하고 더 성장해 나갈 발판으로 삼아 최종적으로 가장 팀워크가 빛났던 팀으로 성장할 수 있었다고 생각합니다.
또한, 바쁜 일정에도 불구하고 최대한 저희의 상담 요구에 응해주시고, 문제가 해결되지 않을 때 적절한 조언으로 도움을 주신 김준민 교수님의 공헌도 매우 컸다고 생각합니다. 교수님의 조언과 팀원들의 협동으로 앞으로 나아갈 길이 정해지고, 모두가 한마음으로 꾸준히 달려왔기에 이와 같이 프로젝트를 성공적으로 마무리 할 수 있었다고 생각합니다.

마지막으로 포기하지 않고 같이 달려온 저희 팀원과 김준민 교수님께 감사의 말을 전하며 이상 종합설계프로젝트 수행 후기를 마치겠습니다.

학번 이재강

온 디바이스 방식으로 안구질환을 분석하는 ai를 개발하는 과정에서 안드로이드 UI/UX 제작을 맡았습니다. 이 프로젝트는 안드로이드 디자인과 앱 기능을 구현하고 딥러닝 모델 파일을 앱에 로드하기만 하면 작동이 되는 줄만 알았던 저에게 프로젝트 시작과 동시에 안드로이드 개발이 결코 생각처럼 쉬운 일이 아니고 실제 개발을 하기 전까지는 간단해 보이는 모든 것들이 실제로는 프로젝트를 하며 수행하는 수많은 작업들이 서로 영향을 미치며 겉잡을 수 없이 오류를 만들어 낸다는 것을 깨달았습니다.
앱 실행시 카메라로 사진을 찍으면 모델이 사진을 분석해서 질병 분류를 해야 하지만 카메라를 사용하면 바로 앱이 종료되는 문제가 생겼습니다. 코드상에는 아무런 문법 오류가 나타나지 않았지만 문제가 쉽게 해결되지 않아서 오랫동안 찾아보다 팀원분들께 도움을 요청했고, 몇 주간의 시행착오를 겪으며 생각지도 못했던 부분에서 문제점을 찾아서 해결했던 기억이 있습니다. 이후에도 저는 개발 경험 부족에 의해 안드로이드에 딥러닝 모델을 로드하고 앱이 제대로 실행되려면 단순히 로드만 하면 된다고 생각했고, 왜 분류가 제대로 안되는지에 대해서 어떻게 대처를 해야 하는지 파악하는데 어려움을 겪었습니다.
이번에도 팀원분들의 도움을 받아 해결했고, 이번에는 이전 오류가 해결됐을 때와 달리 제가 작성한 자바 코드 메서드를 하나씩 분석해 보는 것을 넘어서 제가 맡은 부분만 아는 것이 아니라 제가 맡지 않은 부분에 대해서도 마치 제가 직접 한 것처럼 아주 자세하고 정확하게 알고 있어야 한다는 것을 깨달았습니다.
또한 작성한 코드의 극히 일부만 고쳐야 한다는 편견을 깨고, 문제 발생 시 작성한 모든 코드들 심지어 당연히 올바르게 작업했다고 확신하는 부분까지도 잘못되었을 수 있다고 생각하고 문제에 접근해야 한다는 것도 깨달았습니다.
프로젝트를 하며 팀원분들의 도움을 많이 받았고 제가 맡은 부분만 잘 알면 된다는 기존의 생각이 문제 해결 능력을 떨어트려서 문제 해결을 하기 힘들다는 것을 미처 생각하지 못했습니다. 그럼에도 많은 도움을 주신 팀원분들과 프로젝트를 하며 생긴 문제를 직접적인 해결보다는 문제 해결의 방향성을 제시해 주셔서 문제 발생 시 팀과 협력하여 어떻게 접근을 해야 하는지 알게 해주신 교수님께 감사의 마음을 전합니다.