

| 피부암 예측 및 분류 모델 |

상명대학교
스마트정보통신공학과

201921035 김건동

202121117 심종혜

202121146 최민석

목 차

1. 프로젝트의 내용
2. 데이터 가공 과정
3. 사용한 신경망 구조
4. 학습 과정과 결과
5. 앱 UI
6. 데이터 출처

1. 프로젝트 내용

배경 및 목적

- 2020년 247,952건 전체 암 발생. **피부암 2.9%(7,089건)** 차지
- 점과 혼동하는 경우 많음 => 조기발견이 어려움
- 피부암 환자 비율 중 70대 이상이 61% 차지
- melanoma -> 다른 부위로 암 전이 빠름
- -> 조기발견을 위해, 딥러닝을 활용한 피부암 예측 모델 제작

(출처: 중앙암등록본부, 국가암정보센터)

모델

CNN

AutoEncoder(CNN)

GAN

PCA

MLP

2. 데이터 가공 과정

피부암 종류 (6가지) + 피부암 아닌 피부질환 (1가지)

- Train data size : 5,915
 - Test data size : 516
-

1. CNN

```
# 이미지 로드
target_size = (180, 180)
train_images = []
test_images = []

for category in image_categories:
    category_train_dir = os.path.join(train_dir, category)
    category_test_dir = os.path.join(test_dir, category)
    train_images.extend(load_images(category_train_dir, target_size))
    test_images.extend(load_images(category_test_dir, target_size))

train_images = np.array(train_images)
test_images = np.array(test_images)

# 이미지 정규화 (0과 1 사이로 스케일 조정)
train_images = train_images.astype('float32') / 255.0
test_images = test_images.astype('float32') / 255.0
```

2. 데이터 가공 과정

피부암 종류 (6가지) + 피부암 아닌 피부질환 (1가지)

- 2. AutoEncoder(CNN)

```
# 이미지 로드
target_size = (224, 224)
train_images = []
test_images = []

for category in image_categories:
    category_train_dir = os.path.join(train_dir, category)
    category_test_dir = os.path.join(test_dir, category)

    train_images_category = load_images(category_train_dir, target_size)
    test_images_category = load_images(category_test_dir, target_size)

    train_images.extend(train_images_category)
    test_images.extend(test_images_category)

train_images = np.array(train_images)
test_images = np.array(test_images)

# 이미지 정규화 (0과 1 사이로 스케일 조정)
train_images = train_images.astype('float32') / 255.0
test_images = test_images.astype('float32') / 255.0
```

```
# 이미지 분류를 위한 레이블 생성
train_labels = []
test_labels = []

for i, category in enumerate(image_categories):
    category_train_dir = os.path.join(train_dir, category)
    category_test_dir = os.path.join(test_dir, category)

    train_images_category = load_images(category_train_dir, target_size)
    test_images_category = load_images(category_test_dir, target_size)

    train_labels.extend([i] * len(train_images_category))
    test_labels.extend([i] * len(test_images_category))

train_labels = np.array(train_labels)
test_labels = np.array(test_labels)
```

2. 데이터 가공 과정

피부암 종류 (6가지) + 피부암 아닌 피부질환 (1가지)

- 3. GAN

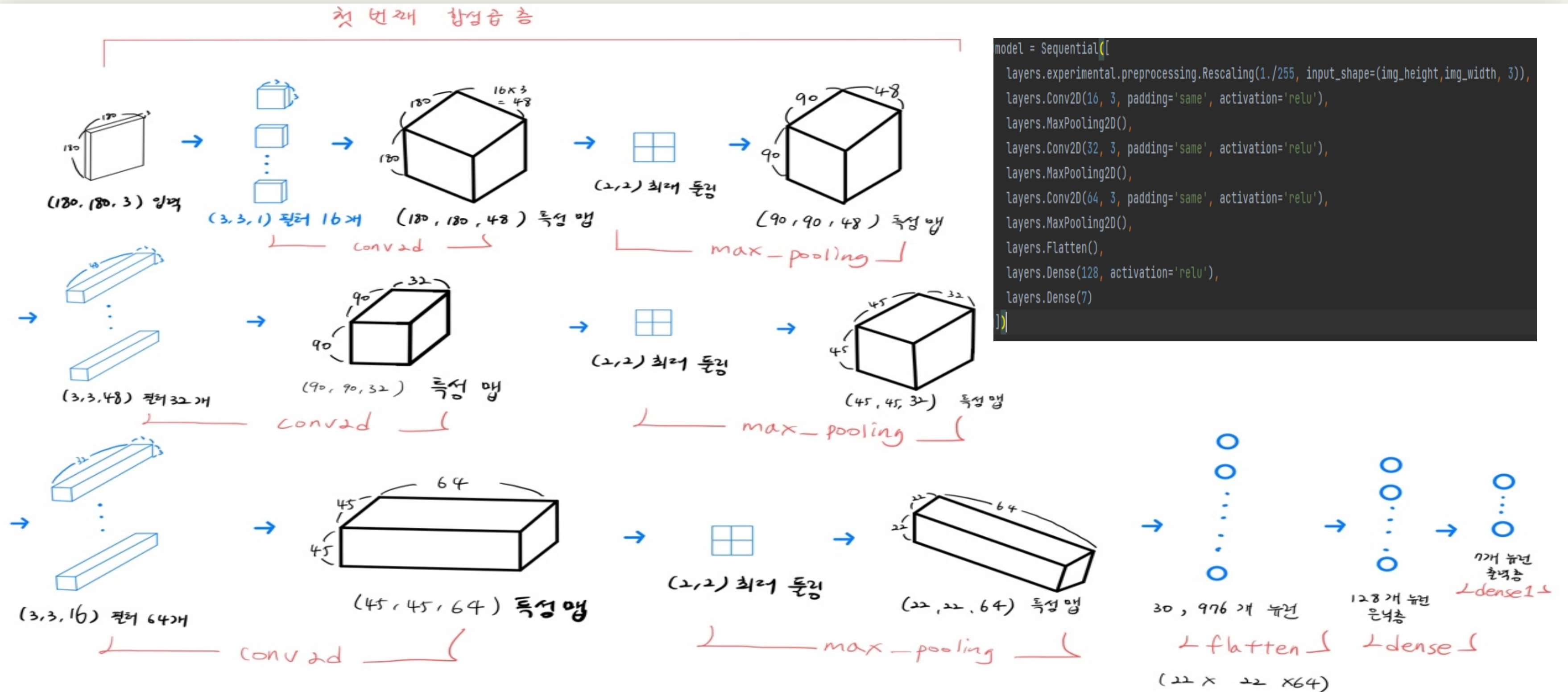
```
# 데이터 로드 및 전처리
image_paths = glob.glob(os.path.join(data_dir, "*", "*.jpg"))
num_images = len(image_paths)
batch_size = 64

if num_images == 0:
    print("No images found in the dataset.")
    return

def load_image(image_path):
    img = load_img(image_path, target_size=(28, 28), color_mode='rgb')
    img = img_to_array(img) / 255.0 # 이미지를 [0, 1] 범위로 정규화
    return img

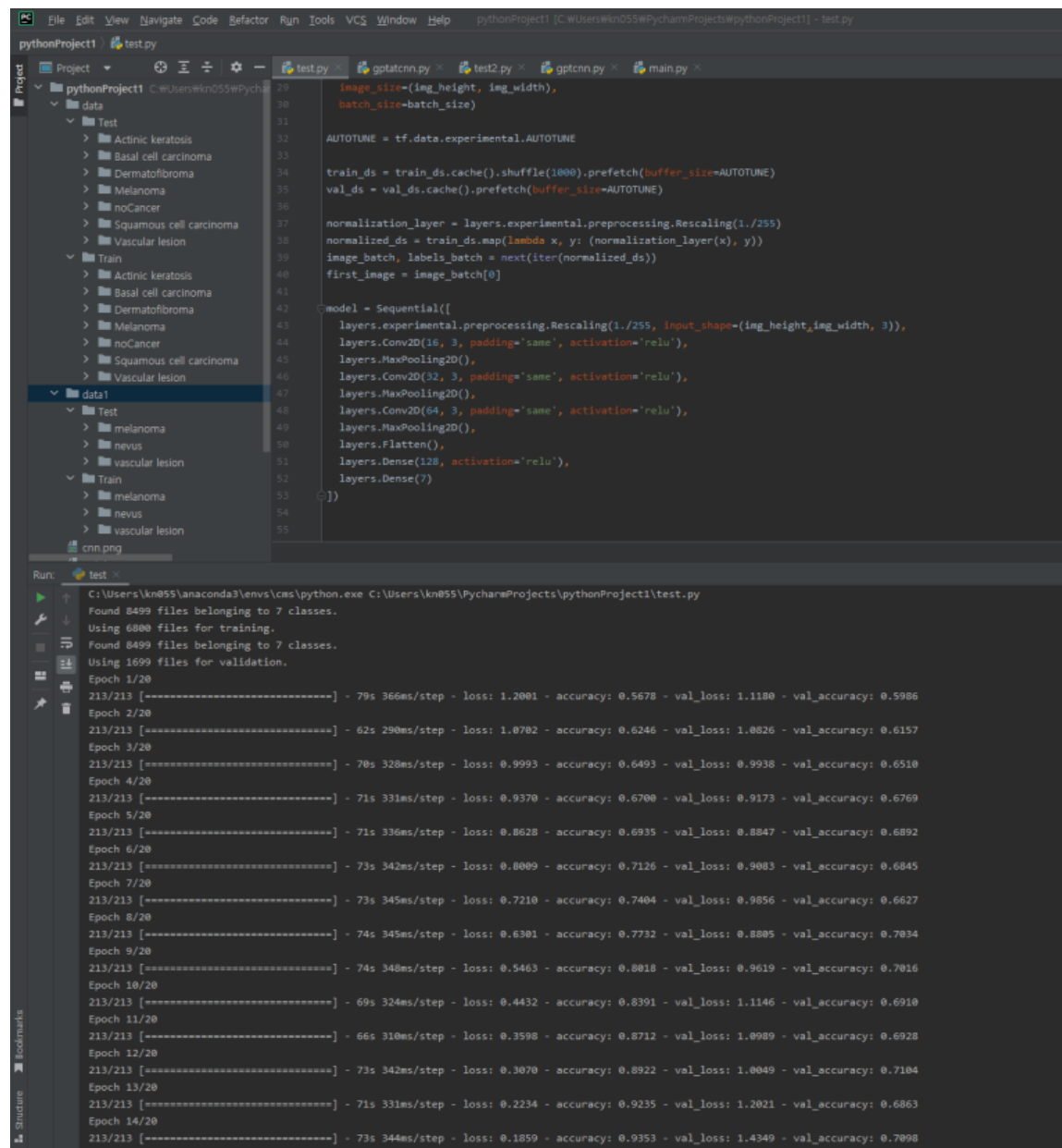
dataset = tf.data.Dataset.from_tensor_slices(image_paths)
dataset = dataset.shuffle(buffer_size=num_images)
dataset = dataset.map(lambda x: tf.numpy_function(load_image, [x], tf.float32),
                    num_parallel_calls=tf.data.experimental.AUTOTUNE)
dataset = dataset.batch(batch_size, drop_remainder=True)
dataset = dataset.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
```

3. 사용한 신경망 구조 - CNN



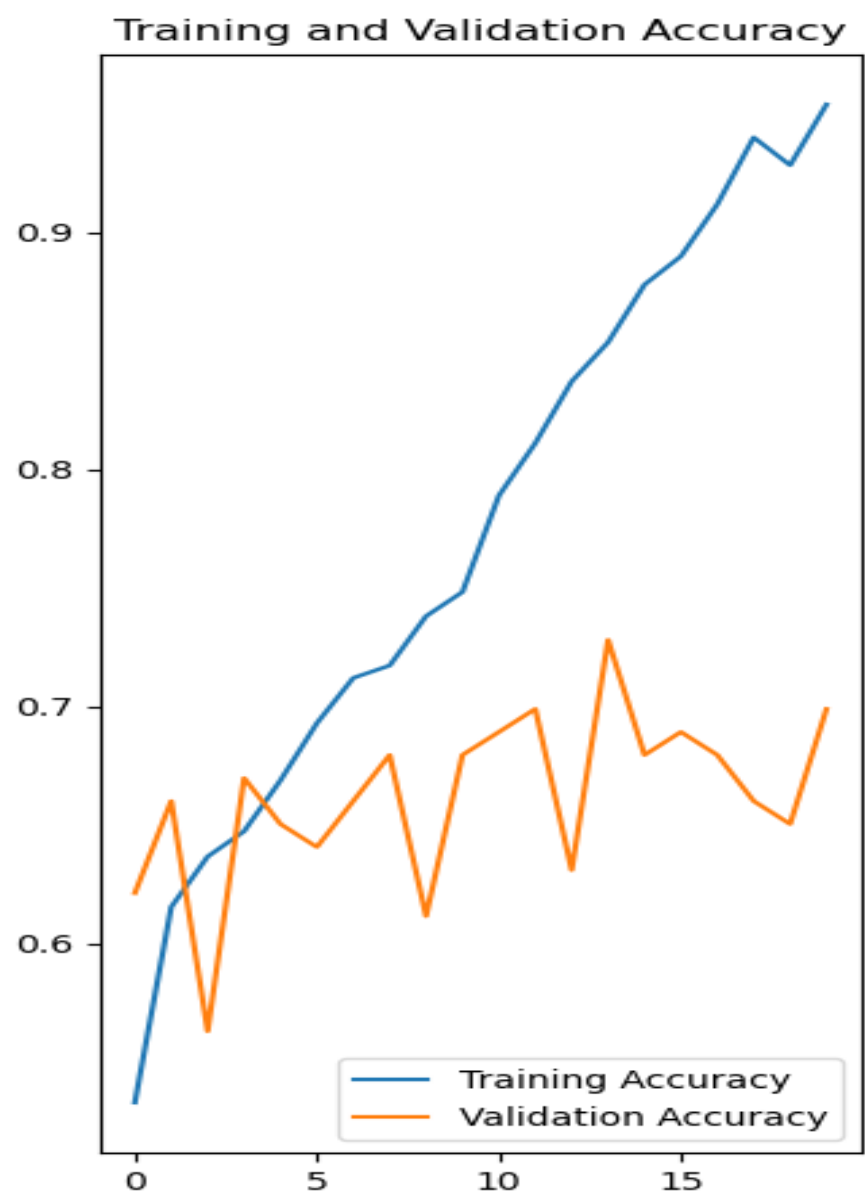
4. 학습 과정과 결과 - CNN

- 1. CNN

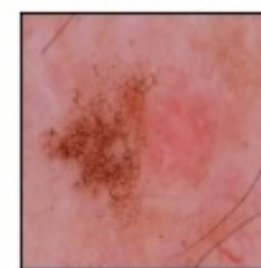


The screenshot shows a Jupyter Notebook with a file explorer on the left, a code editor in the center, and a console output at the bottom. The file explorer shows a project structure with 'data' and 'train' folders. The code editor contains Python code for a CNN model using TensorFlow and Keras. The console output shows the training process, including the number of files found, the number of files used for training and validation, and the training progress over 20 epochs.

```
pythonProject1 C:\Users\kn055\PycharmProjects\pythonProject1 - test.py
pythonProject1 test.py
Project
  pythonProject1 C:\Users\kn055\PycharmProjects\pythonProject1
  data
    Test
      Actinic keratosis
      Basal cell carcinoma
      Dermatofibroma
      Melanoma
      noCancer
      Squamous cell carcinoma
      Vascular lesion
    Train
      Actinic keratosis
      Basal cell carcinoma
      Dermatofibroma
      Melanoma
      noCancer
      Squamous cell carcinoma
      Vascular lesion
  data1
    Test
      melanoma
      nevus
      vascular lesion
    Train
      melanoma
      nevus
      vascular lesion
  cnn.png
Run: test
C:\Users\kn055\anaconda3\envs\csc\python.exe C:\Users\kn055\PycharmProjects\pythonProject1\test.py
Found 8499 files belonging to 7 classes.
Using 6800 files for training.
Found 8499 files belonging to 7 classes.
Using 1699 files for validation.
Epoch 1/20
213/213 [-----] - 79s 366ms/step - loss: 1.2001 - accuracy: 0.5678 - val_loss: 1.1180 - val_accuracy: 0.5986
Epoch 2/20
213/213 [-----] - 62s 290ms/step - loss: 1.0782 - accuracy: 0.6246 - val_loss: 1.0826 - val_accuracy: 0.6157
Epoch 3/20
213/213 [-----] - 70s 328ms/step - loss: 0.9993 - accuracy: 0.6493 - val_loss: 0.9938 - val_accuracy: 0.6510
Epoch 4/20
213/213 [-----] - 71s 331ms/step - loss: 0.9370 - accuracy: 0.6700 - val_loss: 0.9173 - val_accuracy: 0.6769
Epoch 5/20
213/213 [-----] - 71s 336ms/step - loss: 0.8628 - accuracy: 0.6935 - val_loss: 0.8847 - val_accuracy: 0.6892
Epoch 6/20
213/213 [-----] - 73s 342ms/step - loss: 0.8009 - accuracy: 0.7126 - val_loss: 0.9083 - val_accuracy: 0.6845
Epoch 7/20
213/213 [-----] - 73s 345ms/step - loss: 0.7210 - accuracy: 0.7404 - val_loss: 0.9856 - val_accuracy: 0.6627
Epoch 8/20
213/213 [-----] - 74s 345ms/step - loss: 0.6301 - accuracy: 0.7732 - val_loss: 0.8805 - val_accuracy: 0.7034
Epoch 9/20
213/213 [-----] - 74s 348ms/step - loss: 0.5463 - accuracy: 0.8018 - val_loss: 0.9619 - val_accuracy: 0.7016
Epoch 10/20
213/213 [-----] - 69s 324ms/step - loss: 0.4432 - accuracy: 0.8391 - val_loss: 1.1146 - val_accuracy: 0.6910
Epoch 11/20
213/213 [-----] - 66s 310ms/step - loss: 0.3598 - accuracy: 0.8712 - val_loss: 1.0909 - val_accuracy: 0.6928
Epoch 12/20
213/213 [-----] - 73s 342ms/step - loss: 0.3070 - accuracy: 0.8922 - val_loss: 1.0049 - val_accuracy: 0.7104
Epoch 13/20
213/213 [-----] - 71s 331ms/step - loss: 0.2234 - accuracy: 0.9235 - val_loss: 1.2021 - val_accuracy: 0.6863
Epoch 14/20
213/213 [-----] - 73s 344ms/step - loss: 0.1859 - accuracy: 0.9353 - val_loss: 1.4340 - val_accuracy: 0.7098
```



Melanoma



Dermatofibroma Basal cell carcinoma Actinic keratosis I



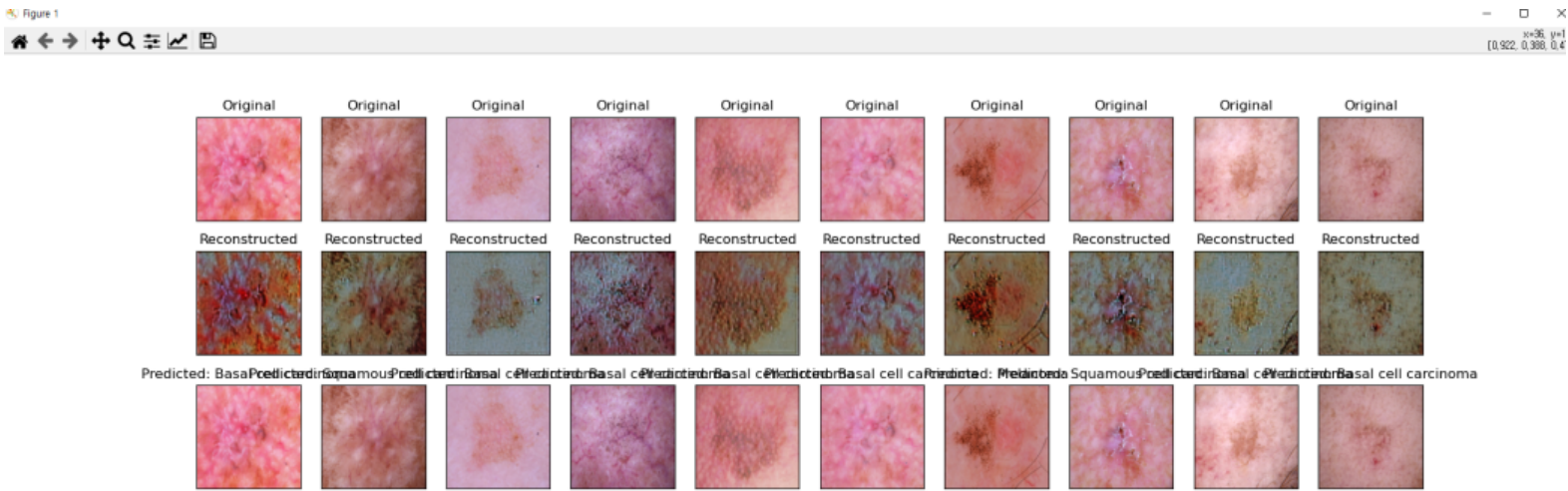
20 epochs 정확도 75%

4. 학습 과정과 결과 - AutoEncoder(CNN)

- 2. AutoEncoder(CNN)

```
pythonProject5 - autoencoder7.py
Project
  data3
    test
      Actinic keratosis
      Basal cell carcinoma
      Dermatofibroma
      Melanoma
      noCancer
      Squamous cell carcinoma
      Vascular lesion
    train
      Actinic keratosis
      Basal cell carcinoma
      Dermatofibroma
      Melanoma
      noCancer
      Squamous cell carcinoma
      Vascular lesion
    data4
    temp
    val
      autoEncoder1.py
      autoEncoder2.py
      autoEncoder3.py
      autoEncoder4.py
      autoEncoder5.py
      autoEncoder6.py
      autoencoder7.py
      autoEncoder_test.py
      data1.zip
      data3.zip
      data4.zip
      learning.py
  autoencoder7.py
    110 batch_size=64,
    111 shuffle=True,
    112 validation_data=(test_images, test_labels),
    113 # callbacks=[early_stopping]
    114 )
    115
    116 # 재구성된 이미지 예측
    117 reconstructed_images = autoencoder.predict(test_images)
    118
    119 # 이미지 시각화
    120 n = 10 # 시각화할 이미지 개수
    121 plt.figure(figsize=(20, 6))
    122 for i in range(n):
    123     # 원본 이미지
    124     ax = plt.subplot(3, n, i + 1)
    125     plt.imshow(test_images[i])
    126     plt.title("Original")
    127     plt.gray()
    128     ax.get_xaxis().set_visible(False)
    129     ax.get_yaxis().set_visible(False)
    130
    131     # 재구성된 이미지
    132     ax = plt.subplot(3, n, i + 1 + n)
    133     plt.imshow(reconstructed_images[i])
    134     plt.title("Reconstructed")
    135     plt.gray()
    136     ax.get_yaxis().set_visible(False)
    137
    138 for i in range(n)

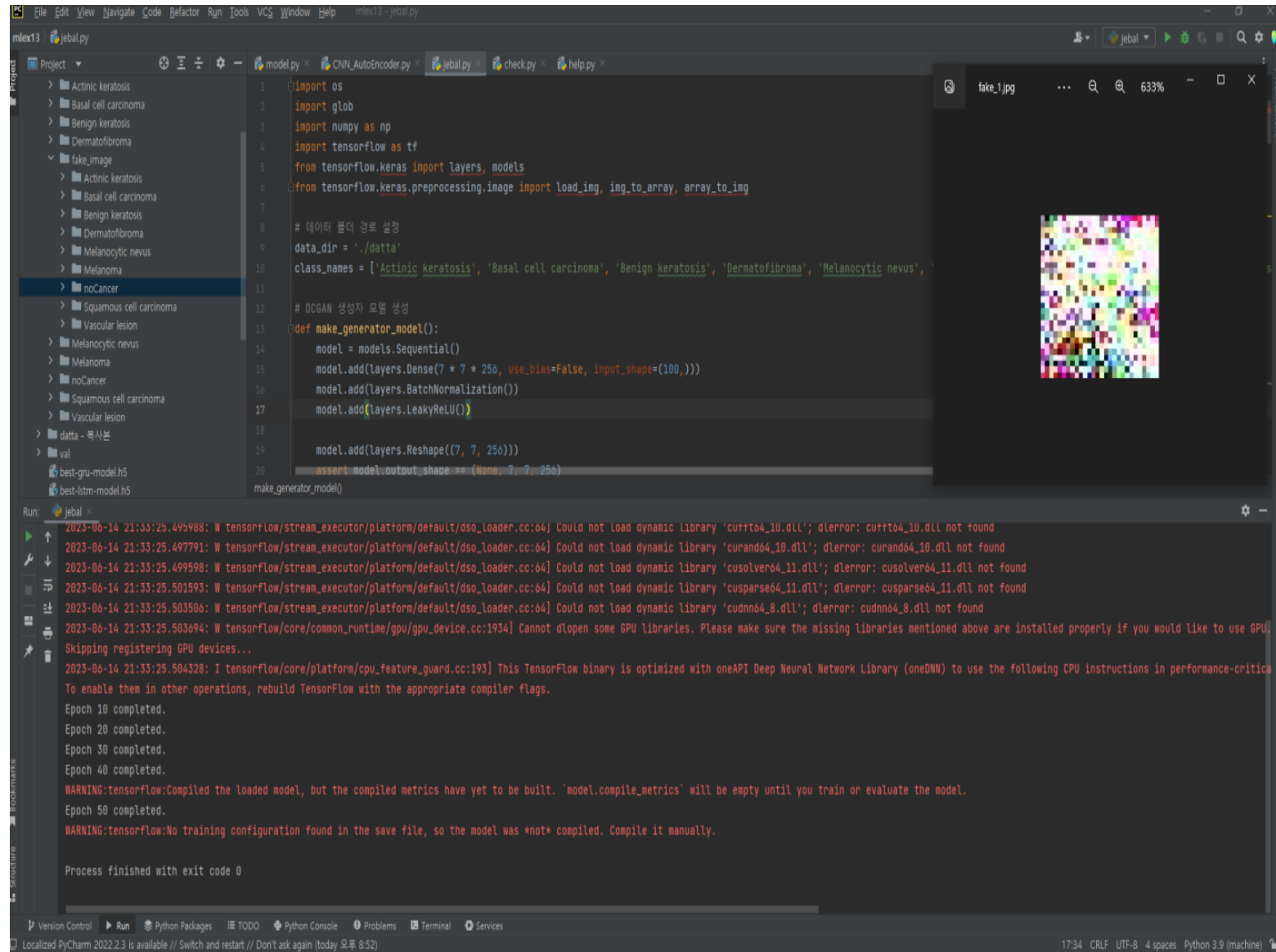
Run:
autoencoder7
90/90 [=====] - 38s 420ms/step - loss: 0.1639 - accuracy: 0.9529 - val_loss: 1.4575 - val_accuracy: 0.6764
Epoch 17/20
90/90 [=====] - 38s 420ms/step - loss: 0.1487 - accuracy: 0.9584 - val_loss: 1.4720 - val_accuracy: 0.6802
Epoch 18/20
90/90 [=====] - 39s 428ms/step - loss: 0.1089 - accuracy: 0.9736 - val_loss: 1.7175 - val_accuracy: 0.6647
Epoch 19/20
90/90 [=====] - 38s 427ms/step - loss: 0.0810 - accuracy: 0.9809 - val_loss: 1.7412 - val_accuracy: 0.6744
Epoch 20/20
90/90 [=====] - 38s 424ms/step - loss: 0.0710 - accuracy: 0.9841 - val_loss: 1.9097 - val_accuracy: 0.6647
17/17 [=====] - 2s 120ms/step
1/1 [=====] - 0s 50ms/step
```



20 epochs 정확도: 68%

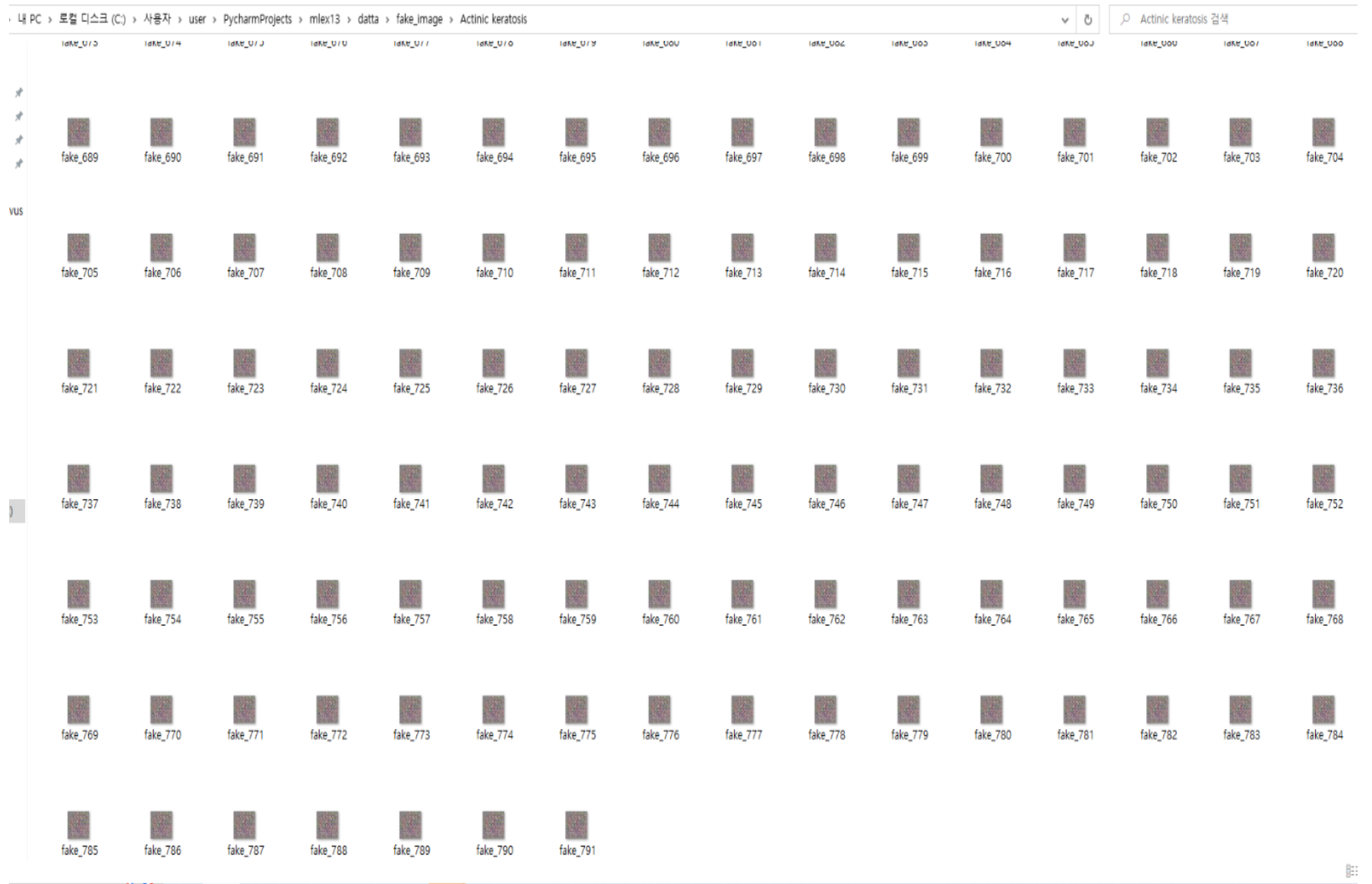
4. 학습 과정과 결과 - GAN

• 3. GAN



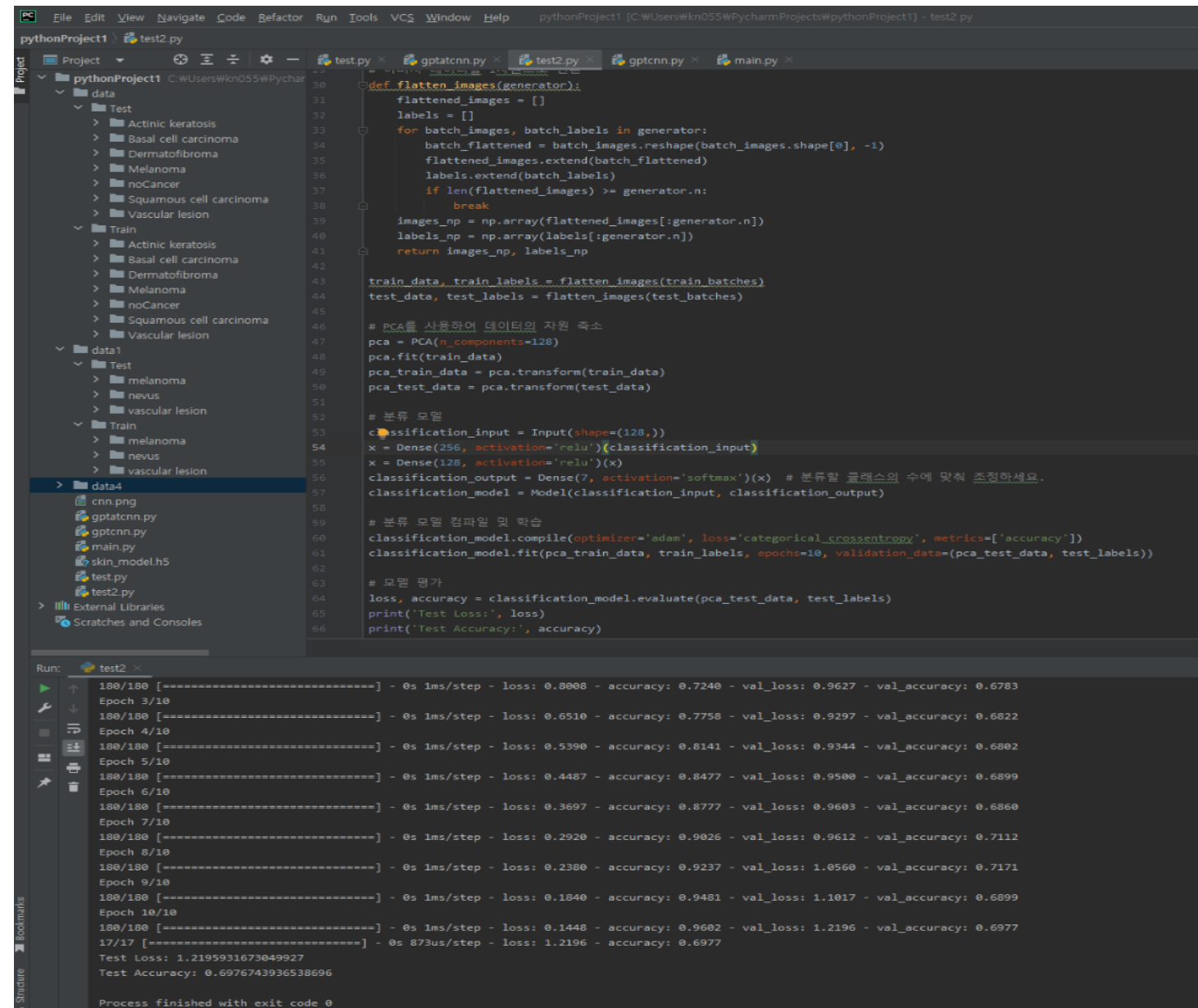
```
1 import os
2 import glob
3 import numpy as np
4 import tensorflow as tf
5 from tensorflow.keras import layers, models
6 from tensorflow.keras.preprocessing.image import load_img, img_to_array, array_to_img
7
8 # 데이터 폴더 설정
9 data_dir = './data'
10 class_names = ['Actinic keratosis', 'Basal cell carcinoma', 'Benign keratosis', 'Dermatofibroma', 'Melanocytic nevus',
11               'noCancer', 'Squamous cell carcinoma', 'Vascular lesion']
12 # DCGAN 생성자 모델 생성
13 def make_generator_model():
14     model = models.Sequential()
15     model.add(layers.Dense(7 * 7 * 256, use_bias=False, input_shape=(100,)))
16     model.add(layers.BatchNormalization())
17     model.add(layers.LeakyReLU())
18
19     model.add(layers.Reshape((7, 7, 256)))
20     assert model.output_shape == (None, 7, 7, 256)
21     return model
```

Run: 2023-06-14 21:33:25.495988: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudrt64_10.dll'; dlerror: cudrt64_10.dll not found
2023-06-14 21:33:25.497791: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'curand64_10.dll'; dlerror: curand64_10.dll not found
2023-06-14 21:33:25.499598: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cusolver64_11.dll'; dlerror: cusolver64_11.dll not found
2023-06-14 21:33:25.501593: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cusparses64_11.dll'; dlerror: cusparses64_11.dll not found
2023-06-14 21:33:25.503506: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudnn64_8.dll'; dlerror: cudnn64_8.dll not found
2023-06-14 21:33:25.503694: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1934] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are installed properly if you would like to use GPU.
Skipping registering GPU devices...
2023-06-14 21:33:25.504328: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations.
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 10 completed.
Epoch 20 completed.
Epoch 30 completed.
Epoch 40 completed.
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be empty until you train or evaluate the model.
Epoch 50 completed.
WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compile it manually.
Process finished with exit code 0



4. 학습 과정과 결과 - PCA

- 4. PCA



The screenshot shows a Python IDE with a project named 'pythonProject1'. The file explorer on the left shows a directory structure with 'data' and 'data1' folders. The main editor displays a Python script 'test2.py' implementing PCA and a simple neural network for skin lesion classification. The script includes functions for flattening images, training the model, and evaluating it. The Run window at the bottom shows the output of the training process over 10 epochs.

```
def flatten_images(generator):
    flattened_images = []
    labels = []
    for batch_images, batch_labels in generator:
        batch_flattened = batch_images.reshape(batch_images.shape[0], -1)
        flattened_images.extend(batch_flattened)
        labels.extend(batch_labels)
        if len(flattened_images) >= generator.n:
            break
    images_np = np.array(flattened_images[:generator.n])
    labels_np = np.array(labels[:generator.n])
    return images_np, labels_np

train_data, train_labels = flatten_images(train_batches)
test_data, test_labels = flatten_images(test_batches)

# PCA를 사용하여 데이터의 차원 축소
pca = PCA(n_components=128)
pca.fit(train_data)
pca_train_data = pca.transform(train_data)
pca_test_data = pca.transform(test_data)

# 분류 모델
classification_input = Input(shape=(128,))
x = Dense(256, activation='relu')(classification_input)
x = Dense(128, activation='relu')(x)
classification_output = Dense(7, activation='softmax')(x) # 분류할 클래스의 수에 맞춰 조정하세요.
classification_model = Model(classification_input, classification_output)

# 분류 모델 컴파일 및 학습
classification_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
classification_model.fit(pca_train_data, train_labels, epochs=10, validation_data=(pca_test_data, test_labels))

# 모델 평가
loss, accuracy = classification_model.evaluate(pca_test_data, test_labels)
print('Test Loss:', loss)
print('Test Accuracy:', accuracy)
```

Run: test2

Epoch	loss	accuracy	val_loss	val_accuracy
180/180	0.8008	0.7240	0.9627	0.6783
Epoch 3/10	0.6510	0.7758	0.9297	0.6822
Epoch 4/10	0.5390	0.8141	0.9344	0.6802
Epoch 5/10	0.4487	0.8477	0.9500	0.6899
Epoch 6/10	0.3697	0.8777	0.9603	0.6860
Epoch 7/10	0.2920	0.9026	0.9612	0.7112
Epoch 8/10	0.2380	0.9237	1.0560	0.7171
Epoch 9/10	0.1840	0.9481	1.1017	0.6899
Epoch 10/10	0.1448	0.9602	1.2196	0.6977

Test Loss: 1.2195931673049927
Test Accuracy: 0.6976743936538696

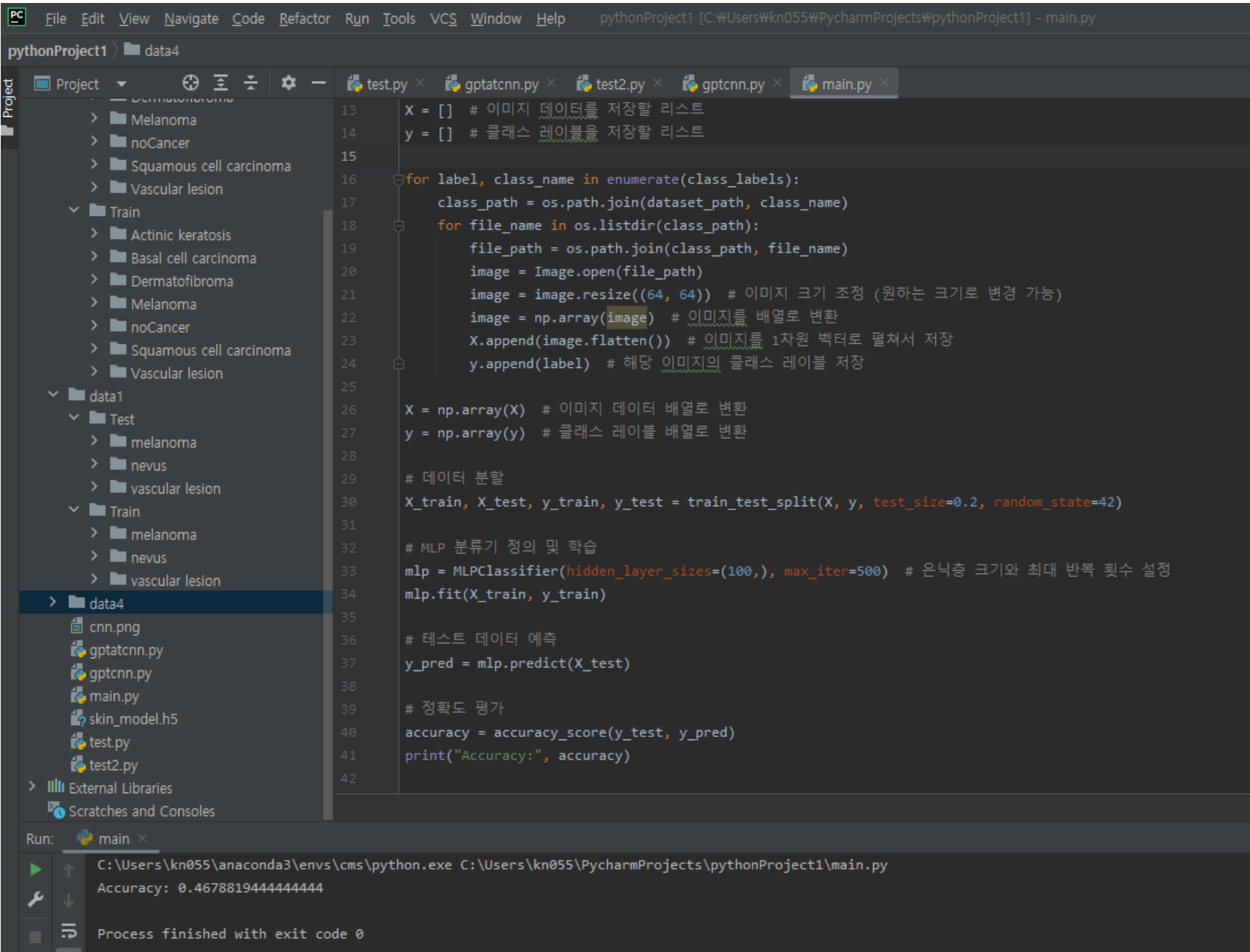
Process finished with exit code 0

Test Loss: 1.2195931673049927
Test Accuracy: 0.6976743936538696

10 epochs 정확도: 69%

4. 학습 과정과 결과 - MLP

- 5. MLP



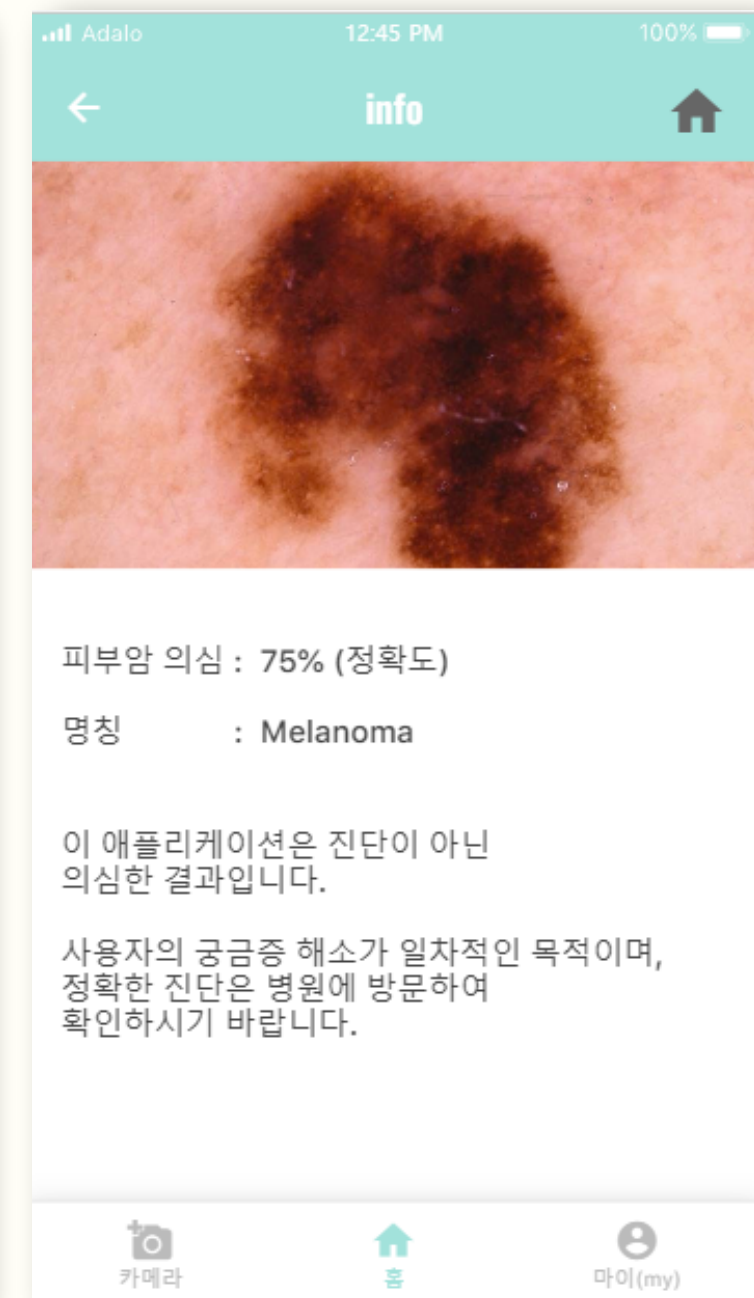
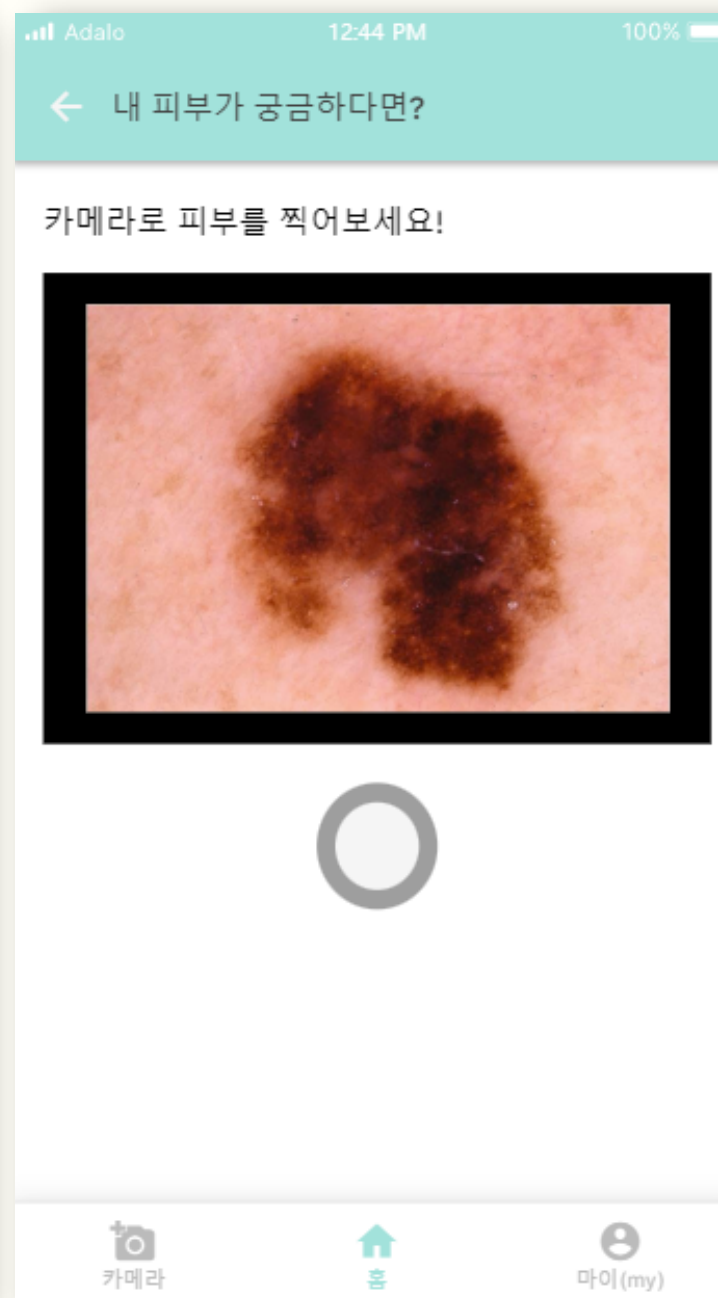
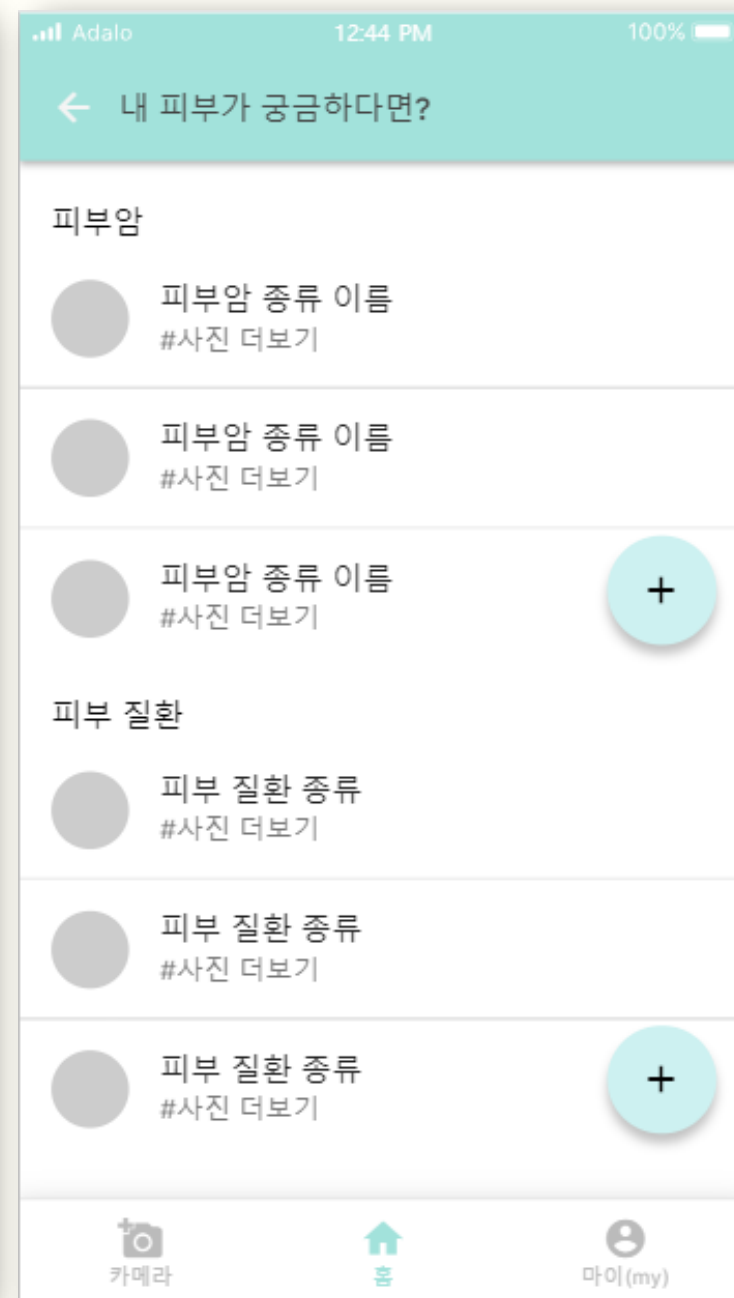
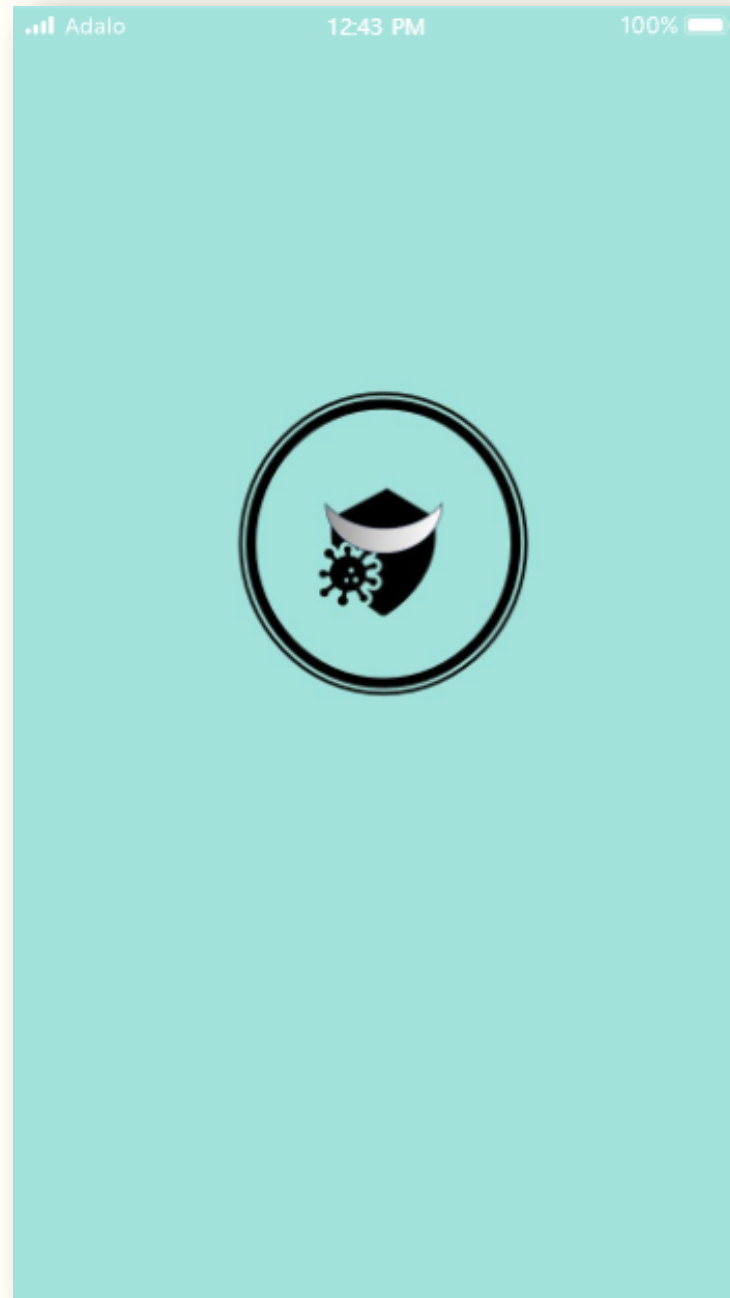
```
pythonProject1 [C:\Users\kn055\PycharmProjects\pythonProject1] - main.py
pythonProject1 > data4
Project
  - pythonProject1
    > data4
      > data1
        > Test
          > melanoma
          > nevus
          > vascular lesion
        > Train
          > melanoma
          > nevus
          > vascular lesion
      > data4
        cnn.png
        gptatcnn.py
        gptcnn.py
        main.py
        skin_model.h5
        test.py
        test2.py
      > External Libraries
      > Scratches and Consoles
Run: main
C:\Users\kn055\anaconda3\envs\cms\python.exe C:\Users\kn055\PycharmProjects\pythonProject1\main.py
Accuracy: 0.4678819444444444
Process finished with exit code 0
```

```
13 X = [] # 이미지 데이터를 저장할 리스트
14 y = [] # 클래스 레이블을 저장할 리스트
15
16 for label, class_name in enumerate(class_labels):
17     class_path = os.path.join(dataset_path, class_name)
18     for file_name in os.listdir(class_path):
19         file_path = os.path.join(class_path, file_name)
20         image = Image.open(file_path)
21         image = image.resize((64, 64)) # 이미지 크기 조정 (원하는 크기로 변경 가능)
22         image = np.array(image) # 이미지를 배열로 변환
23         X.append(image.flatten()) # 이미지를 1차원 벡터로 펼쳐서 저장
24         y.append(label) # 해당 이미지의 클래스 레이블 저장
25
26 X = np.array(X) # 이미지 데이터 배열로 변환
27 y = np.array(y) # 클래스 레이블 배열로 변환
28
29 # 데이터 분할
30 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
31
32 # MLP 분류기 정의 및 학습
33 mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500) # 은닉층 크기와 최대 반복 횟수 설정
34 mlp.fit(X_train, y_train)
35
36 # 테스트 데이터 예측
37 y_pred = mlp.predict(X_test)
38
39 # 정확도 평가
40 accuracy = accuracy_score(y_test, y_pred)
41 print("Accuracy:", accuracy)
42
```

Accuracy: 0.4678819444444444

정확도:46%

5. 앱 UI



6. 사용한 데이터의 출처(인터넷 주소 포함)

kaggle: skin-cancer-isic2019,

<https://www.kaggle.com/datasets/shobujchandradas/skin-cancer-isic2019>