전이학습(TL)을 통한 피부질환 판별

팀명 : 스킨푸우드
팀장: 이혜은
팀원: 노윤지, 송나단, 성현아

목차

01 연구 배경

02 연구 목적

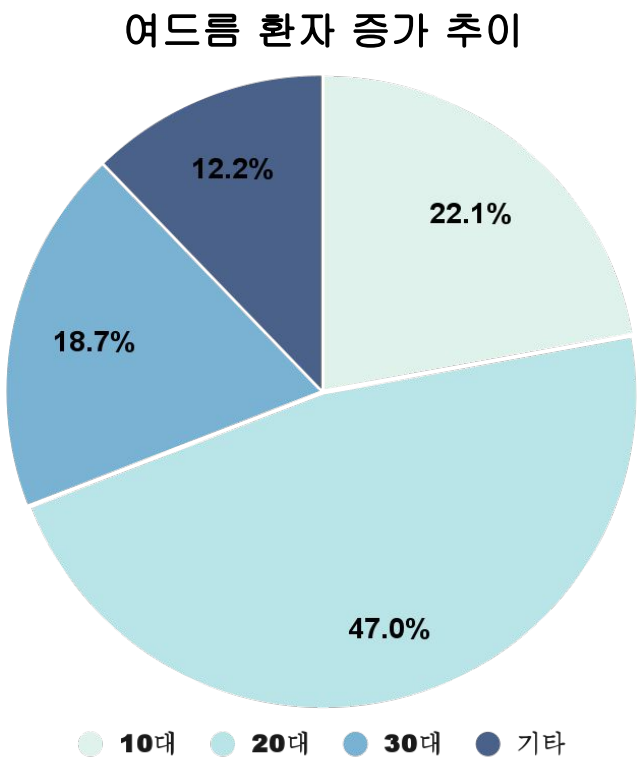
03 연구 방법

04 연구 결과

05 결과 분석

06 시행 착오

01. 연구 배경



여드름 환자 매년 약 10% 증가

- 2018년 대비 2022년 29.4% 증가
- 진료비 1인당 약 8만 2천원

환절기로 인한 피부자극

- 바쁜 스케줄로 인한 치료 방치
- 외부 환경 변화로 인한 피부 자극

02. 연구 목적

피부질환 데이터에 대한 모델의 정확도 향상 및 앱 배포

1. 외형적으로 유사한 환부를 정확히 판별하는지
2. 앱을 배포하여 간단한 피부 상태 확인

03. 연구 방법 - 데이터 전처리 및 모델 구현

데이터 전처리

- **데이터 구축 & 증강**
  - Dermnet Data (Kaggle)
  - Transforms 모듈
    - Resize, Random Crop
    - Rotate, Flipped

학습 모델 구현

- **학습 모델**
  - ResNet 50
    - ImageNet으로 학습된 사전 학습 모델 적용

03. 연구 방법 - 데이터 전처리 및 모델 구현

데이터 전처리 및 학습결과

모델	Img Size	Epochs	Batch Size	Test Accuracy	Test Loss
ResNet 50	224	10 (추가학습: 30)	32	0.79	0.97

03. 연구 방법 - 데이터 전처리 및 모델 구현 코드

데이터 증강 코드

```
# 데이터 증강 정의
train_transform = transforms.Compose([
    transforms.Resize((224, 224)),  # Resizing to 224x224
    transforms.RandomHorizontalFlip(),  # Randomly flip images horizontally
    transforms.RandomRotation(degrees=15),  # Randomly rotate images within ±15 degrees
    transforms.RandomAffine(degrees=0, translate=(0.1, 0.1)),  # Slightly translate the images
    transforms.RandomCrop(size=224, padding=10),  # Random crop with padding
    transforms.ToTensor(),  # Convert images to tensors
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])

test_transform = transforms.Compose([
    transforms.Resize((224, 224)),  # No augmentation for test data, just resize
    transforms.ToTensor(),  # Convert to tensor
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
```

03. 연구 방법 - 데이터 전처리 및 모델 구현 코드

ResNet 50 코드

```
#학습 진행
model = train_model(model, train_loader, criterion, optimizer, num_epochs=10)
```

```
# ResNet50 모델 불러오기
model = models.resnet50(pretrained=True)

# 출력층을 데이터셋 클래스 수에 맞게 수정
model.fc = torch.nn.Linear(model.fc.in_features, len(train_dataset.classes))  # 클래스 수에 맞게 변경

# 손실 함수 정의
criterion = torch.nn.CrossEntropyLoss()

# 옵티마이저 정의
optimizer = optim.Adam(filter(lambda p : p.requires_grad, model.parameters()), lr=0.001)
```

```
#추가 학습 진행
model = train_model(model, train_loader, criterion, optimizer, num_epochs=15)

#학습 끝내고 저장하기(클래스 이름도 포함)
torch.save({'model_state_dict' : model.state_dict(),
            'class_names' : test_dataset.classes}, './ResNet50.pth')
```

03. 연구 방법 - 앱 구현

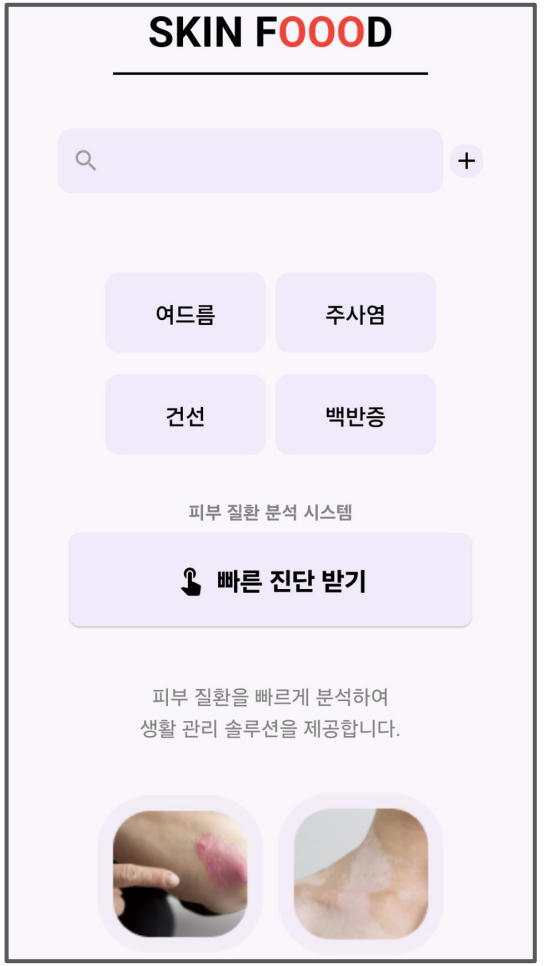
앱 구현 (Android)

Frontend

- Flutter를 활용하여 애플리케이션 화면 구현

Backend

- Flask를 활용하여, 모델의 결과를 반환하는 API 개발
- SQLAlchemy를 사용해 상세 설명을 데이터베이스에서 조회해 가져오도록 구현
- pythonanywhere 사용해 API 배포



03. 연구 방법 - 데이터 전처리 및 모델 구현 코드

앱 구현 코드

```python
def post(self):
    # 이미지 파일을 요청에서 가져오기
    if 'image' not in request.files:
        return {"error": "No image file found in the request."}, 400

    image_file = request.files['image']

    # 이미지 파일을 열고 예외 처리
    try:
        image = Image.open(image_file.stream).convert("RGB")
    except Exception as e:
        return {"error": f"Failed to open image: {str(e)}"}, 400

    # 이미지 전처리 및 모델에 입력
    predict_image = self.transform(image).unsqueeze(0).to(self.device)
    with torch.no_grad():
        output = self.model(predict_image)
        _, predicted = torch.max(output, 1)

    # 예측된 클래스 이름
    if len(self.class_names) > predicted.item():
        predicted_class = self.class_names[predicted.item()]
    else:
        return {"error": "Predicted class index out of range."}, 500
```

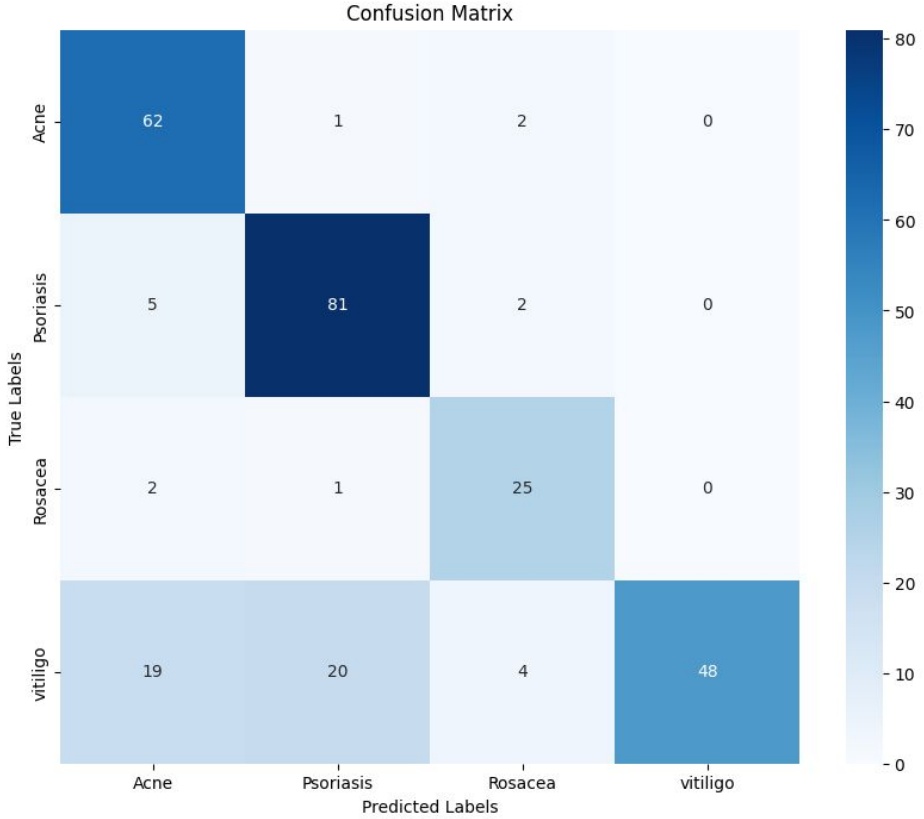
03. 연구 방법 - 데이터 전처리 및 모델 구현 코드

웹 구현 코드

```
# 해당 클래스에 대한 설명을 데이터베이스에서 조회
class_info = ClassDescription.query.filter_by(class_name=predicted_class).first()
if class_info:
    response = {
        "disease_name": class_info.disease_name,
        "description": class_info.description
    }
else:
    response = {
        "disease_name": "No Disease Name available",
        "description": "No description available."
    }

return response, 200
```

04. 연구 결과



- **건선(Psoriasis) 높은 정확도**
  - 88건 중 81건 올바르게 분류
- **백반(Vitiligo) 오류 많음**
  - 19건 여드름 & 20건 건선으로 오분류
- **여드름(Acne) /주사병(Rosacea)**
  - 전반적으로 가장 잘 분류

04. 연구 결과 - 어플 시연 영상 https://www.youtube.com/shorts/cif_keXog50

05. 결과 분석

- **유사한 병변 부위를 가진 경우 성능 저하**
  - 여드름(Acne) & 모낭염
  - 건선 & 습진

여드름 | 모낭염

- **이미지에 다른 요소가 포함될 경우 분류 혼란**
  - 백반증(Vitiligo)의 경우, 색으로 구분하는 경향이 있어, 그림자 혹은 머리카락 포함시 성능 저하

【백반증】

06. 시행 착오

01 EfficientNet B5는 모델이 너무 무거워 GPU 한계로 실패

02 EfficientNet B2는 테스트 정확도가 약 40%로 성능 개선 실패

03 유사 병변을 가진 질환끼리의 분류 혼란으로 반복적인 Class 교체

04 Parameter 튜닝에 따른 성능 차이가 심함

감사합니다

팀장: 이혜은
팀원: 노윤지, 송나단, 성현아