Ai Skin Doctor

Ai 시스템 반도체 설계 (2기)
엄찬하, 김민규, 임재홍, 신상학

Index

OverView

1. 배경 및 목표
2. 특징 및 장점
3. 기대 효과
4. 구성 요소

일정 / Design

1. 개발 일정
2. 개발 과정
3. Design Flow

Technology

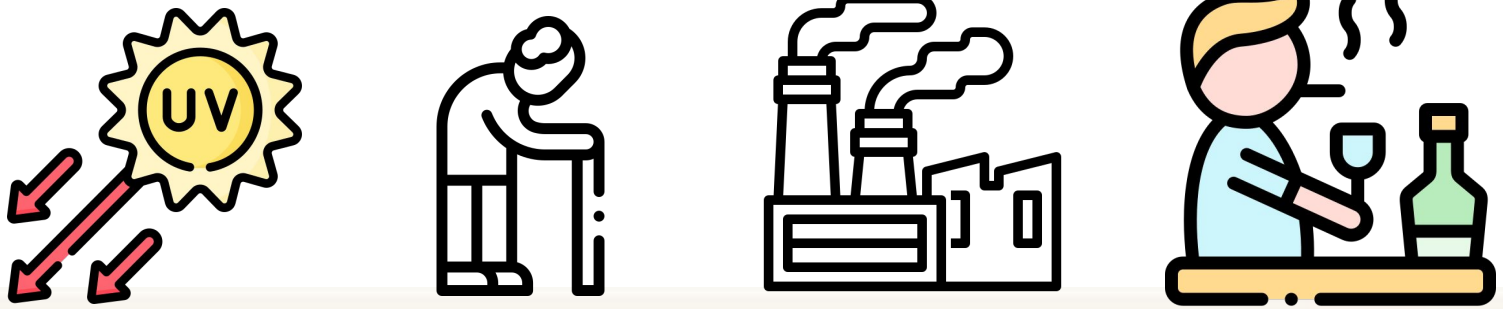
1. DataSet
2. 모델 학습
3. Performance
4. 핵심 기술

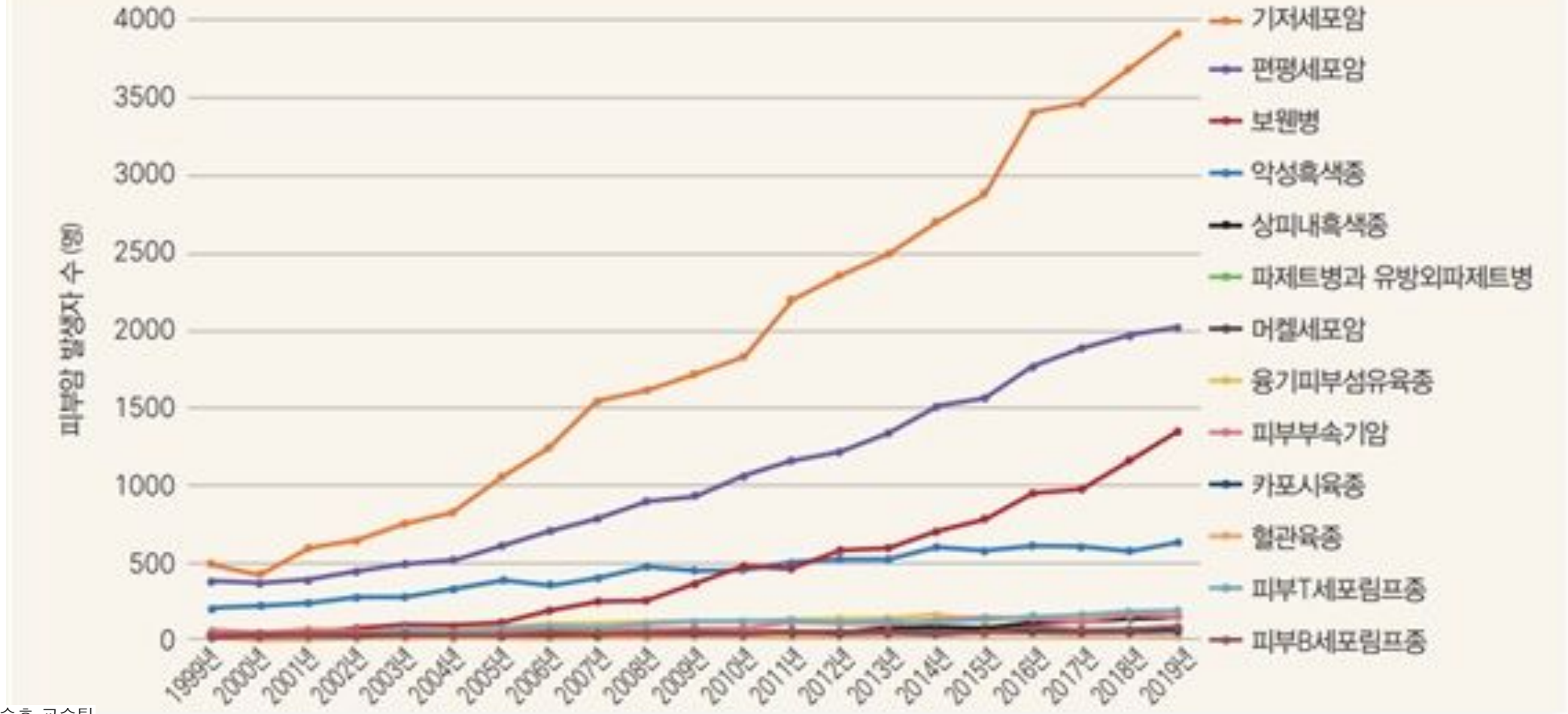
고찰 / 마무리

1. Trouble shooting
2. 고찰
3. 시연 영상


기저세포암

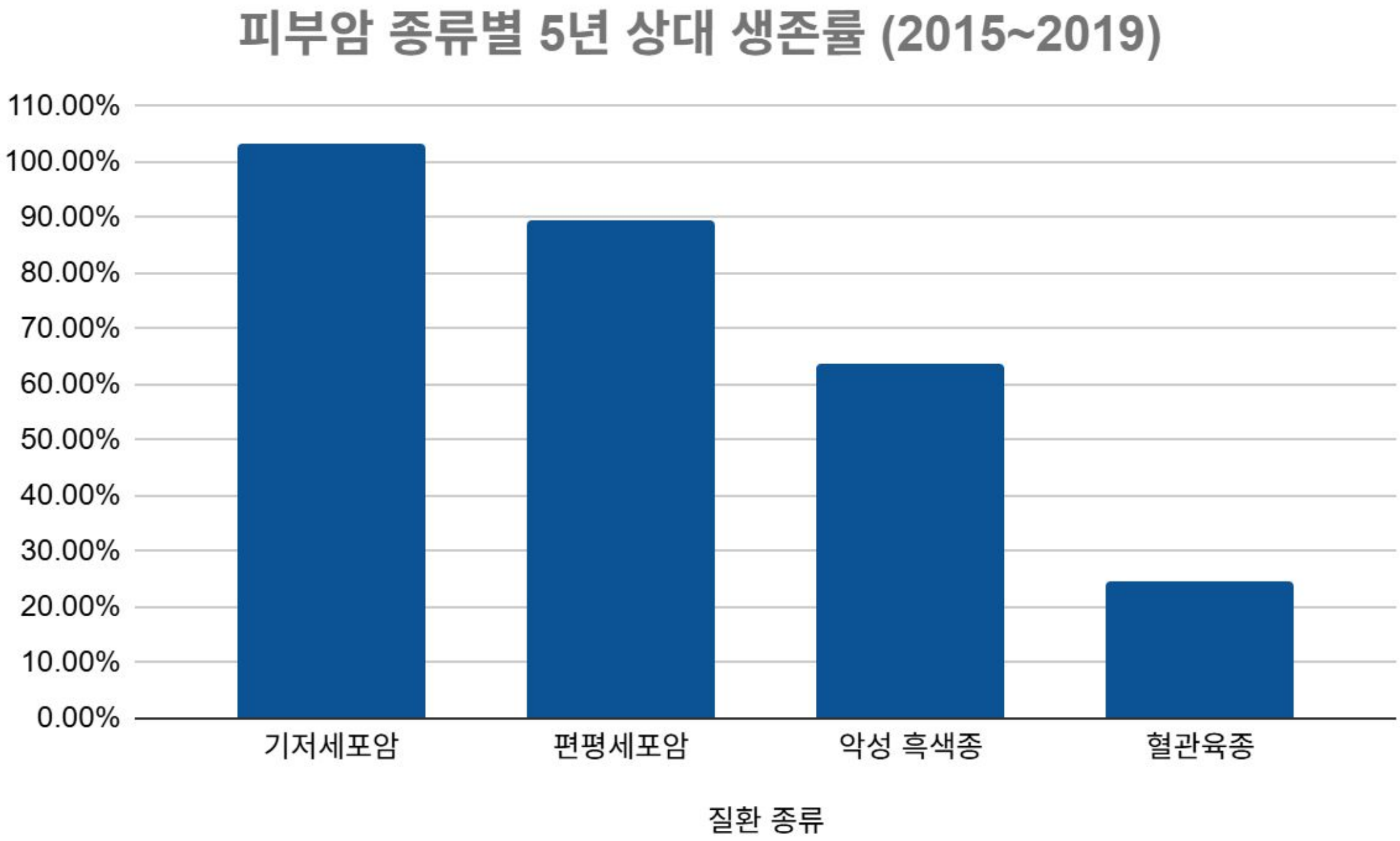
배경





중앙암등록본부 권순효 교수팀

출처 : 헬스비즈(http://www.healthbiz.co.kr)

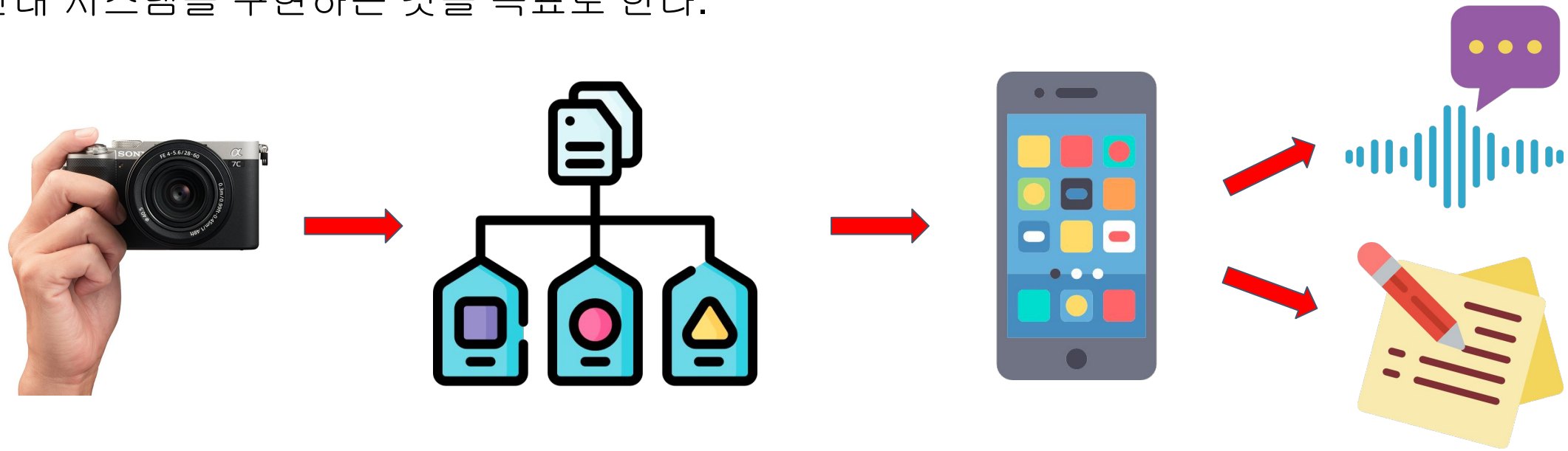

피부암 종류별 5년 상대 생존률 (2015~2019)
110.00%
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
기저세포암
편평세포암
악성 흑색종
혈관육종
질환 종류

개요

AI 기반 피부 질환 인식 & 안내 시스템

카메라를 통해 실시간으로 피부 상태를 인식, 이를 기반으로 피부 질환의 종류를 분류

해당 질환에 대한 정보를 사용자에게 텍스트 및 음성을 제공하는 AI 기반 피부 질환 인식 및 안내 시스템을 구현하는 것을 목표로 한다.



목표

1. 피부 질환 인식 AI 모델

- AIHub 피부별 종양 DataSet
- MNv2 적용

2. 신뢰도 기반 확정 로직

- 5초 이상 동일 class 감지 시 확정
- 오진 가능성 최소화

3. 질환 정보 자동 생성

- LLM (Gemma 3:1b) 기반 설명 텍스트 생성
- 증상, 관리법 등 자연어로 안내

4. 음성 및 App 등의 편의 기능 확장

- 생성된 텍스트 → TTS 음성 변환
- 사용자 접근성 향상을 위한 APP 및 Flutter 형식 제작

주요 특징 및 강점

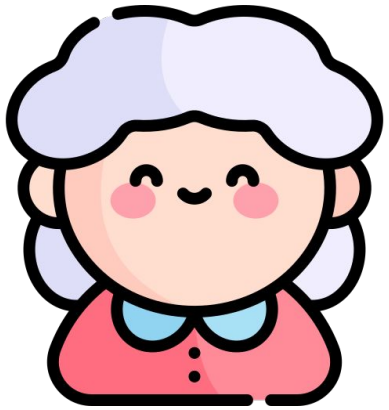
사용자 친화적 **UI/UX**
(텍스트 + 음성 안내)

비대면 진료
홈 헬스케어

확장성 높은 구조
(질환 추가, 다국어 등)

기대 효과

자가 진단
보조 도구



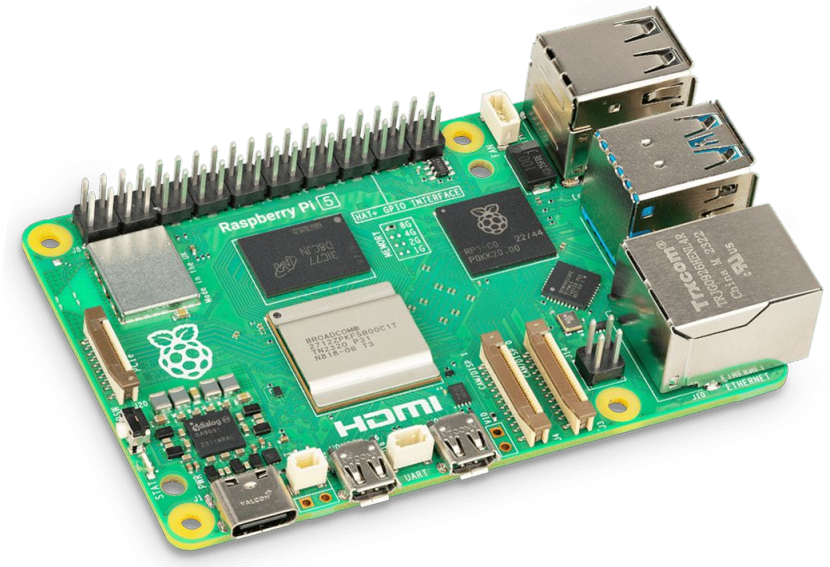
취약계층
음성
안내

디지털
헬스케어
시장
경쟁력

서비스
확장성

사용 HW

- 인터페이스 : **USB**
- 화소 : **200만 화소**
- 해상도 : **1920 x 1080 (FHD)**
- 프레임 : **30fps**
- 시야각 : **60°**



프로세서 : 브로드컴 BCM2712 쿼드 코어 64비트 Arm Cortex-A76 CPU, 2.4GHz
메모리 : 4GB LPDDR4X-4267 SDRAM
그래픽 : VideoCore VII GPU, OpenGL ES 3.1, Vulkan 1.2 지원
디스플레이 : 듀얼 4Kp60 HDMI 디스플레이 출력, HDR 지원
네트워크 : 듀얼 밴드 802.11ac Wi-Fi, 블루투스 5.0/BLE, 기가비트 이더넷 (PoE+ 지원)
전원 : 5V/5A DC 전원 (USB-C PD 지원)

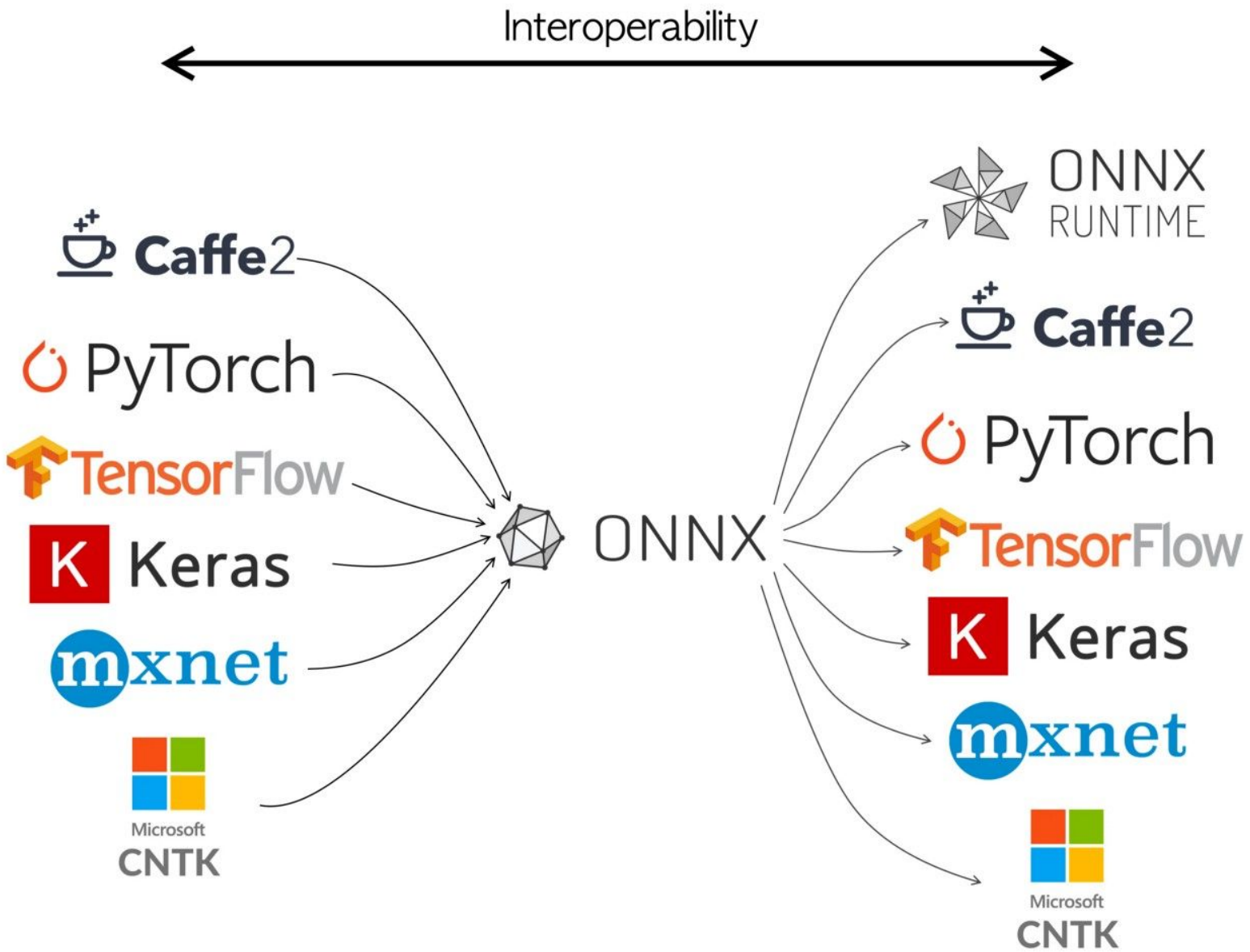
사용 플랫폼

FrameWork

Model Format

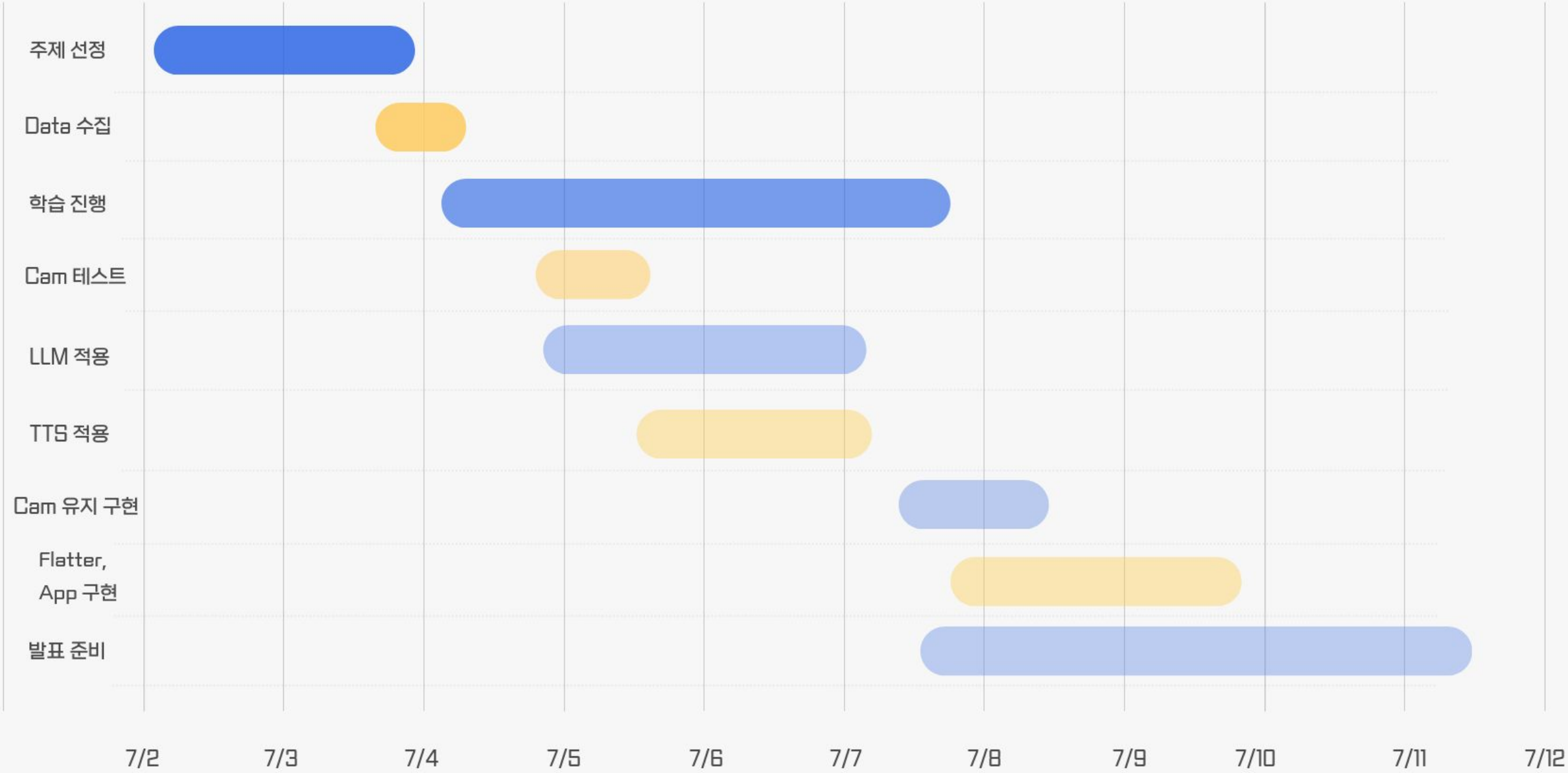
개발 도구

개발 언어

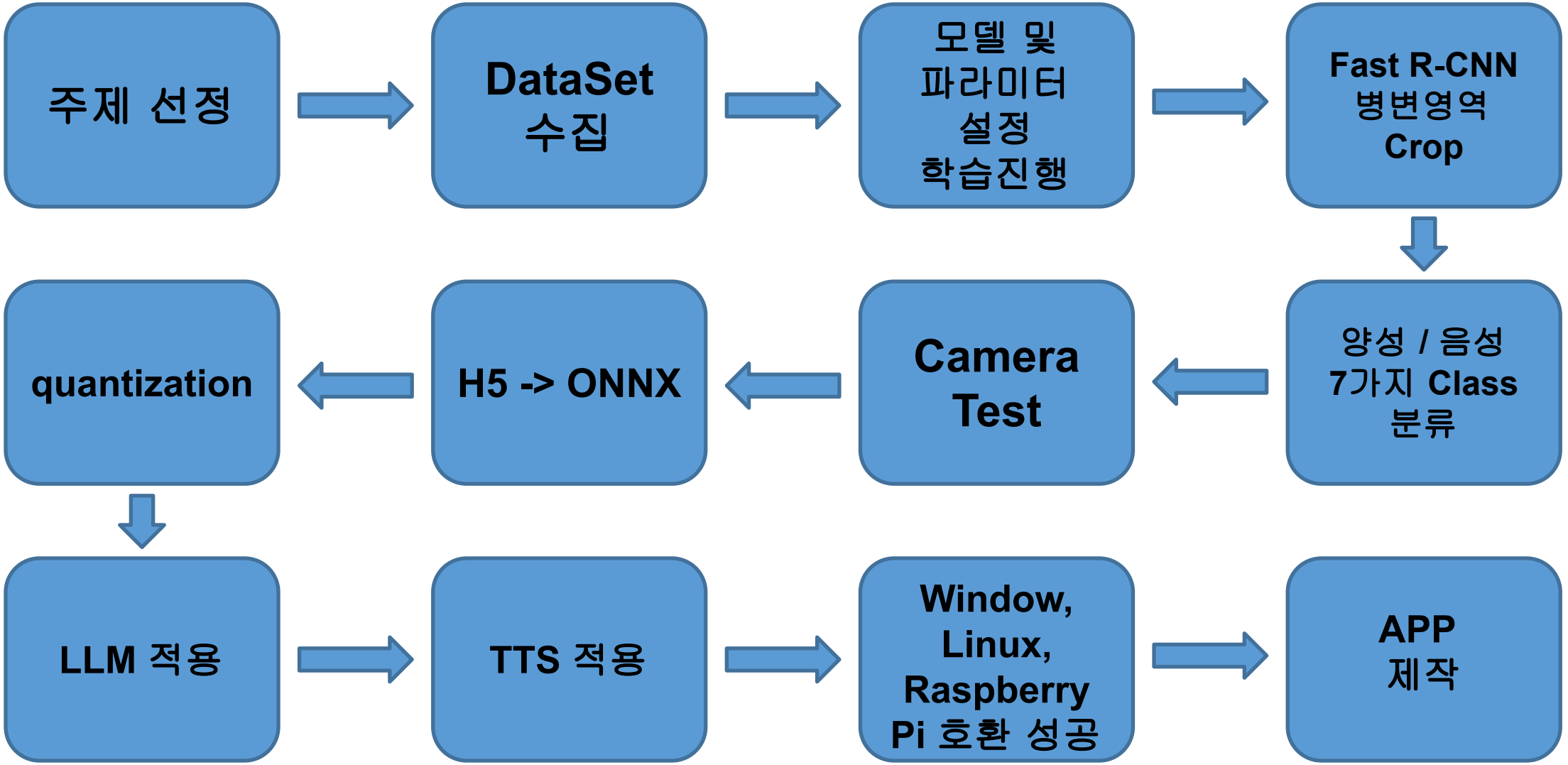

Interoperability
Caffe2
PyTorch
TensorFlow
Keras
mxnet
Microsoft
CNTK
ONNX
ONNX
RUNTIME
Caffe2
PyTorch
TensorFlow
Keras
mxnet
Microsoft
CNTK
export to onnx
load from onnx

이꾸조 Gantt Chart

<개발 일정>

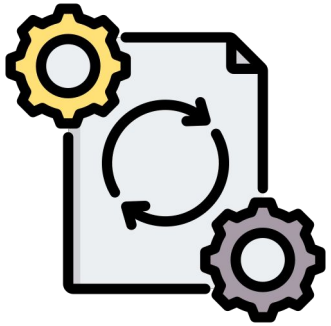


개발 과정

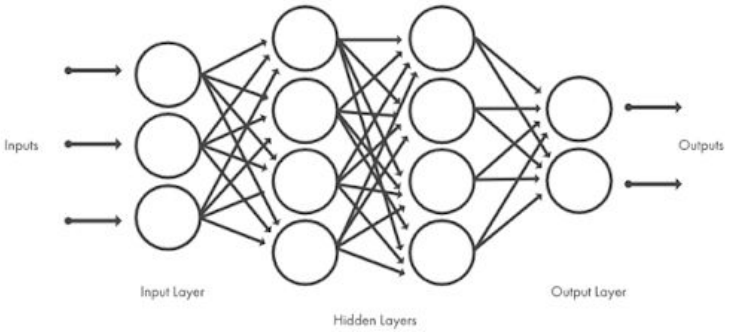

주제 선정
DataSet 수집
모델 및 파라미터 설정 학습진행
Fast R-CNN 병변영역 Crop
양성 / 음성 7가지 Class 분류
Camera Test
H5 -> ONNX
quantization
LLM 적용
TTS 적용
Window, Linux, Raspberry Pi 호환 성공
APP 제작

전체 design flow

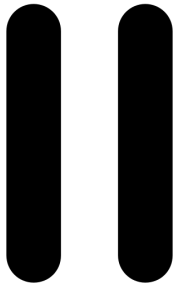
Camera Input



Preprocessing

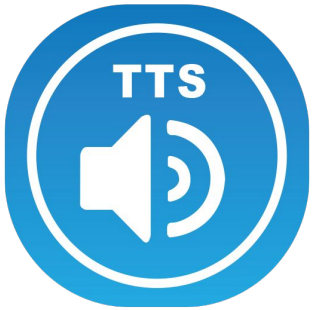


CNN Inference



Class Decision Logic

LLM 호출



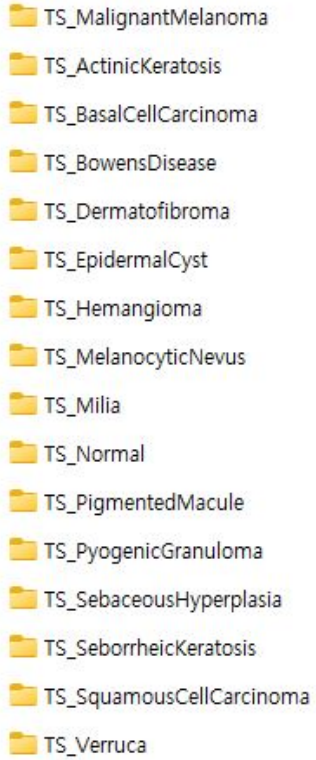
TTS

모델 요약표

모델	MobileNetV2 (Pretrained, include_top=False)
입력 크기	96x96x3
커스텀 레이어	GAP → Dense(128, ReLU) → Dropout(0.5) → Dense(n, Softmax)
Optimizer	Adam (lr=0.0001)
출력	softmax (다중 클래스 분류, 클래스 수=7)
기술 요소	전이학습, 데이터 증강, EarlyStopping, Checkpoint, Dropout
성능	Train acc: 98.4%, Val acc: 98.5% (Epoch 34 기준)

초기 모델 학습 (Class : 15)

<Dataset>

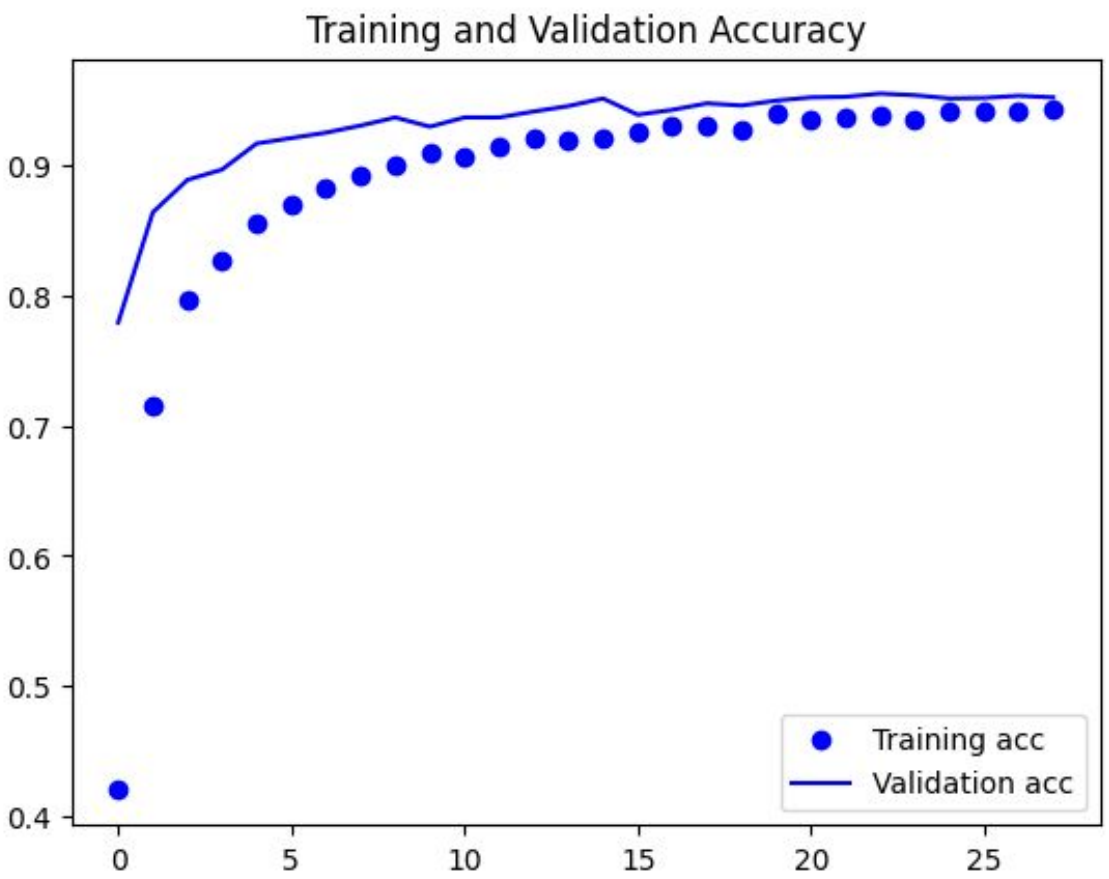


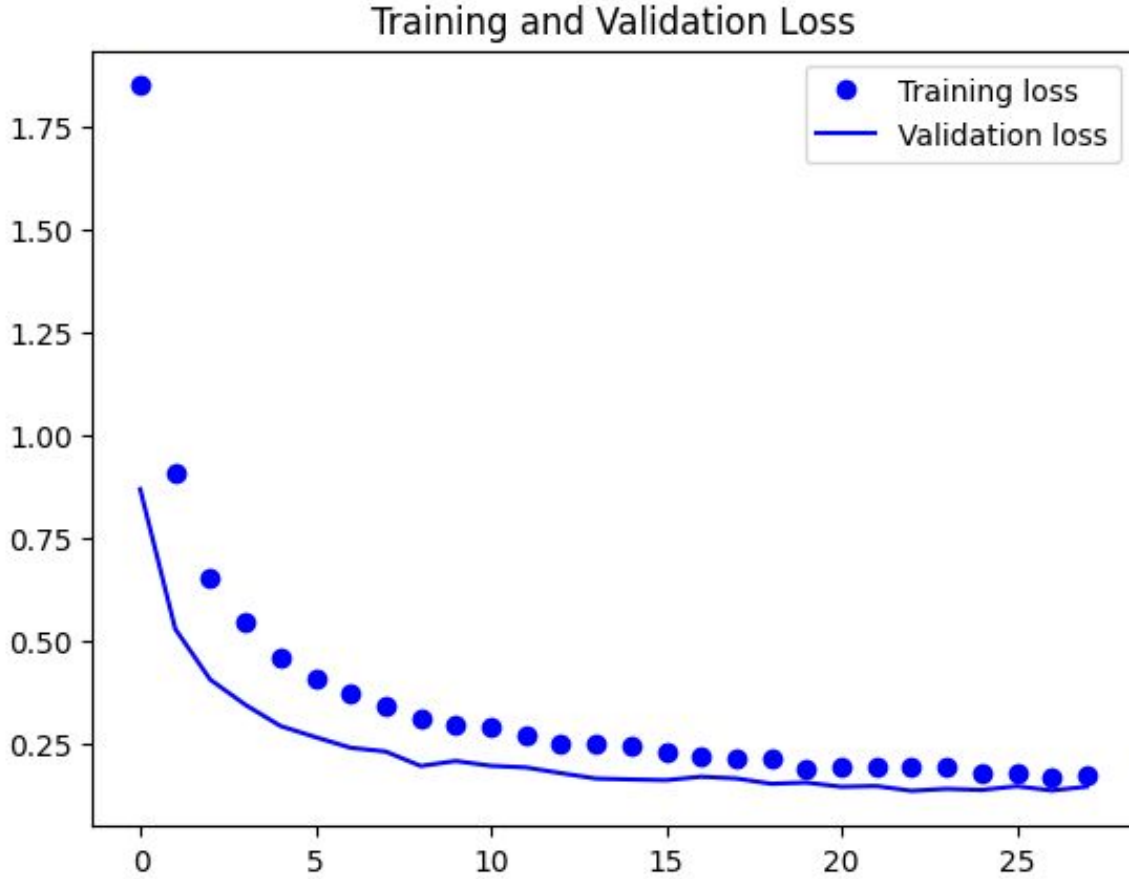

```python
# ✅ 클래스 이름 명시 (15개 클래스)
class_names = [
    'TS_ActinicKeratosis', 'TS_BasalCellCarcinoma', 'TS_BowensDisease', 'TS_Dermatofibroma',
    'TS_EpidermalCyst', 'TS_Hemangioma', 'TS_MalignantMelanoma', 'TS_MelanocyticNevus',
    'TS_Milia', 'TS_PigmentedMacule', 'TS_PyogenicGranuloma', 'TS_SebaceousHyperplasia',
    'TS_SeborrheicKeratosis', 'TS_SquamousCellCarcinoma', 'TS_Verruca'
]

# ✅ 데이터 증강 설정
datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=90,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)

# ✅ 학습 데이터
train_generator = datagen.flow_from_directory(
    dataset_path,
    target_size=(96, 96),
    batch_size=32,
    class_mode='categorical',
    subset='training',
    shuffle=True
)
```

```python
# ✅ 검증 데이터
val_generator = datagen.flow_from_directory(
    dataset_path,
    target_size=(96, 96),
    batch_size=32,
    class_mode='categorical',
    subset='validation',
    shuffle=True
)

# ✅ 전이학습 모델 구성
base_model = MobileNetV2(input_shape=(96, 96, 3), include_top=False, weights='imagenet')
base_model.trainable = False  # Freeze pretrained weights

model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.4),
    layers.Dense(len(class_names), activation='softmax')  # 🔁 클래스 수 자동 반영 (15개)
])

# ✅ 컴파일
model.compile(optimizer=Adam(learning_rate=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

초기 모델 결과 (Class : 15)

```
Epoch 28/30
300/300 - 139s - 465ms/step - accuracy: 0.9425 - loss: 0.1742 - val_accuracy: 0.9521 - val_loss: 0.1469
```





최종 모델 학습 (Class : 7)

<Dataset>

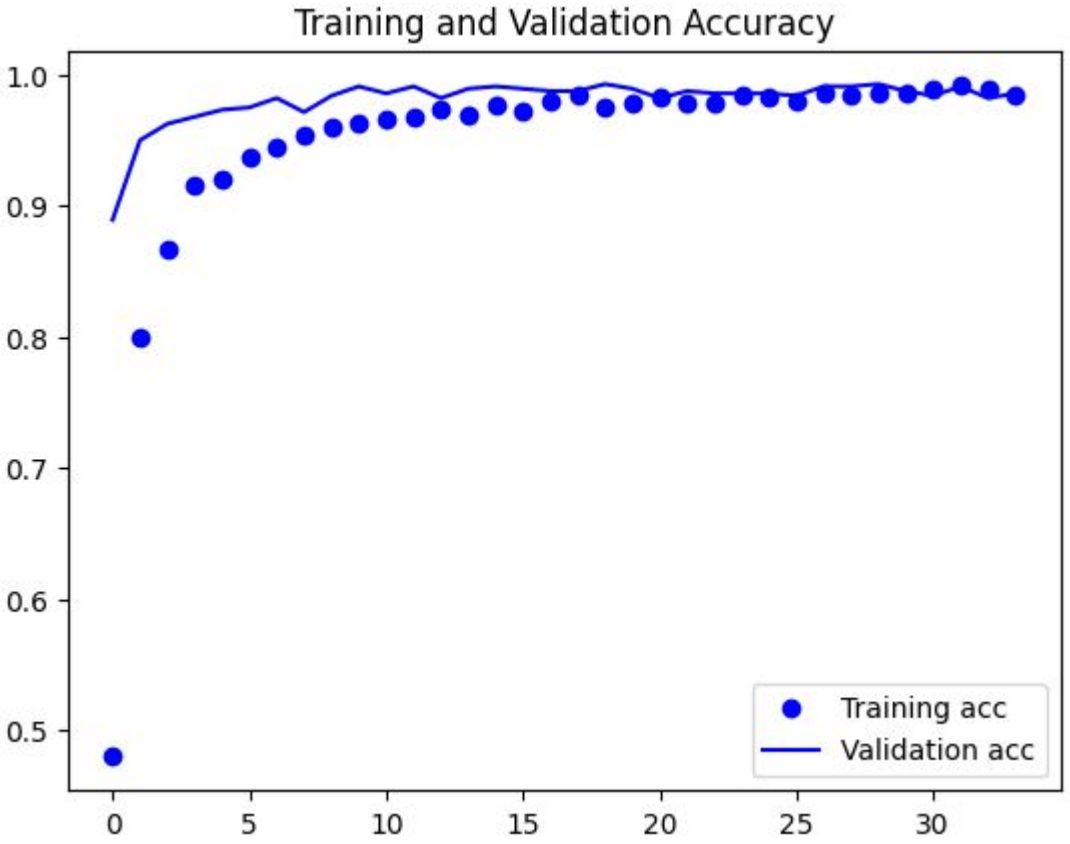
- BasalCellCarcinoma
- EpidermalCyst
- Hemangioma
- Milia
- Normal
- SquamousCellCarcinoma
- Verruca

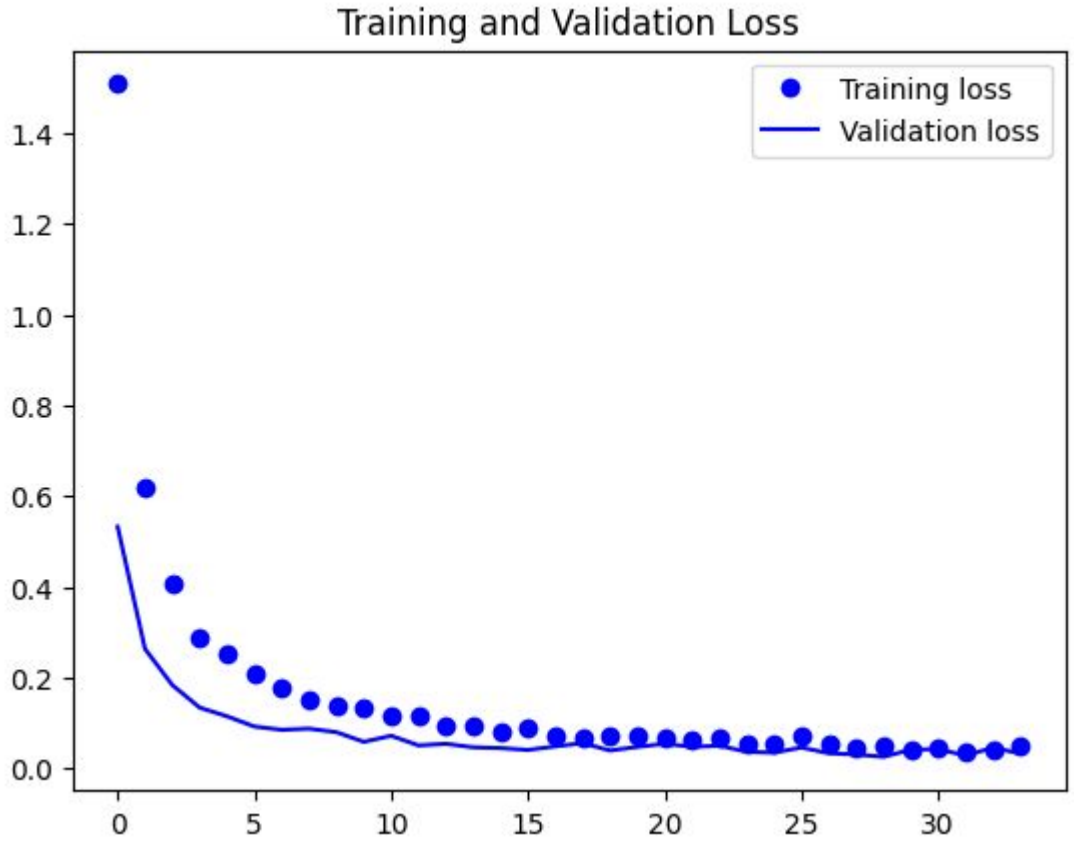
```python
# ✅ 데이터 증강 설정
datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=15,
    width_shift_range=0.05,
    height_shift_range=0.05,
    shear_range=0.1,
    zoom_range=0.1,
    brightness_range=[0.8, 1.2],
    horizontal_flip=True,
    fill_mode='nearest'
)

# ✅ 데이터 로딩
train_generator = datagen.flow_from_directory(
    dataset_path,
    target_size=(96, 96),
    batch_size=32,
    class_mode='categorical',
    subset='training',
    shuffle=True
)

val_generator = datagen.flow_from_directory(
    dataset_path,
    target_size=(96, 96),
    batch_size=32,
    class_mode='categorical',
    subset='validation',
    shuffle=True
)
```

```python
# ✅ 클래스 이름 자동 추출
class_names = list(train_generator.class_indices.keys())
print("클래스 인덱스:", train_generator.class_indices)

# ✅ MobileNetV2 기반 모델 구성
base_model = MobileNetV2(input_shape=(96, 96, 3), include_top=False, weights='imagenet')
base_model.trainable = False

model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(len(class_names), activation='softmax')
])

model.compile(optimizer=Adam(learning_rate=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

최종 모델 결과 (Class : 7)

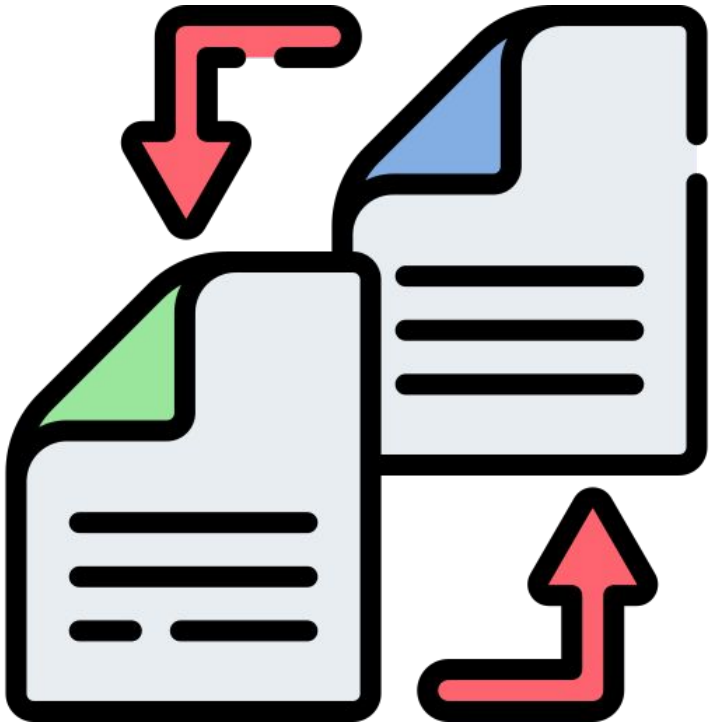
```
Epoch 34/50
70/70 - 136s - 2s/step - accuracy: 0.9844 - loss: 0.0493 - val_accuracy: 0.9857 - val_loss: 0.0324
```

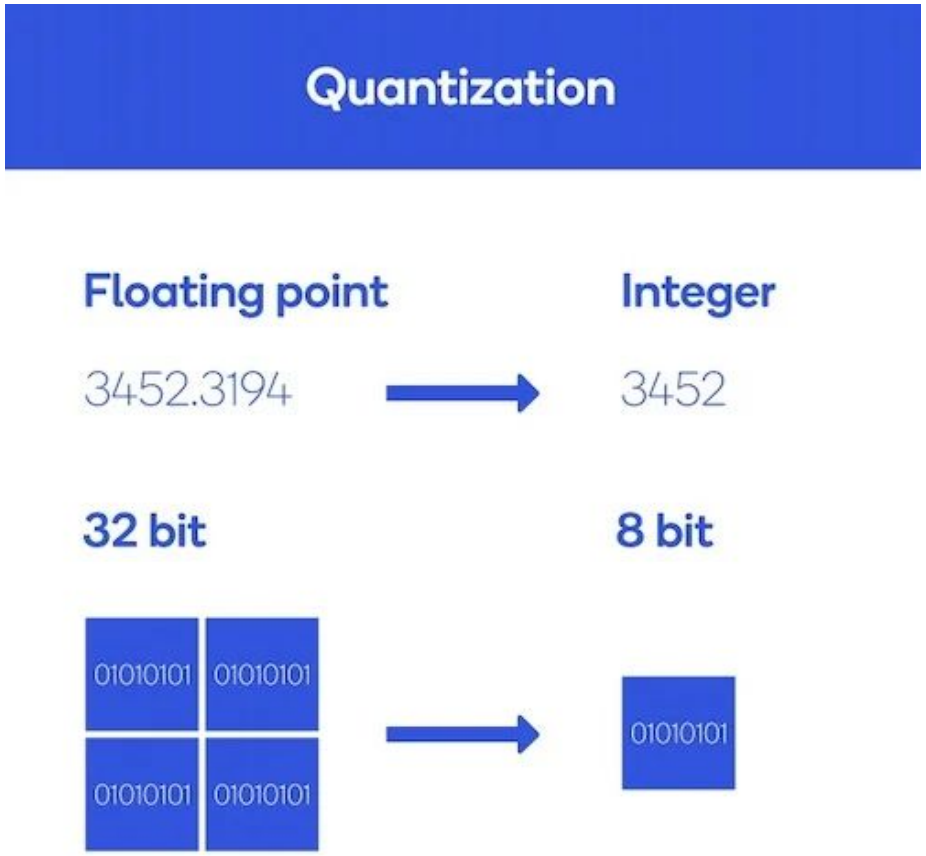




핵심 기술 1. 멀티 플랫폼 호환성

<Converter>





핵심 기술 2. 악성 피부 병변 탐지 `

📸 HAM10000 (ISIC challange 2018 mask.ver)

↓

🎯 Fast R-CNN 구조 (ResNet-50 백본)

↓

✂️ 병변 영역 자동 크롭 (ISIC challange 2023)

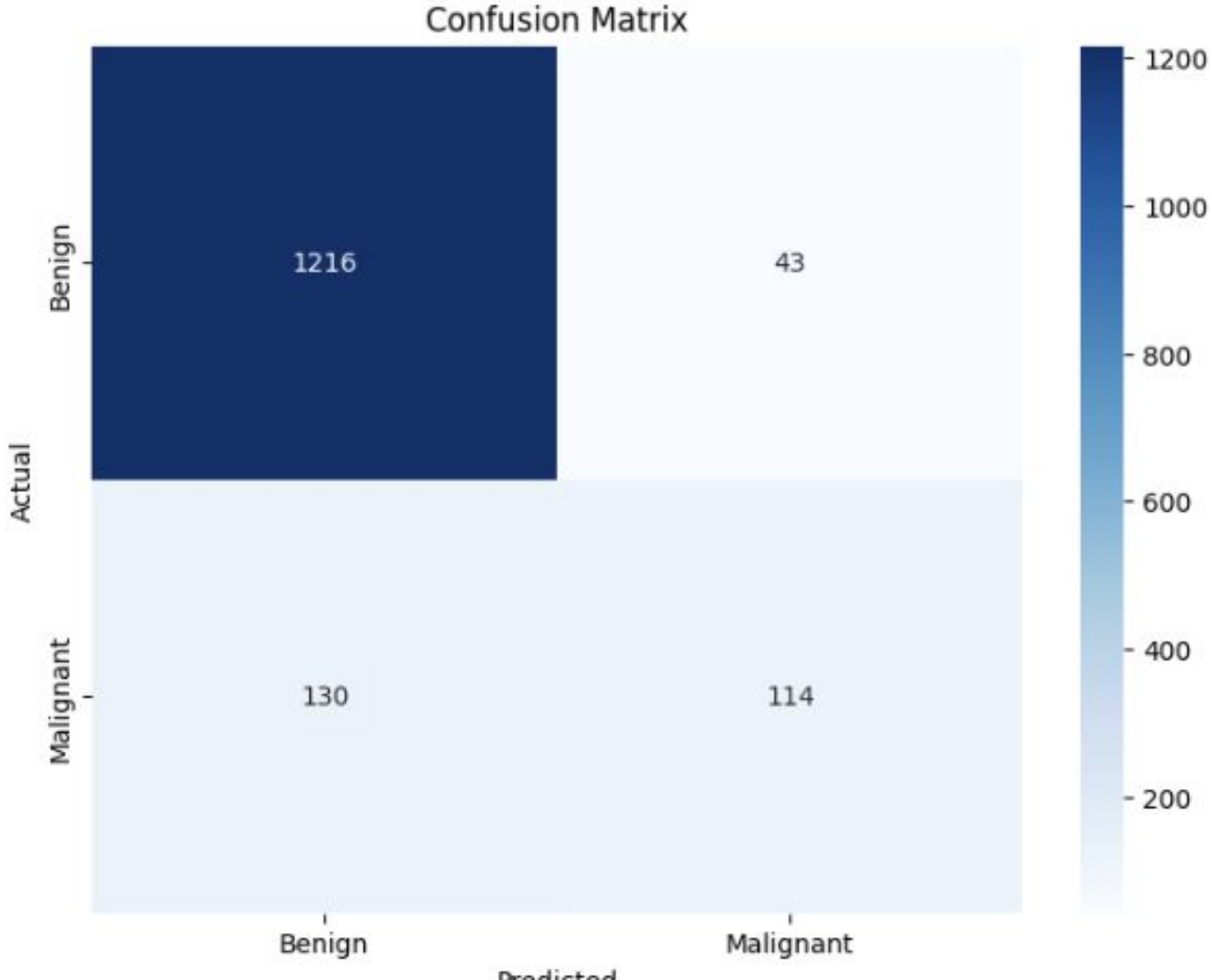
↓

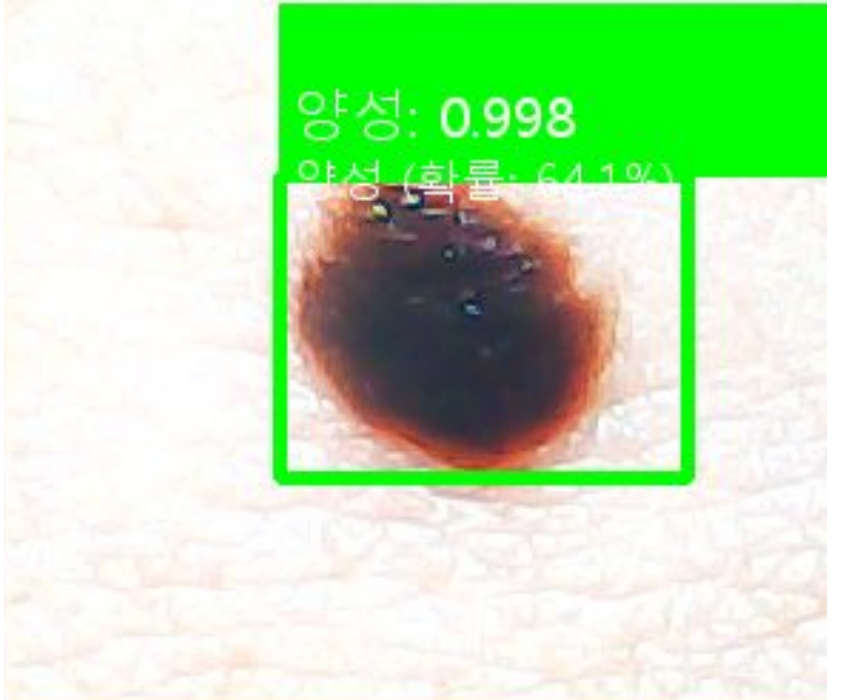
🧠 EfficientNet-B3 이진 분류 (악성/양성)

↓

📊 완전한 성능 평가 (정확도, F1)

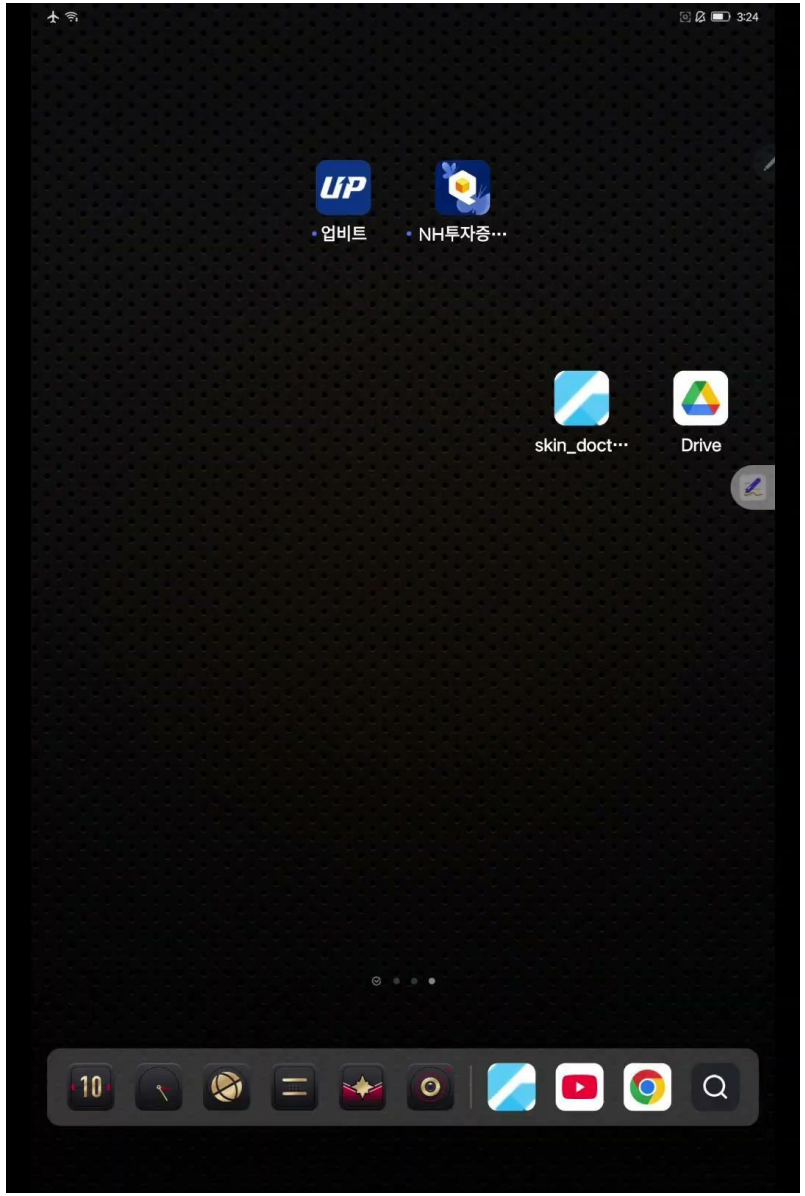
Classification Results:
Accuracy: 0.8849
Precision: 0.7261
Recall: 0.4672
F1 Score: 0.5686



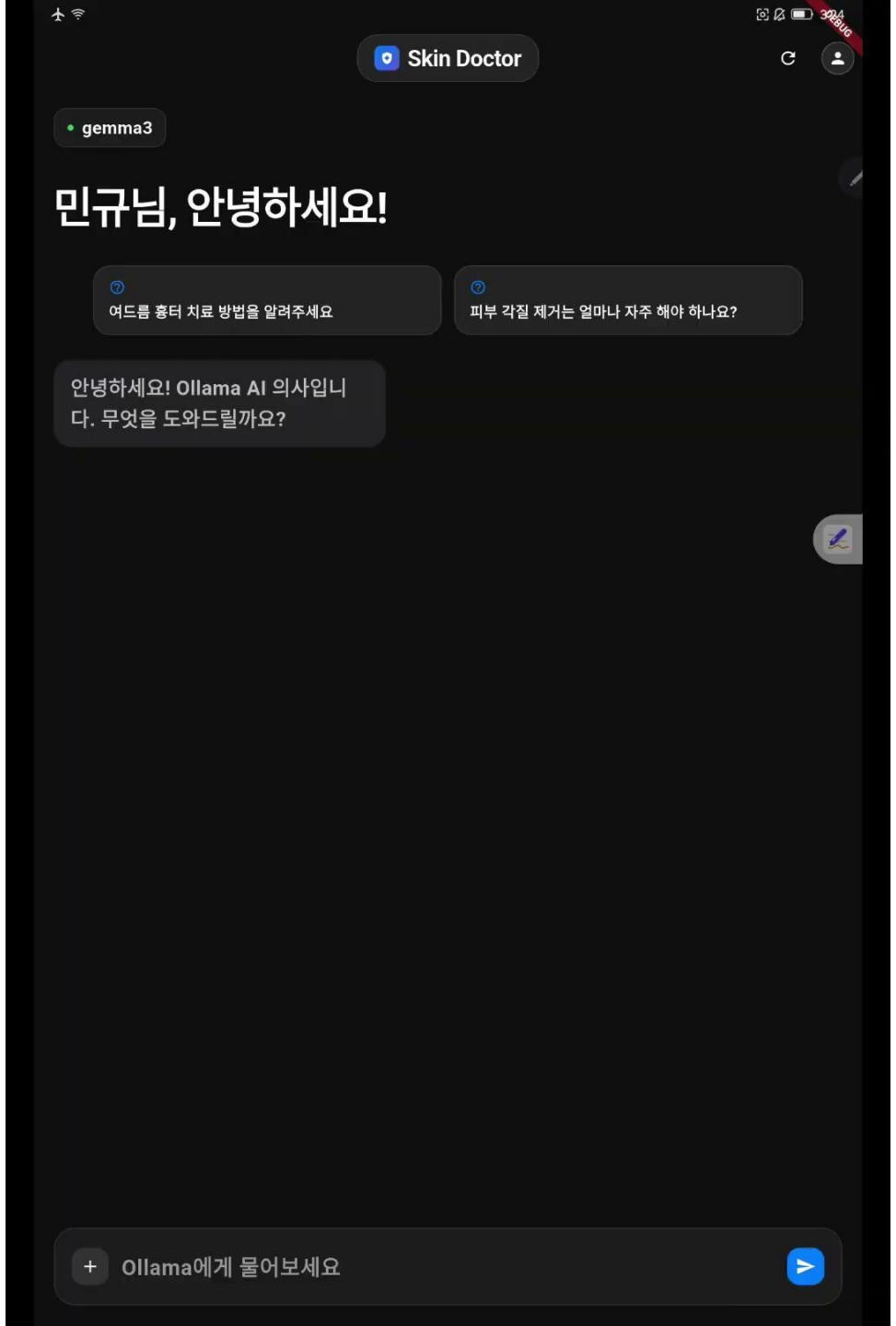


APP

material Design& scaffold layout




```dart
@override
 Widget build(BuildContext context) {
   return Scaffold(
     key: _scaffoldKey,
     backgroundColor: const Color(0xFF101010),
     appBar: AppBar(
       backgroundColor: const Color(0xFF101010),
       elevation: 0,
       title: Container(
         padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
         decoration: BoxDecoration(
           color: Colors.white.withOpacity(0.08),
           borderRadius: BorderRadius.circular(20),
           border: Border.all(color: Colors.white.withOpacity(0.1)),
         ),
```



카메라 초기화

```
final cameras = await availableCameras();
_controller = CameraController(cameras.first, ResolutionPreset.veryHigh, enableAudio: false);
```

사진 촬영

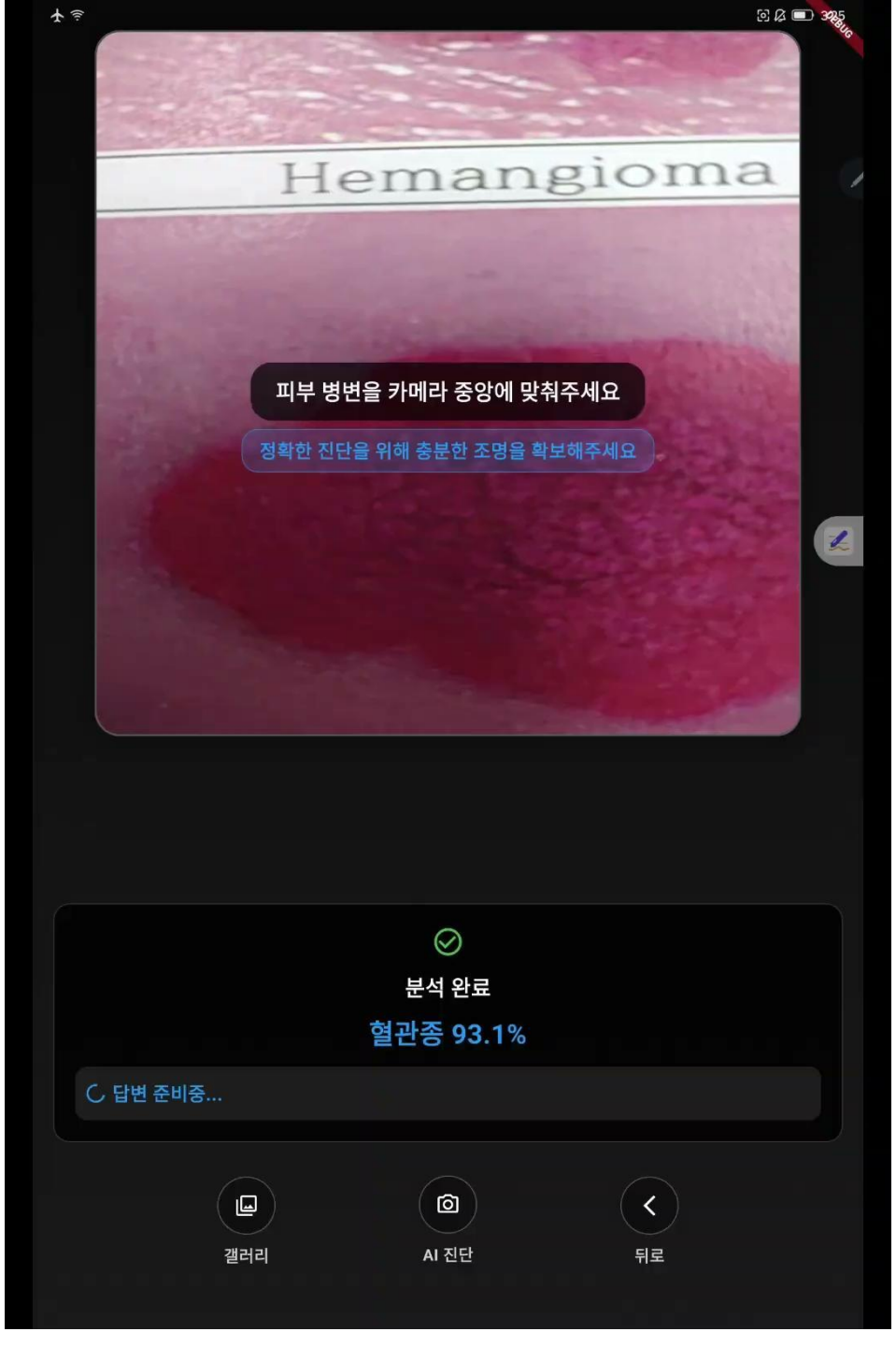
```
img.Image? image = img.decodeImage(bytes);
image = _cropToViewfinderArea(image);
```

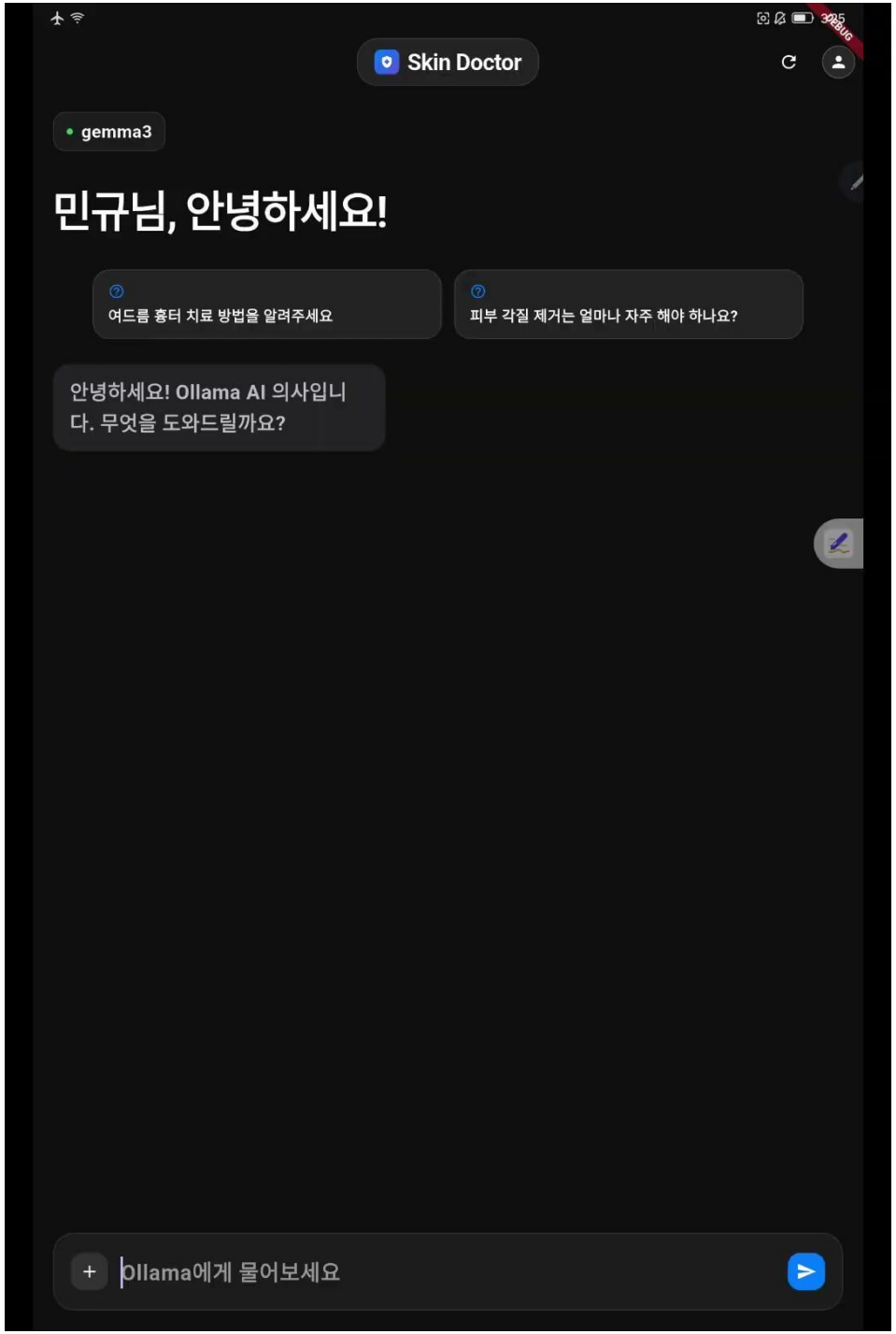
AI 분류

```
final pred = await SkinClassifier.classify(image);
```

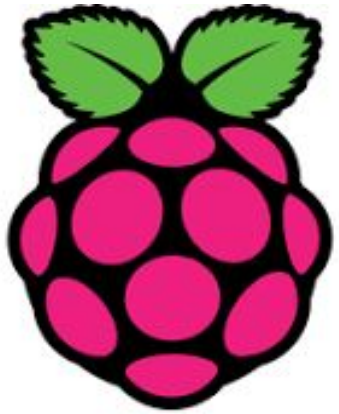
결과 종합

```
final confidence = confList.reduce((a, b) => a + b) / confList.length;
resultText = '$label ${(confidence * 100).toStringAsFixed(1)}%';
```


Hemangioma
피부 병변을 카메라 중앙에 맞춰주세요
정확한 진단을 위해 충분한 조명을 확보해주세요
분석 완료
혈관종 93.1%
답변 준비중...
갤러리
AI 진단
뒤로

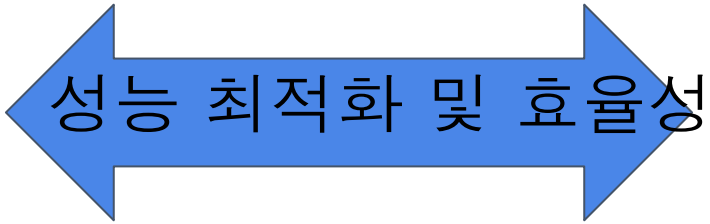

Skin Doctor
gemma3
민규님, 안녕하세요!
여드름 흉터 치료 방법을 알려주세요
피부 각질 제거는 얼마나 자주 해야 하나요?
안녕하세요! Ollama AI 의사입니다. 무엇을 도와드릴까요?
Ollama에게 물어보세요

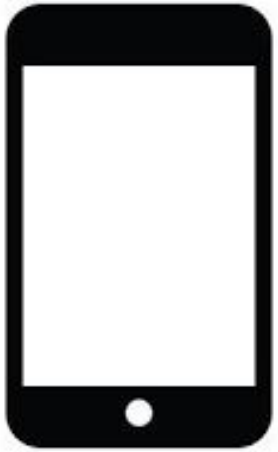
핵심 기술 4. Interface 2 -> Raspberry Pi



RaspberryPi

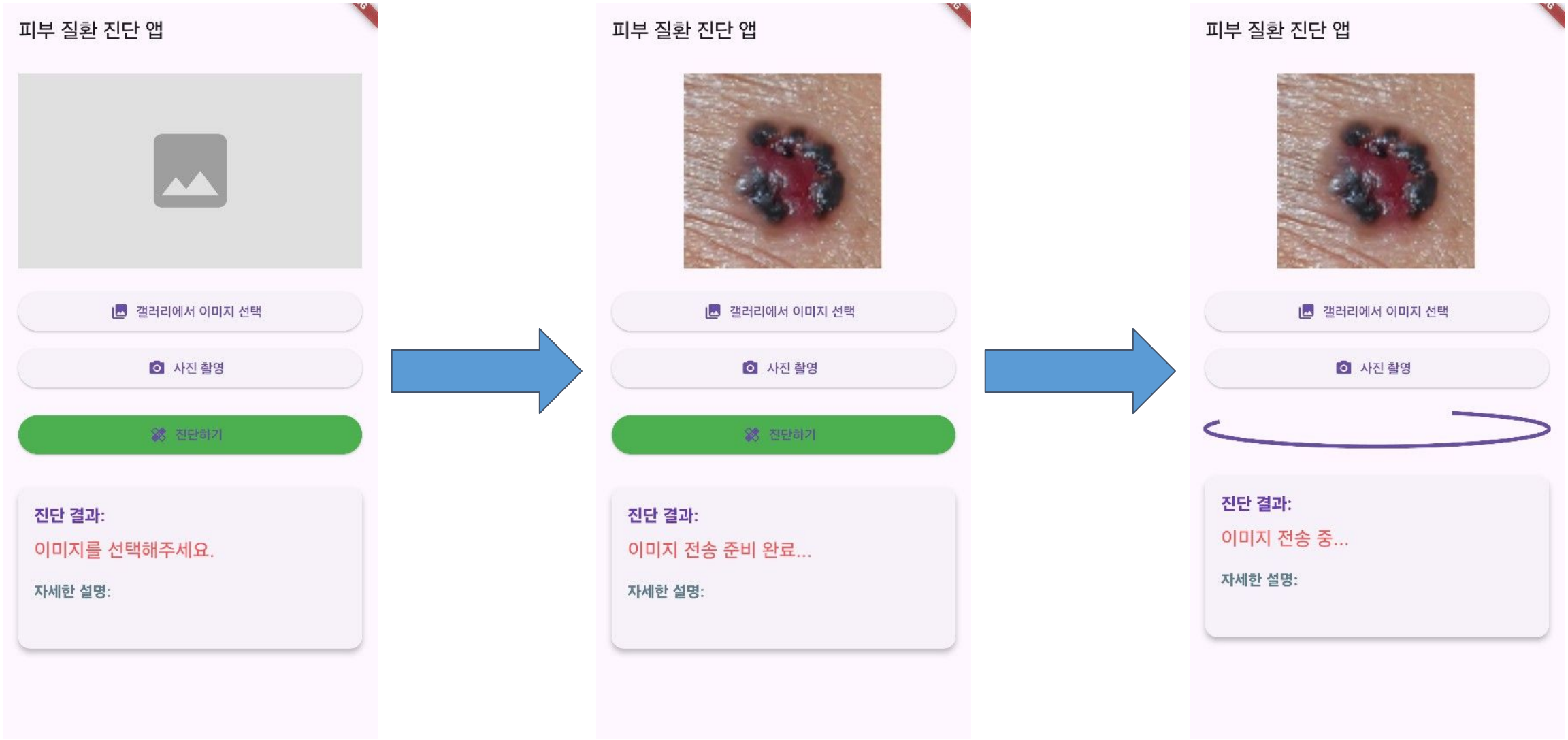
- API 엔드 포인트 제공
- 이미지 수신 및 전처리
- AI 모델 추론
- LLM 연동
- 취합 및 응답



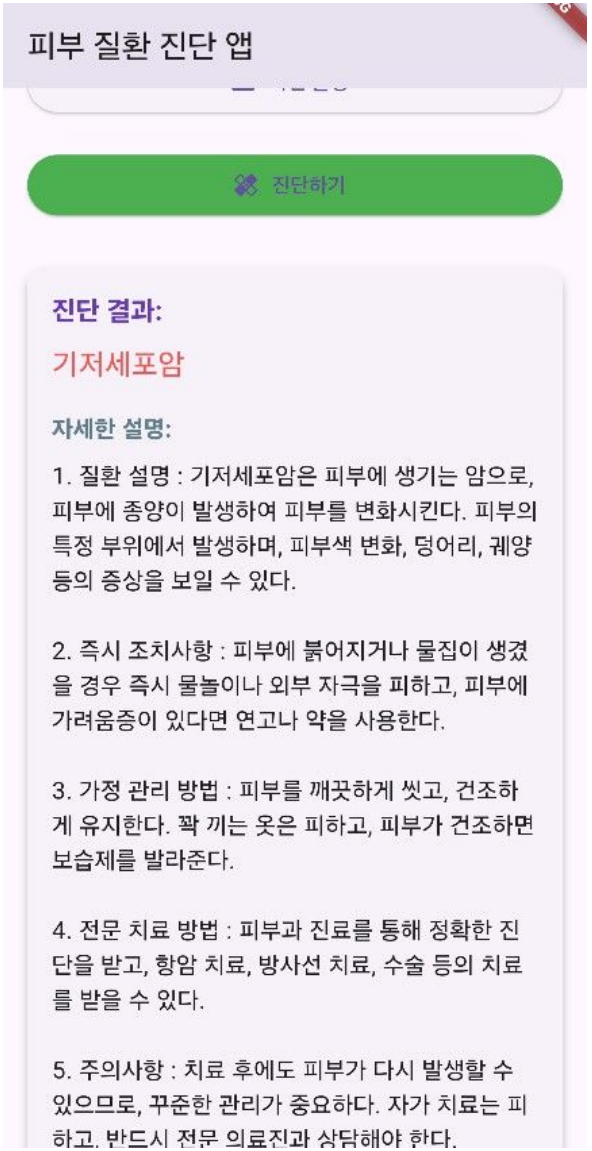


- UI 제공
- 이미지 획득
  (카메라 / 갤러리)
- 결과 표시
- 음성 안내 (TTS)

핵심 기술 4. Interface 2 -> Raspberry Pi



핵심 기술 4. Interface 2 -> Raspberry Pi



트러블슈팅 1. 세팅 문제

Windows 11

Linux™

Project_skindoctor Public

Pin | Watch 0

main | 1 Branch | 0 Tags | Go to file t | Add file | Code

MEGA-MK05 Update README.md — f800cff · 3 days ago — 8 Commits

captures	Add files via upload	3 days ago
model	Add files via upload	3 days ago
OPTIMIZED_option.md	Add files via upload	3 days ago
README.md	Update README.md	3 days ago
camera_h5_diagnosis.py	Update camera_h5_diagnosis.py	3 days ago
camera_onnx_diagnosis.py	Add files via upload	3 days ago
camera_onnx_optimized.py	Update camera_onnx_optimized.py	3 days ago
convert_h5_to_onnx.py	Add files via upload	3 days ago
outline.md	Update outline.md	3 days ago
requirements.txt	Add files via upload	3 days ago

README

🚀 설치 및 설정

▼ # window

1. 필요한 패키지 설치

```
# 기본 패키지
pip install -r requirements.txt

## 📁 사용 방법

### 1단계: 모델 변환

먼저 H5 모델을 ONNX/TFLite로 변환합니다:

```bash
python3 convert_h5_to_onnx.py
```

변환 결과:

- ✅ `skin_model.onnx` - ONNX 모델 생성
- ✅ `skin_model_quantized.tflite` - 8비트 양자화 TFLite 모델 생성
- ✅ `skin_model_quantized_dynamic.onxx` - 8비트 동적 양자화 onxx 모델 생성
- ✅ `skin_model_static_dynamic.onxx` - 8비트 정적 양자화 onxx 모델 생성

2단계: 진단 프로그램 실행

```
# ollama 실행 (터미널 하나 더 열어서 진행행)
ollama run gemma3:1b
```

```
# h5 기본
python camera_h5_diagnosis.py

# onxx 기본
python camera_onnx_diagnosis.py

# onxx runtime 적용
python camera_onnx_optimized.py
```

▼ # Linux

1. 필요한 패키지 설치

```
# 기본 패키지 설치
pip install -r requirements.txt

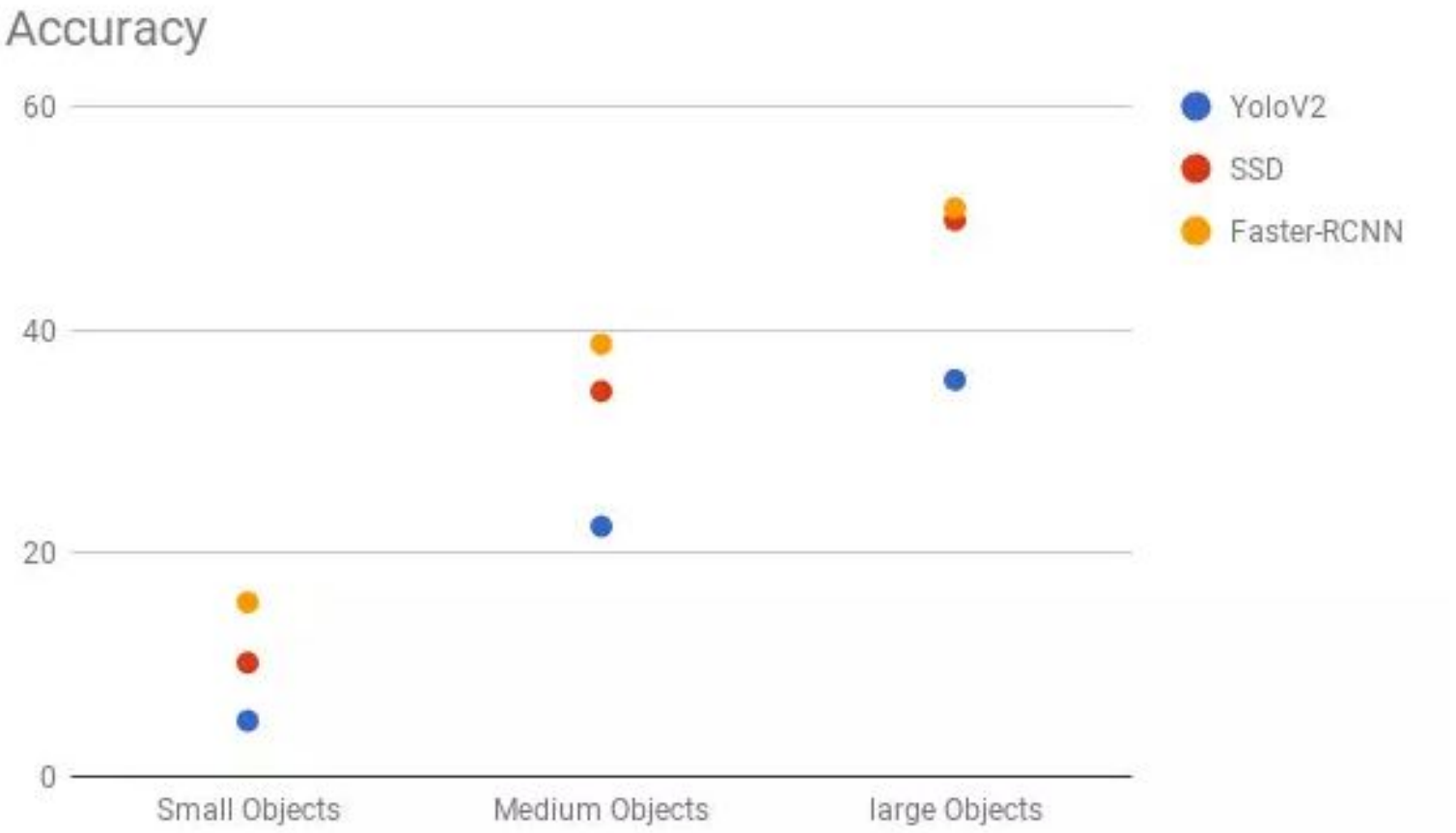
# Pillow 최신 버전 업그레이드 (텍스트 렌더링 오류 방지용)
pip install --upgrade pillow

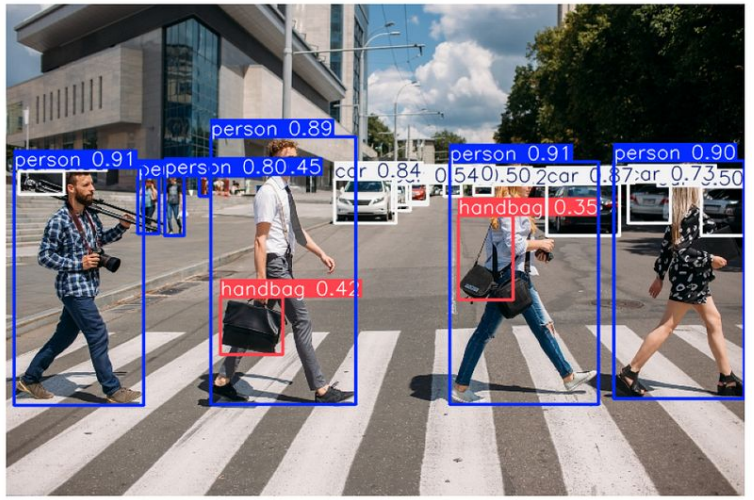
# 리눅스(Ubuntu) 환경에서 한글 폰트가 깨질 경우 아래 명령어로 나눔글꼴 설치
sudo apt update
sudo apt install fonts-nanum

💡fonts-nanum은 한글을 깨지지 않게 표시하기 위해 필요합니다. 설치 후 코드에서 다음과 같이 경로를 설정하세요:
font_path = "/usr/share/fonts/truetype/nanum/NanumGothic.ttf"

# Ollama 설치 (Snap 기반)
sudo snap install ollama
```
```

트러블슈팅 2. Yolo -> Faster-RCNN

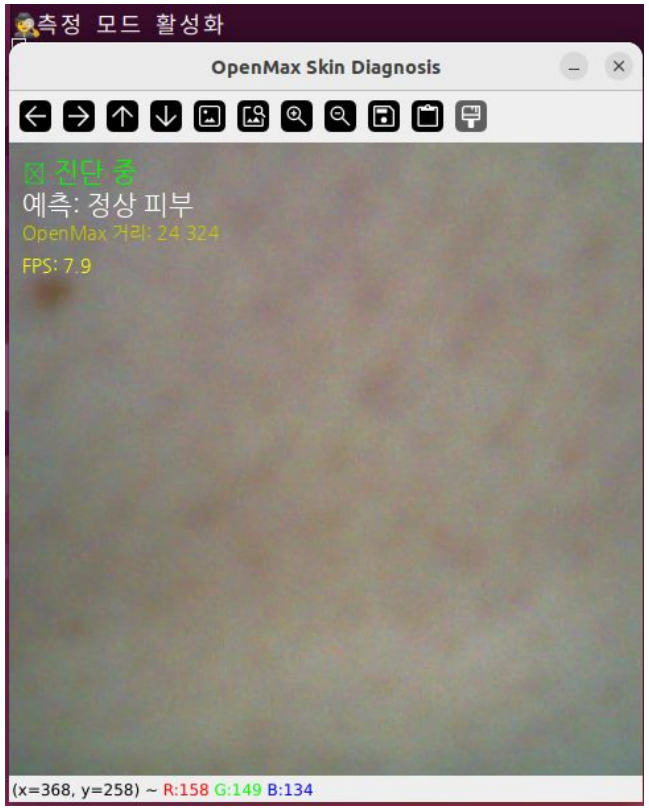




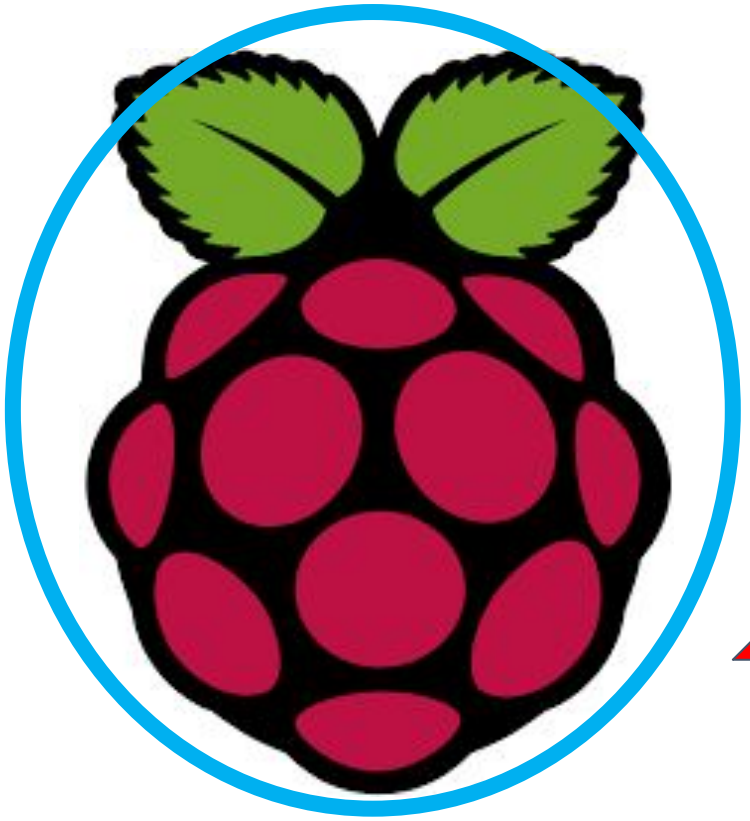
VS

트러블슈팅 3. 모델 성능 향상

feature_vec	입력 이미지의 시각적 특징을 담은 벡터
MAV	각 클래스의 평균 특징 벡터
거리	예측 이미지가 각 클래스와 얼마나 유사한지를 수치화
threshold	값이 너무 크면 Unknown 으로 분류



고찰 – 제한된 성능 해결 FPGA 가속



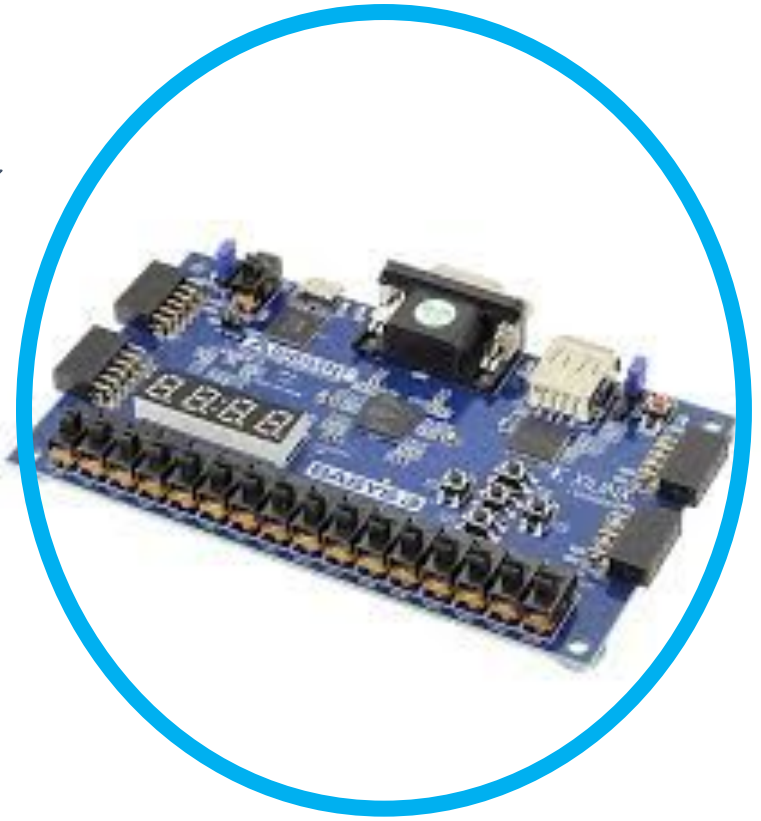
Master

이미지와 명령 전송

SPI

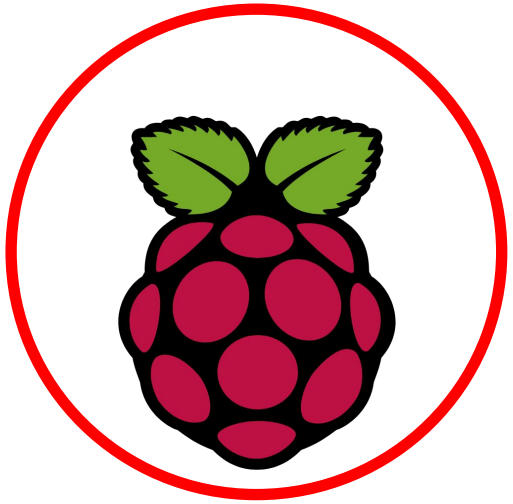
Pi는 Camera, LLM, TTS
FPGA는 ONNX 기반의 변환된
CNN 구조와 가중치 연산

CNN 연산 후 결과만 Pi로 전송



Slave

고찰 – 제한된 성능 해결 FPGA 가속



2.4GHz

ResNet-50
224×224 RGB 이미지 1장 처리 시 약 **3.8 GFLOPs**의 연산

병렬 처리

100MHz

FLOPs:
부동소수점 연산
GOPS:
고정소수점 연산

y=a×b+c
2개의 기본 연산

4코어×2.4GHz×4FLOPs/cycle ≈ 38.4GFLOPs

$$T_{cpu} = 3.8GFLOPs \Big/ \frac{12GFLOPs}{\text{sec}}$$

≈ 0.316 sec ≈ 316ms

3.8 GFLOPs ≈ 7.6 GOPS (assuming 2 ops/FLOP for conv)

$$T_{cpu} = 7.6GFLOPs \Big/ \frac{10GFLOPs}{\text{sec}}$$

≈ 0.76 sec ≈ 76ms

이론상 성능

고찰 – 제한된 성능 해결 FPGA 가속

현실적 제약 -> Basys3 성능 문제로 30% 가속 사용

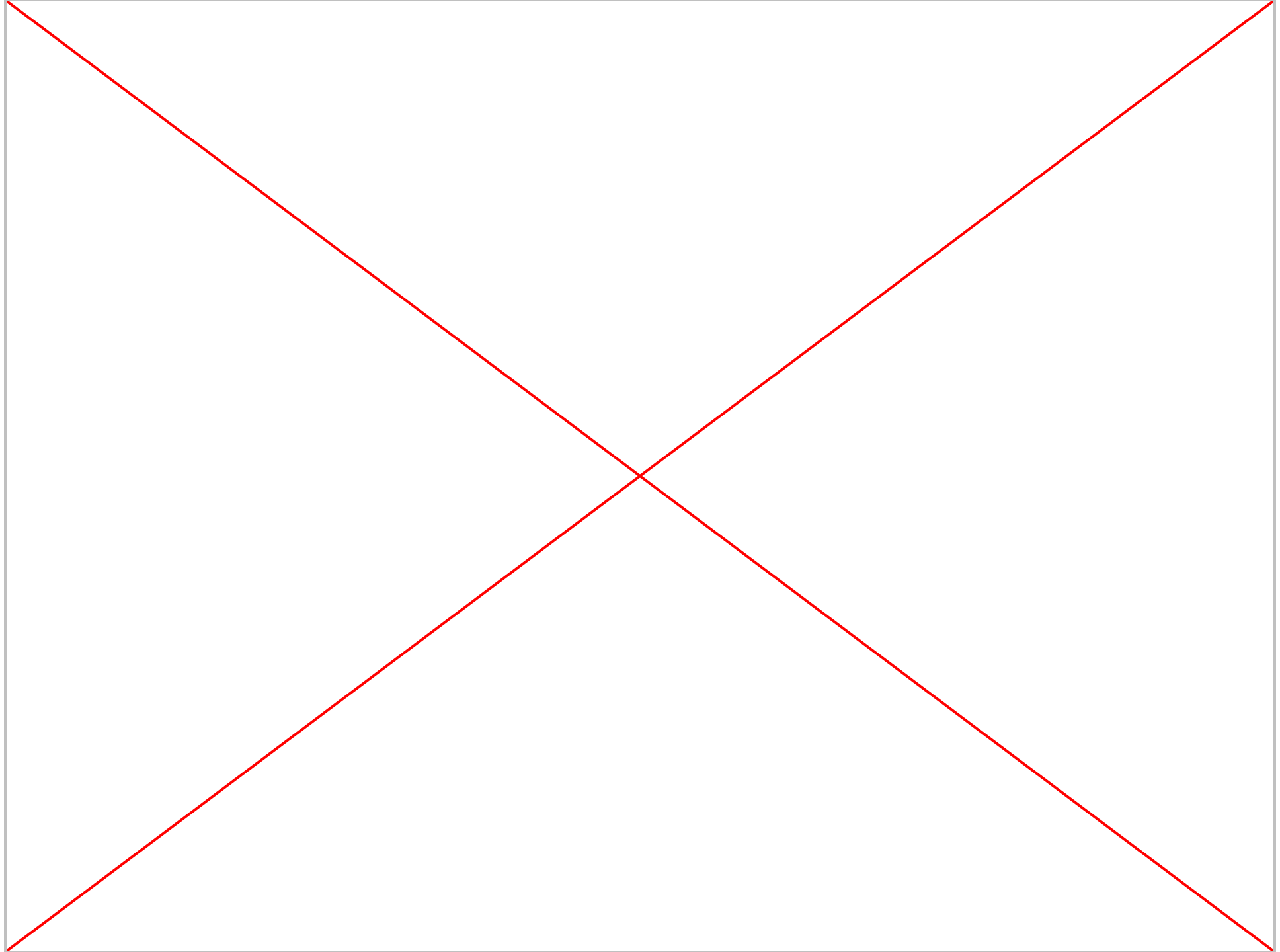
$$T_{total} = T_{cpu}\ X\ 0.7 + T_{FPGA}\ X\ 0.3$$

$$\boldsymbol{T_{total}} = (316\text{ms x } 0.7) + (76\text{ms x } 0.3) =$$
$$221\text{ms} + 22.8\text{ms} = 243.8\text{ms}$$

316ms → 244ms / 약 23% 속도 향상

시연

라즈베리파이



감사합니다
Q&A