

# 기계학습

Term Project

2021년 12월 21일  
발표자 : 김영환

# 두피 상태 진단

---

두피 질환 진단 및 치료 과정에서 필요한 두피 이미지 데이터를 기반으로 모델 학습  
두피 상태를 구분하는 7개의 레이블 분류

# Contents

- 01. 목표
- 02. 분류 개요
- 03. 데이터
- 04. 데이터 처리
- 05. 모델 학습
- 06. 결과 분석
- 07. 발전 방향
- 08. 느낀점

# 목표

## Goal

1. 두피 관련 질환으로 고민하는 수많은 사람들에게 맞춤형 정보 제공
2. VGG16 아키텍처 활용 모델, VGG16 Transfer learning, ResNet Transfer learning 모델의 성능비교
3. 분류 결과(두피 유형)를 바탕으로 추가적인 정보를 제공할 수 있도록 데이터셋에 포함된 메타 데이터를 적절히 활용

# 목표

## Goal

1. 두피 관련 질환으로 고민하는 수많은 사람들에게 맞춤형 정보 제공
2. VGG16 아키텍처 활용 모델, VGG16 Transfer learning, ResNet Transfer learning 모델의 성능비교
3. 분류 결과(두피 유형)를 바탕으로 추가적인 정보를 제공할 수 있도록 데이터셋에 포함된 메타 데이터를 적절히 활용

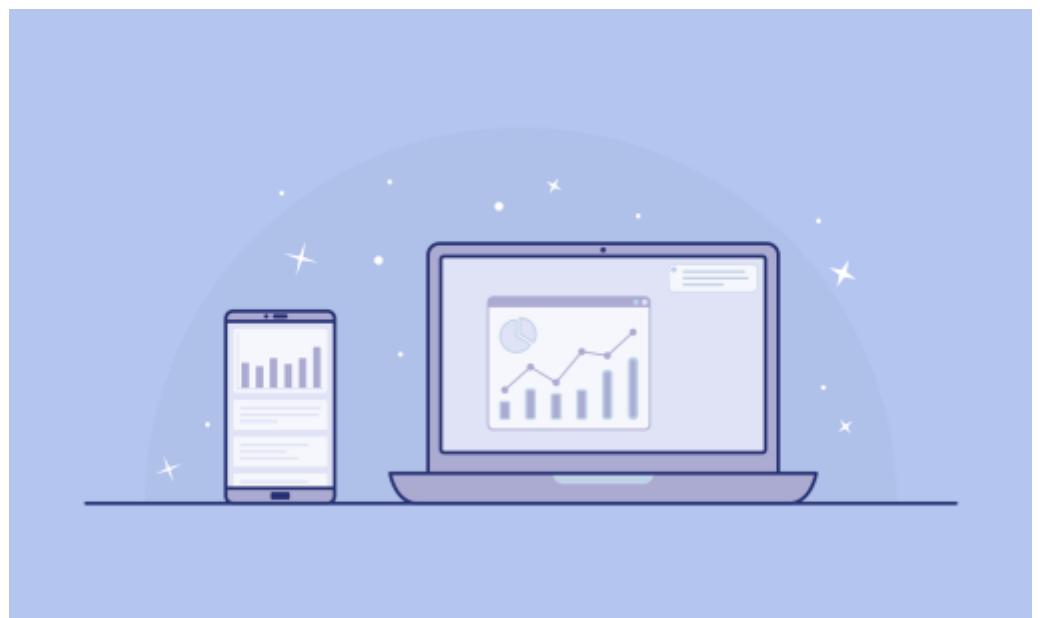
# 목표

Goal



두피 상태에 따른 메타 데이터 분석  
연령, 성별, 염색 주기, 펌 주기, 탈모 관련 제품 사용 정보 사용

분석 결과를 활용해 맞춤형 정보 제공  
ex) 해당 성별, 연령대의 평균 염색, 펌 주기 혹은 사용 제품 추천



# 분류 개요

## Classification



# 데이터

## Data

### AI-hub 두피 데이터

URL : <https://aihub.or.kr/aidata/30758>

기존 육안으로 진단영상과 비교해 보고 진단자가 수동으로 유사한 영상과 비교하여 진단값을 선택하고 제품을 추천해 주는 기존 방식의 시스템에서 두피측정 빅데이터를 이용한 인공지능(AI) 진단(Deebleaning)으로 자동 진단하고 자동제품 추천시스템 제공으로 고객의 신뢰 및 진단의 정확성으로 사업 경쟁력 확보

구분	성과기준	결과
데이터 구축 증상별 구축 건수	수집	100,000건
	가공	100,000건
	검수	100,000건
	미세각질	1,250건 이상
	피지과다	1,250건 이상
	모낭사이홍반	1,250건 이상
	모낭홍반/농포	1,250건 이상
	비듬	1,250건 이상
	탈모	1,250건 이상
	양호	-
		811건

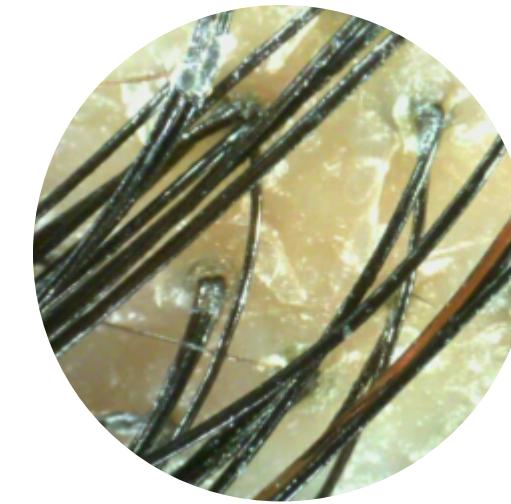
# 데이터

## Data - 각 레이블의 중증 상태 이미지 첨부



미세각질

두피에 미세하게 각질이 생겨있는 상태



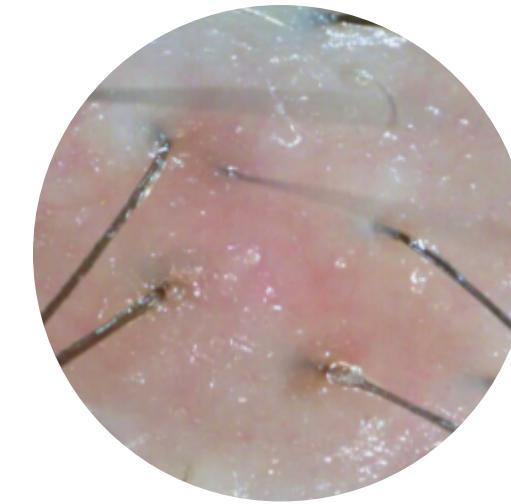
피지과다

육안으로도 두피 및 모발에 상당한 유분기가 느껴지는 상태



비듬

두피에 비듬이 많이 생긴 상태



탈모

탈모가 상당히 진행된 상태

# 데이터 처리

## Data

```
# Unzip
path_to_zip_file = '/content/HairLossData.zip'
directory_to_extract_to = '/content'

with zipfile.ZipFile(path_to_zip_file, 'r') as zip_ref:
    zip_ref.extractall(directory_to_extract_to)
```

Class 별로 정리된 Image File의 압축을 해제한다.

```
trainDataPath = '/content/HairLossData/Training/image'
validDataPath = '/content/HairLossData/Validation/image'

trData = ImageDataGenerator()
trainData = trData.flow_from_directory(directory=trainDataPath, target_size=(224,224))
valData = ImageDataGenerator()
validData = valData.flow_from_directory(directory=validDataPath, target_size=(224,224))
```

ImageDataGenerator를 사용해 Class 별로 정리된 데이터를 224 \* 224 크기로 받는다.

# VGG16

Conv2D \* 2 (64 channel)  
MaxPool2D (2 \* 2)  
Conv2D \* 2 (128 channel)  
MaxPool2D (2 \* 2)  
Conv2D \* 3 (256 channel)  
MaxPool2D (2 \* 2)  
Conv2D \* 3 (512 channel)

A diagram illustrating the VGG16 architecture. On the left, a vertical list of layers is shown: Conv2D \* 2 (64 channel), MaxPool2D (2 \* 2), Conv2D \* 2 (128 channel), MaxPool2D (2 \* 2), Conv2D \* 3 (256 channel), MaxPool2D (2 \* 2), and Conv2D \* 3 (512 channel). A large brown arrow originates from the bottom of the MaxPool2D (2 \* 2) layer and points diagonally upwards and to the right, indicating the flow of data to the subsequent layers. To the right of this arrow, a vertical list of layers continues: MaxPool2D (2 \* 2), Conv2D \* 3 (512 channel), MaxPool2D (2 \* 2), Flatten, Dense \* 2 (4096), and Dense (output, softmax).

MaxPool2D (2 \* 2)  
Conv2D \* 3 (512 channel)  
MaxPool2D (2 \* 2)  
Flatten  
Dense \* 2 (4096)  
Dense (output, softmax)

# VGG16

Optimizer : Adam

Metric : accuracy

CheckPoint & EarlyStopping(patience=20)

Epoch : 100

```
Epoch 99/100
100/100 [=====] - 84s 842ms/step - loss: 0.8011 - accuracy: 0.7272 - val_loss: 0.7979 - val_accuracy: 0.7250
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
Epoch 100/100
100/100 [=====] - 84s 838ms/step - loss: 0.8150 - accuracy: 0.7113 - val_loss: 0.8235 - val_accuracy: 0.6969
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
```

loss : 0.81

accuracy : 0.71

val\_loss : 0.82

val\_accuracy : 0.69

# VGG16 Transfer Learning

사전 학습 모델 사용

```
from keras.applications.vgg16 import VGG16  
vggmodel = VGG16(weights='imagenet', include_top=True)
```

마지막 layer를 7개의 class 구분이 가능하도록 구성

```
H = vggmodel.layers[-2].output  
predictions = Dense(7, activation="softmax")(H)  
model_final = Model(inputs = vggmodel.input, outputs = predictions)
```

# VGG16 Transfer Learning

Optimizer : Adam

Metric : accuracy

CheckPoint & EarlyStopping(patience=40)

Epoch : 100

```
Epoch 64/100
2/2 [=====] - ETA: 0s - loss: 0.8678 - accuracy: 0.6562
Epoch 00064: val_accuracy did not improve from 0.75000
2/2 [=====] - 2s 1s/step - loss: 0.8678 - accuracy: 0.6562 - val_loss: 1.7607 - val_accuracy: 0.5000
Epoch 00064: early stopping
<keras.callbacks.History at 0x7fca5c2e7790>
```

val\_accuracy : 0.75

# ResNet

## Model

```
# base pre-trained model
base_model = ResNet50(input_tensor=input_tensor,weights='imagenet',include_top=False)

for layer in base_model.layers:
    layer.trainable=False

x = base_model.output
x = GlobalAveragePooling2D(data_format='channels_last')(x)
x = Dense(num_classes, activation='softmax')(x)

updatedModel = Model(base_model.input, x)
```

## Augmentation

```
train_datagen = ImageDataGenerator(
    rotation_range=90,
    horizontal_flip=True,
    vertical_flip=True,
    zoom_range=0.4)
```

# ResNet

Optimizer : Adam

Metric : accuracy

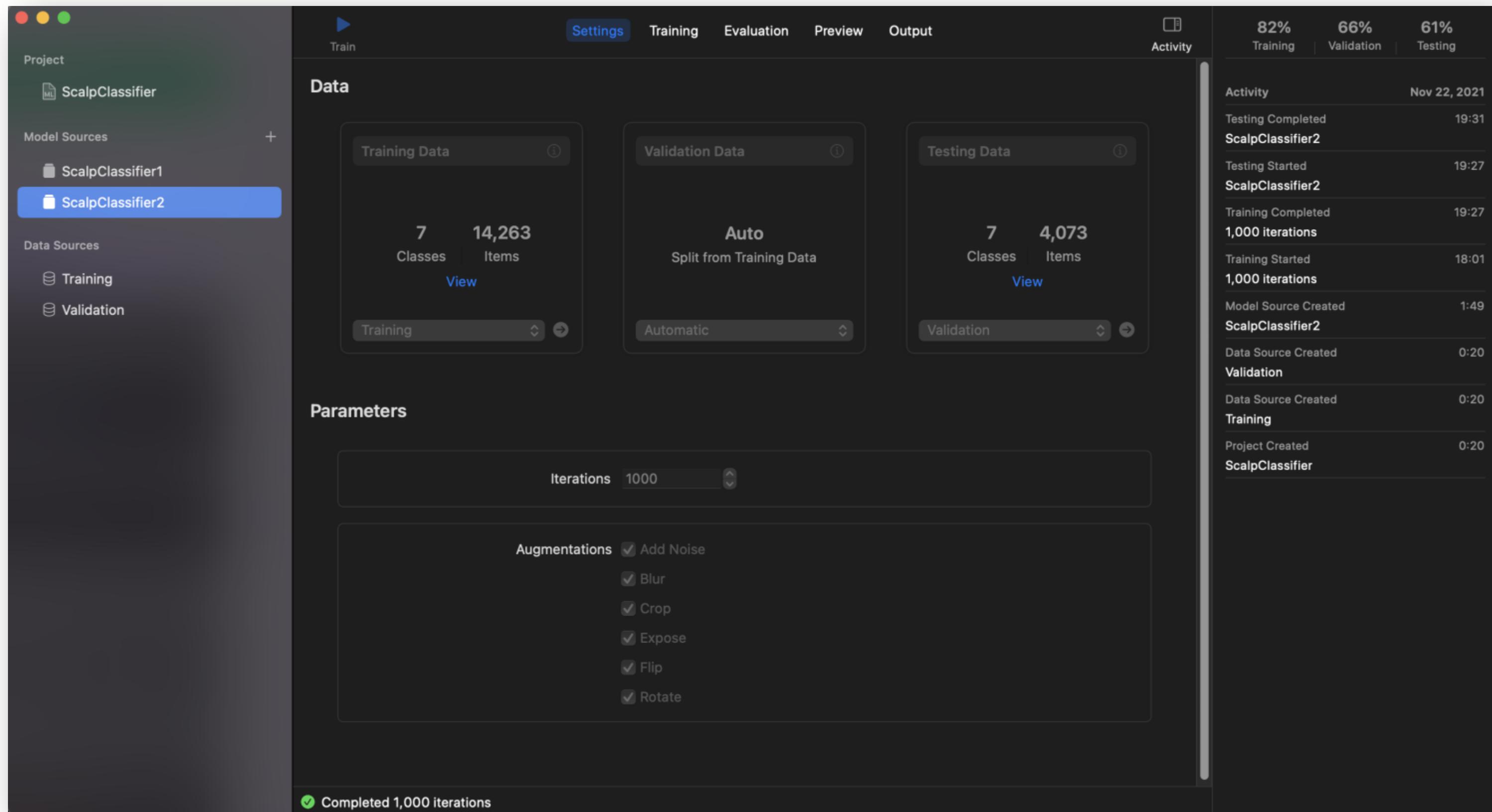
CheckPoint & EarlyStopping(patience=10)

Epoch : 100

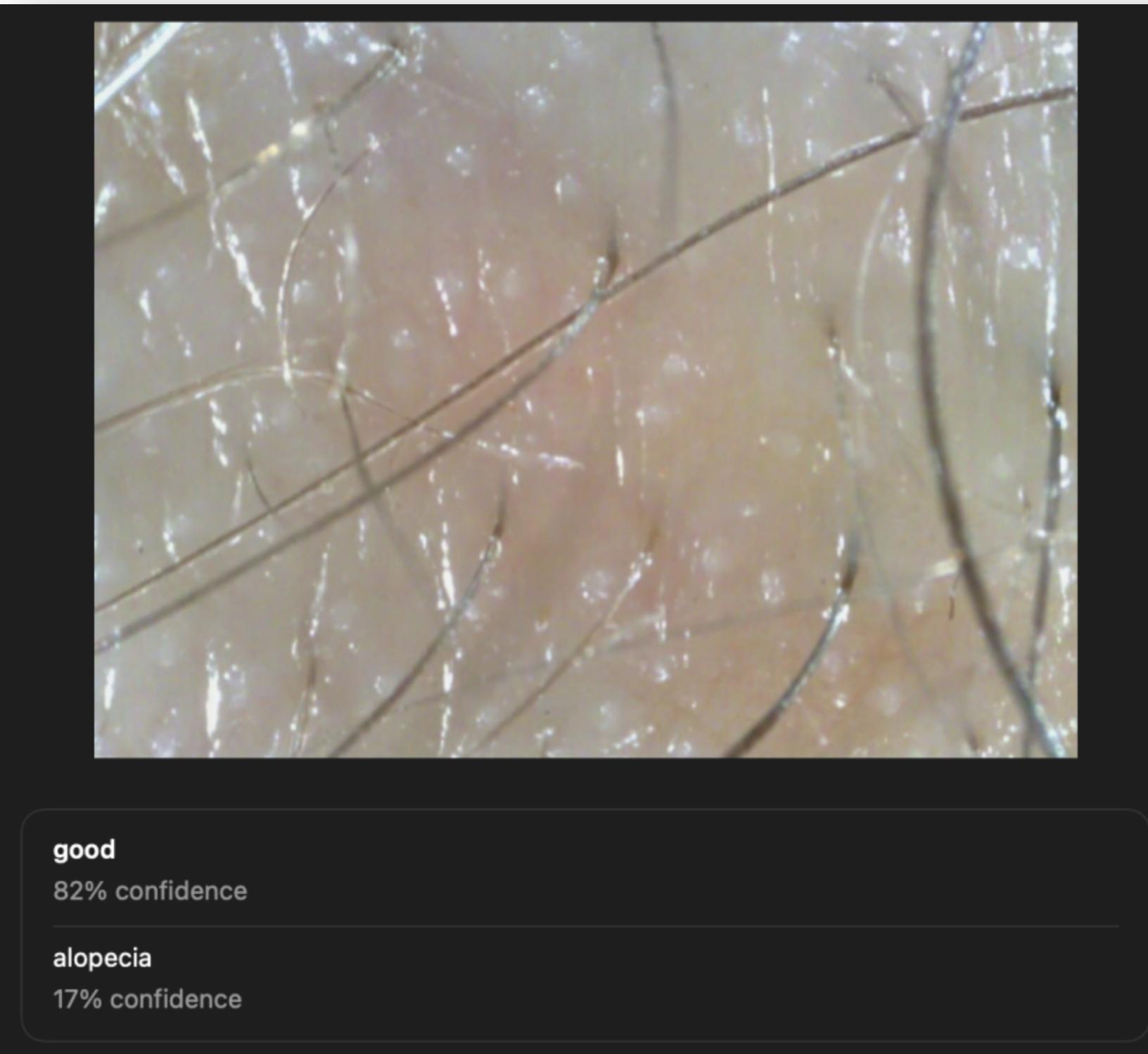
```
Epoch 99/100
891/891 [=====] - ETA: 0s - loss: 1.0549 - accuracy: 0.6157
Epoch 00099: val_accuracy improved from 0.57308 to 0.57456, saving model to resnet50.h5
891/891 [=====] - 273s 306ms/step - loss: 1.0549 - accuracy: 0.6157 - val_loss: 1.0877 - val_accuracy: 0.5746
Epoch 100/100
891/891 [=====] - ETA: 0s - loss: 1.0560 - accuracy: 0.6161
Epoch 00100: val_accuracy did not improve from 0.57456
891/891 [=====] - 272s 305ms/step - loss: 1.0560 - accuracy: 0.6161 - val_loss: 1.0884 - val_accuracy: 0.5731
```

val\_accuracy : 0.57

# CreateML



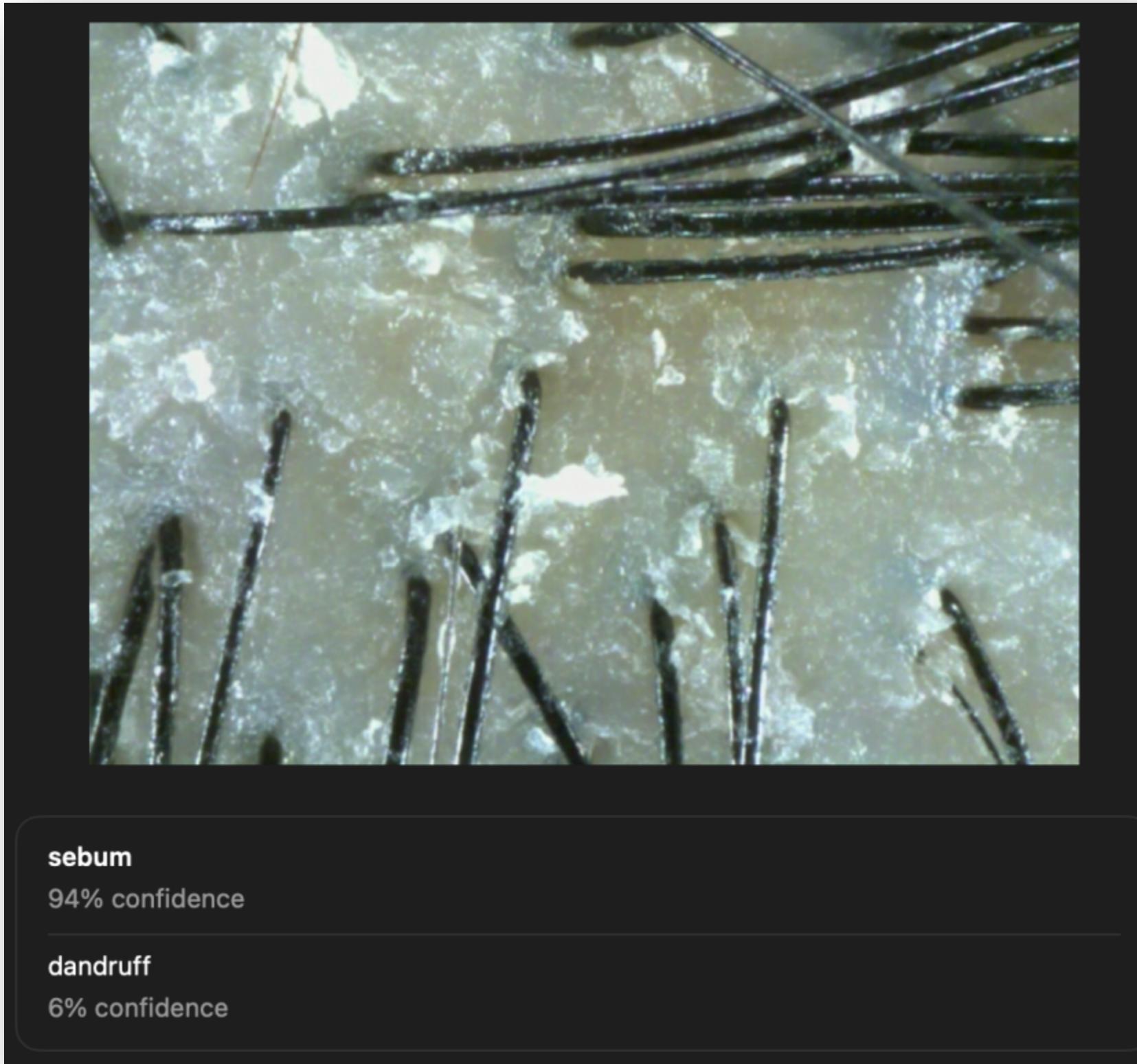
# CreateML



Original Class label : alopecia

실제로는 alopecia 레이블을 가진 이미지이다.  
해당 두피 상태의 경우,  
탈모와 동시에 좋은 두피 컨디션을  
가지고 있기 때문에  
good, alopecia 레이블이  
결과로 나오는 것을 확인할 수 있다.

# CreateML



Original Class label : sebum

실제로는 sebum 레이블을 가진 이미지이다.  
해당 두피 상태의 경우,  
피지와 동시에 비듬을 유발할 여지를  
가지고 있기 때문에  
sebum, dandruff 레이블이  
결과로 나오는 것을 확인할 수 있다.

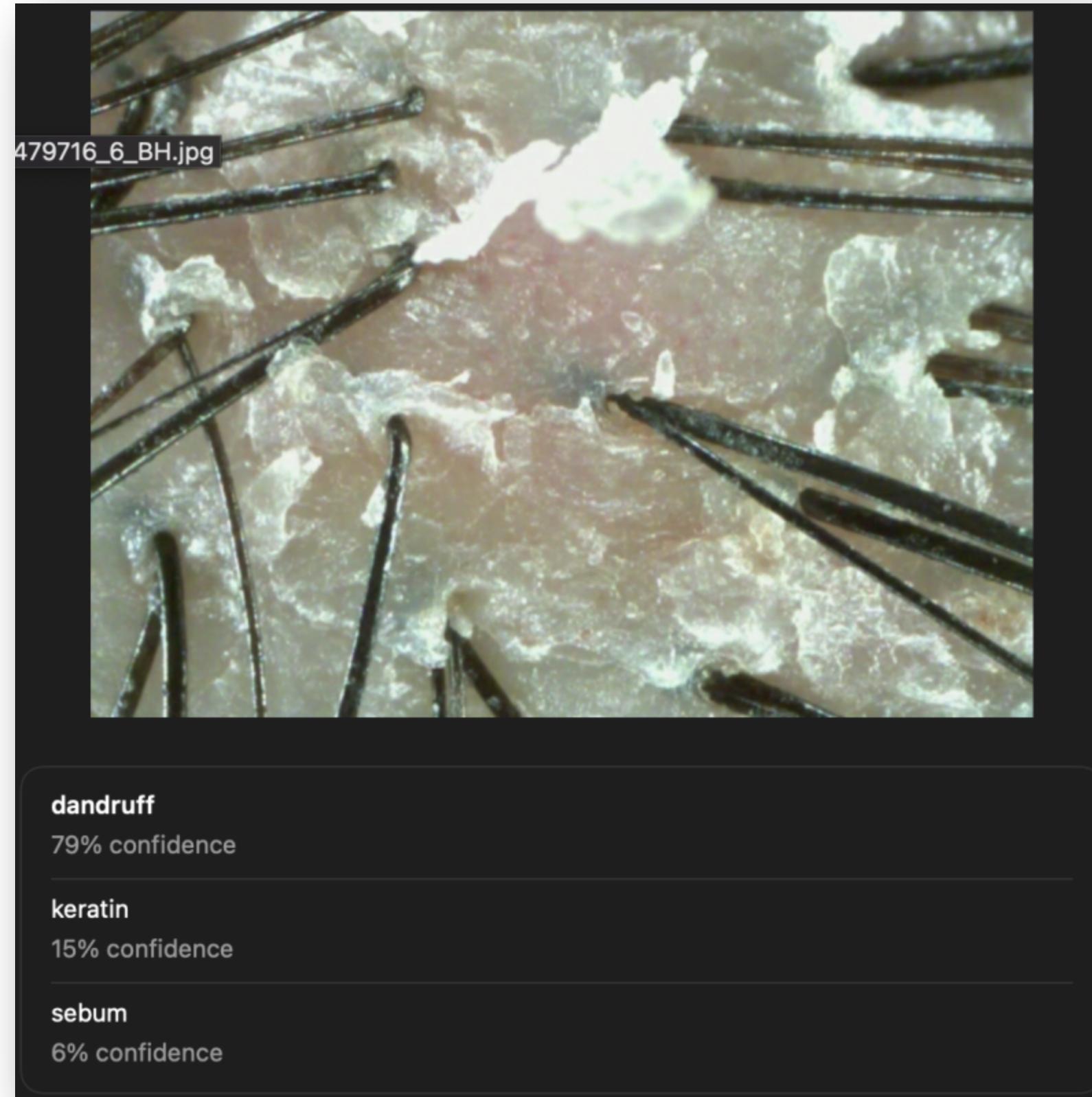
sebum

94% confidence

dandruff

6% confidence

# CreateML



Original Class label : dandruff

실제로는 dandruff 레이블을 가진 이미지이다.  
해당 두피 상태의 경우,  
비듬과 각질, 동시에 피지를  
가지고 있기 때문에

dandruff, keratin, sebum 레이블이  
결과로 나오는 것을 확인할 수 있다.

# 결과분석

## Result

위의 예시를 통해 두피 상태를 단순하게 하나의 상태로 분류해내는 것은 제 성능을 발휘하기 힘들다는 것을 확인했다. 그 이유는 사람의 두피상태는 정확히 하나의 상태로 정의하기 어렵다는 데에 있다고 생각한다. 예를 들어 탈모가 있는데 두피가 건성인 사람이 존재하는 것과 같다. 이러한 경우가 존재하기에 모델은 제 성능을 보이기 힘들다 생각한다.

이러한 점을 개선하기 위해 마지막 단에서 분류를 하는 것이 아니라 분류 전의 confidence를 활용해 솔루션을 제공할 수 있을 것이라 판단했다.

# 발전 방향

## Improvements

1. 더 효율적인 규제 기법을 찾아 적용한다.
2. 더 성능 좋은 GPU를 활용해 여러 실험을 진행해보며 프로젝트의 방향에 대한 고민이 필요하다.
3. confidence 활용을 통한 솔루션 제공의 알고리즘에 대한 고민이 필요하다.
4. 메타 데이터를 활용한 데이터 분석을 추가적으로 수행해 두피 상태에 대한 솔루션 제공에 활용한다.
5. Pretrained model 활용 시, 사전 학습된 모델의 학습에 사용된 데이터와 현재 학습을 위해 사용하는 데이터의 차이가 상당하다 생각한다. 또한 가지고 있는 두피 데이터의 양이 적기 때문에 학습 시 모든 layer를 unfreeze 함과 동시에 최적의 파라미터를 찾아내는 과정을 반복해 모델의 성능을 개선해야 한다.

# 느낀 점

## retrospect

- 계속해서 공부하고 실험해 관련 경험을 쌓은 것이 정말 중요하다는 생각을 했다.
- 프로젝트의 진행에 있어 전문적인 도메인 지식이 꼭 필요함을 느꼈다. 코드를 통해 두피의 상태 분류가 일어나지만 그 분류가 정확한 것인지 수치 상으로만 확인이 가능할 뿐 전문적 지식을 통한 확인은 불가능했기 때문이다.
- 이를 통해 여러 경험과 지식을 가진 동료들과 협업하는 것의 중요성 또한 느낄 수 있었다.

감사합니다.