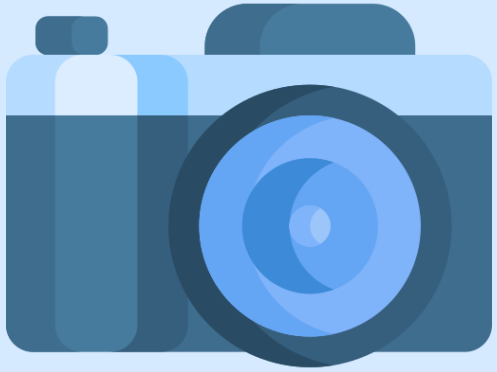
2020년 2학기
졸업프로젝트2



인공지능을 이용한 인물 사진 분류 어플리케이션

PhotoPick

T-12
이다솔 손혜림 한지은 임세빈
지도교수 박 능 수 교수님

CONTENTS

작품 소개

작품 개요

인물 얼굴 인식 1

자동 분류 2

새로운 사진 태그 3

한눈에 사진 보기 4

소중한 사람들과의 추억들을 더욱 소중히

현대 사회에서 우리는 스마트폰으로 많은 사람들과 사진을 찍습니다. 그러나 너무 많은 사진들을 한 공간에 몰아넣으면 우리는 금방 잊어버리게 될 거예요.
PhotoPick은 머신러닝 알고리즘을 이용하여 인물의 얼굴을 인식하고, 또 사진에 출연한 인물의 수만큼 세세하게 폴더를 나누어서 특정 인물의 사진을 찾을 때에 유용합니다!
가족들, 친한 친구들과의 사진을 직관적인 디자인의 PhotoPick을 통해 언제든지 꺼내 볼 수 있게 하세요.

작품 개발 과정


```
import sys
import cv2

face_cascade = cv2.CascadeClassifier(r'(

for i in range(310):
    try:
        image = cv2.imread('                              '+str(i)+'.jpg')
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        face = face_cascade.detectMultiScale(gray, 1.03, 5)

        for(x,y,w,h) in face:
            cropped = image[y - int(h / 4):y + h + int(h / 4), x - int(w / 4):x + w + int(w / 4)]
            cv2.imwrite('(                              '+str(i)+".jpg",cropped)
    except:
        print(str(i)+'번 사진 스킵')
        continue
print('종료')
```

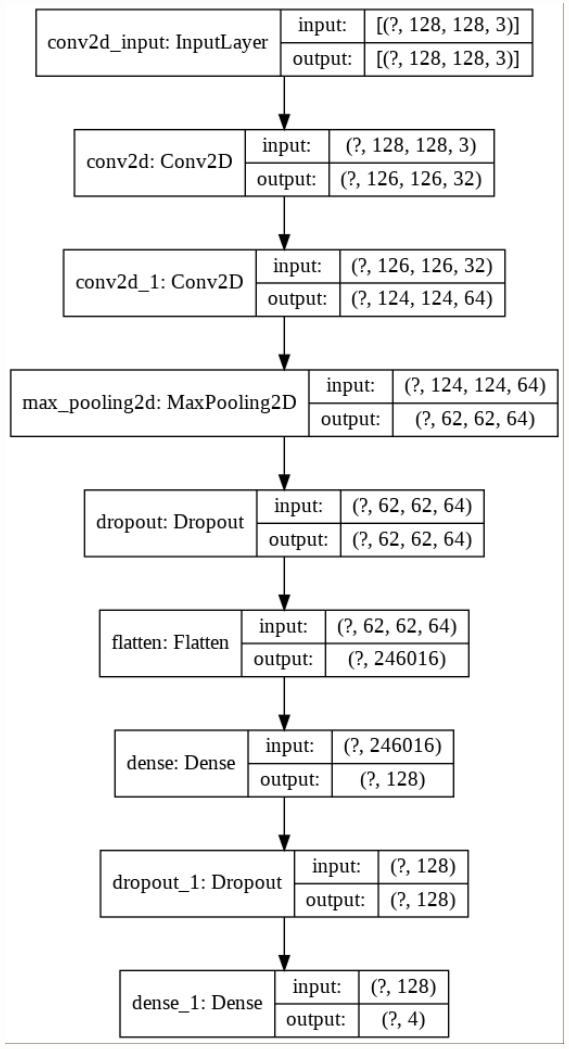
초기 모델을 학습시키기 위한 데이터를 크롤링합니다.

이 때 사용할 수 없는 데이터는 1차적으로 수동으로 필터링을 하고, 얼굴 사진들에 대해 opencv의 face detection을 이용하여 얼굴을 인식하고, 그레이 스케일 비트맵으로 변환하고, 일정한 크기로 사진을 크롭합니다.

이 때 구글링으로 얻을 수 있는 유효한 데이터는 인물 당 약 300장 내외로, 충분한 데이터셋을 만들어 주기 위하여 차후에 Tensorflow의 image data generator을 사용합니다.

작품 개발 과정

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 126, 126, 32)      896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 124, 124, 64)      18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 62, 62, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 62, 62, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 246016)            0
_________________________________________________________________
dense (Dense)                (None, 128)               31490176
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 516
=================================================================
Total params: 31,510,084
Trainable params: 31,510,084
Non-trainable params: 0
_________________________________________________________________
```



모델의 CNN 층을 생성합니다.

작품 개발 과정

```
# 오차함수로 categorical_crossentropy, 최적화로 adam 사용
model.compile(loss = 'categorical_crossentropy',
              optimizer = 'adam',
              metrics = ['accuracy'])

MODEL_DIR = 'model'
if not os.path.exists(MODEL_DIR):
    os.mkdir(MODEL_DIR)

modelpath = 'model/{epoch:02d}-{val_loss:.4f}.hdf5'
checkpointer = ModelCheckpoint(filepath = modelpath, monitor='val_loss', verbose = 1, save_best_only = True)
early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience=10)
```

```
#8
hist = model.fit(train_generator, steps_per_epoch=4, epochs = 500, validation_data = test_generator, validation_steps = 1)

from keras.models import load_model
model.save('model.h5')
```

```
Epoch 497/500
4/4 [==============================] - 4s 876ms/step - loss: 0.1828 - accuracy: 0.9416 - val_loss: 0.2405 - val_accuracy: 0.9266
Epoch 498/500
4/4 [==============================] - 3s 845ms/step - loss: 0.1399 - accuracy: 0.9459 - val_loss: 0.2550 - val_accuracy: 0.9209
Epoch 499/500
4/4 [==============================] - 4s 981ms/step - loss: 0.1604 - accuracy: 0.9416 - val_loss: 0.3522 - val_accuracy: 0.9040
Epoch 500/500
4/4 [==============================] - 4s 974ms/step - loss: 0.1520 - accuracy: 0.9459 - val_loss: 0.3149 - val_accuracy: 0.9040
```

```
#9
score = model.evaluate_generator(test_generator, steps = 5)
print('Test loss :', score[0])
print('Test accuracy : ', score[1])
```

```
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at le
Test loss : 0.2709798514842987
Test accuracy :  0.909604549407959
```

모델을 학습시킵니다.

SRS의 비기능 요구사항에서 정확도를 90% 이상으로 잡았기 때문에, epoch은 100, 200, … 시도하여 500번 학습시켰을 때 안정적으로 정확도 90% 이상이 나온다고 판단하여 epoch을 500으로 설정하고 학습합니다.

작품 개발 과정

```
public String classify(Bitmap bitmap){
    String result = null;

    float[][][][] input = new float[1][128][128][3];
    float[][] output = new float[1][4]; //아빠, 할아버지, 할머니, 엄마

        int batchNum = 0;

        //128*128*3
        for (int x = 0; x < 128; x++) {
            for (int y = 0; y < 128; y++) {
                int pixel = bitmap.getPixel(x, y);
                input[batchNum][x][y][0] = Color.red(pixel) / 1.0f;
                input[batchNum][x][y][1] = Color.green(pixel) / 1.0f;
                input[batchNum][x][y][2] = Color.blue(pixel) / 1.0f;
            }
        }

        Interpreter lite = getTfliteInterpreter((MainActivity)context,  modelPath: "converted_model.tflite");
        lite.run(input, output);
```

```
for(int i=0; i<4; i++) {
    if (output[0][i] * 100 > 90) {
        if (i == 0) {
            result = "person1";
        } else if (i == 1) {
            result = "person2";
        } else if (i == 2) {
            result = "person3";
        } else {
            result = "person4";
        }
    }
    else {
        result = "others";
    }
}
return result;
```

tflite 를 안드로이드 스튜디오에 연동시켜 모델을 run 하고, 얼굴을 인식하여 인물을 분류합니다.
일치도가 90% 미만일 경우 others로 분류합니다.

작품 개발 과정

```
Mat image = new Mat();
Mat resizingGray = new Mat();
MatOfRect faces = new MatOfRect();
Bitmap bitmap;
int count = 0;

for(String filepath : non_tag_images) {
    count++;
    String classified_tag = "";
    try {
            bitmap = Glide.with( activity: this) RequestManager
                    .asBitmap() RequestBuilder<Bitmap>
                    .load(filepath)
                    .submit() FutureTarget<Bitmap>
                    .get();

            Utils.bitmapToMat(bitmap, image);

        Imgproc.cvtColor(image,resizingGray,Imgproc.COLOR_BGR2GRAY);

        cascade.detectMultiScale(resizingGray,faces, scaleFactor: 1.3, minNeighbors: 3, flags: 0);
```

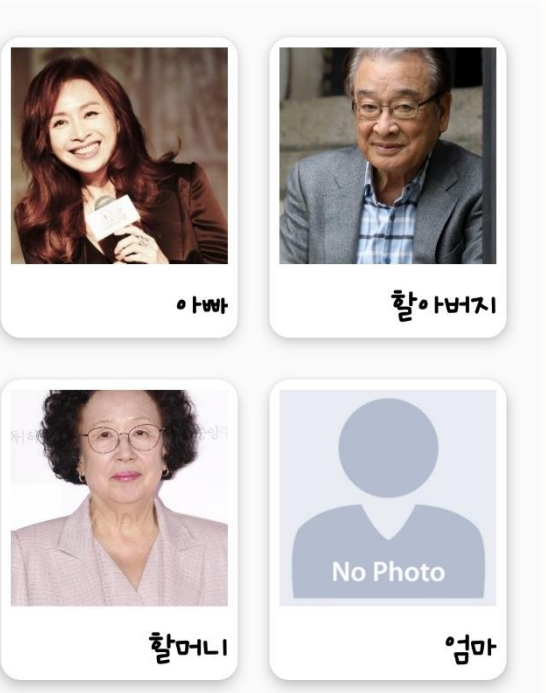
이미지에서 얼굴을 찾아 크롭하여 모델에 넣은 후, 결과를 토대로 태그를 만들어 저장합니다.

작품 개발 과정

```
public void imagePathClassification() {

    for(String filepath : pathOfAllImages) {
        try {
            ExifInterface exif = new ExifInterface(filepath);
            String tag_description = exif.getAttribute(ExifInterface.TAG_IMAGE_DESCRIPTION);
            if(tag_description != null) {
            String tag_split[] = tag_description.split( regex: "/

                if(tag_split.length == 1) { // 인물 1명일 경우
                    switch (tag_split[0]) {
                        case "person1":
                            person1.add(filepath);
                            person1_alone.add(filepath);
                            break;
                        case "person2":
                            person2.add(filepath);
                            person2_alone.add(filepath);
                            break;
                        case "person3":
                            person3.add(filepath);
                            person3_alone.add(filepath);
                            break;
```

```
for(int i=0;i<tag_split.length;i++) {
    switch (tag_split[i]) {
        case "person1":
            person1.add(filepath);
            person1_together.add(filepath);
            break;
        case "person2":
            person2.add(filepath);
            person2_together.add(filepath);
            break;
        case "person3":
            person3.add(filepath);
            person3_together.add(filepath);
            break;
        case "person4":
            person4.add(filepath);
            person4_together.add(filepath);
            break;
        case "others":
            others.add(filepath);
            break;
        default:
            break;
```

휴대폰에 저장된 이미지를 불러와 모든 이미지를 태그에 따라 인물별, 인원별로 경로를 분리하여 저장해줍니다.



작품 개발 과정

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    ((MainActivity) getActivity()).bottomNavigation.setVisibility(View.VISIBLE);

    // Inflate the layout for this fragment
    ViewGroup rootView = (ViewGroup)inflater.inflate(R.layout.fragment_person, container,  attachToRoot: false);

    alone_view_all_text = rootView.findViewById(R.id.alone_view_all_text);
    together_view_all_text = rootView.findViewById(R.id.together_view_all_text);

    recyclerView1 = rootView.findViewById(R.id.recyclerView_albumtab_1);
    recyclerView2 = rootView.findViewById(R.id.recyclerView_albumtab_2);

    TextView person_fragment_title_textView;
    person_fragment_title_textView = rootView.findViewById(R.id.person_fragment_title);
    person_fragment_title_textView.setText("아빠");

    LinearLayoutManager layoutManager1 = new LinearLayoutManager(getActivity(), LinearLayoutManager.HORIZONTAL,
    LinearLayoutManager layoutManager2 = new LinearLayoutManager(getActivity(), LinearLayoutManager.HORIZONTAL,
    recyclerView1.setLayoutManager(layoutManager1);
    recyclerView2.setLayoutManager(layoutManager2);

    adapter1 = new Image_hometab_Adapter(getActivity());
    adapter2 = new Image_hometab_Adapter(getActivity());
```

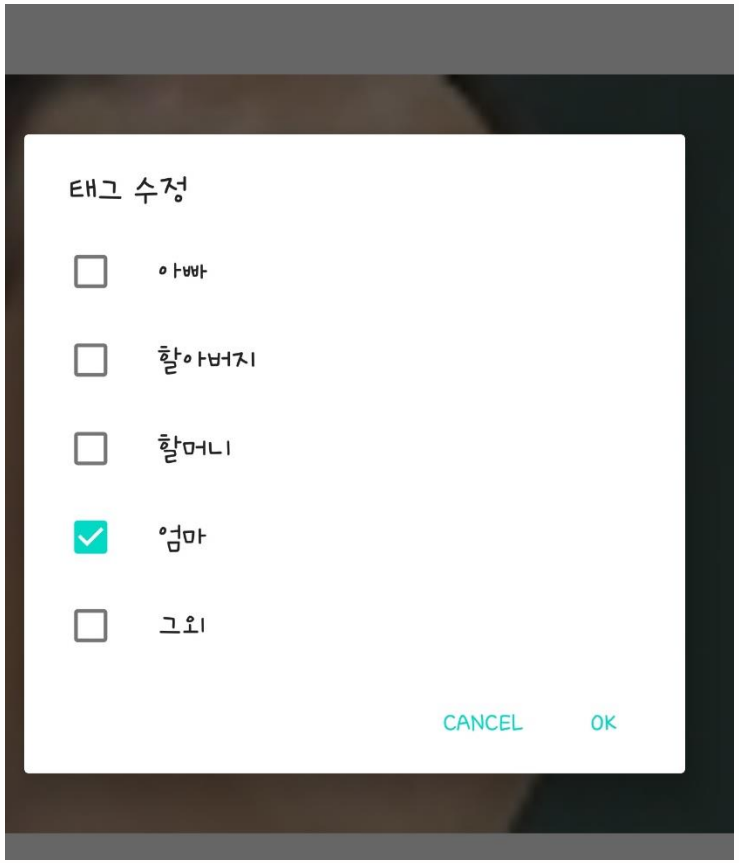
저장된 이미지는 각각의 뷰에서 리스트의 형태로 확인할 수 있습니다.

혼자

작품 개발 과정

```
ViewGroup rootView = (ViewGroup)inflater.inflate(R.layout.selected_image, container,  attachToRoot: false);

getActivity().getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_F

tag_mod_btn = (ImageButton) rootView.findViewById(R.id.tag_mod_button);
img = (PhotoView)rootView.findViewById(R.id.selected_image);

Glide.with(getActivity()).load(((MainActivity) getActivity()).selected_image_uri).into(img);

((MainActivity) getActivity()).bottomNavigation.setVisibility(View.GONE);
tag_mod_btn.setVisibility(View.INVISIBLE);

tag_mod_btn.setOnClickListener((view) → { tag_modification(); });

img.setOnClickListener((view) → {
        if(ui_flag == 1) {
            img.setBackgroundColor(Color.BLACK);
            tag_mod_btn.setVisibility(View.INVISIBLE);
            ui_flag = 0;
        }
        else {
            img.setBackgroundColor(Color.WHITE);
            tag_mod_btn.setVisibility(View.VISIBLE);
            ui_flag = 1;
```

이미지를 선택하여 볼 수 있고, Digalog 를 띄워 선택한 이미지의 태그를 수정할 수 있습니다.



다양한 테스트

02 다양한 테스트

데모 시나리오 #1

〈Story〉
사용자 ‘민호’ 는 평소 SNS를 자주 사용한다. 가족 여행을 다녀온 후, SNS에 사진을 올리고 싶은데 너무 많은 사진들 사이에서 엄마와 같이 찍은 사진을 찾아 올리고자 한다.

1) PhotoPick 어플리케이션을 실행한다.
2) Home 메뉴에서 ‘엄마‘ 폴더를 들어간다.
3) ‘엄마‘ 폴더의 하위 폴더인 ‘같이‘ 폴더를 들어간다.
4) 엄마가 여러 사람들과 함께 같이 찍힌 사진들 중, 나와 같이 찍은 사진을 찾는다.

02 다양한 테스트

데모 시나리오 #2

〈Story〉
사용자 ‘민호’ 는 가족 여행을 다녀온 후, 할머니의 사진을 보게 된다. 할아버지께서 찍으신 사진이나 사진 속에는 할머니 밖에 없어 할머니의 폴더로 분류되었다. 누가 찍은 사진인지 기억하기 위해 태그를 수정하고자 한다.

1) PhotoPick 어플리케이션을 실행한다.
2) Home 메뉴에서 ‘할머니‘ 폴더를 들어간다.
3) ‘할머니‘ 폴더의 하위 폴더인 ‘혼자‘ 폴더를 들어간다.
4) 할아버지가 찍으신 사진을 클릭한다.
5) 수정 버튼을 누르고 할아버지에 체크한다.

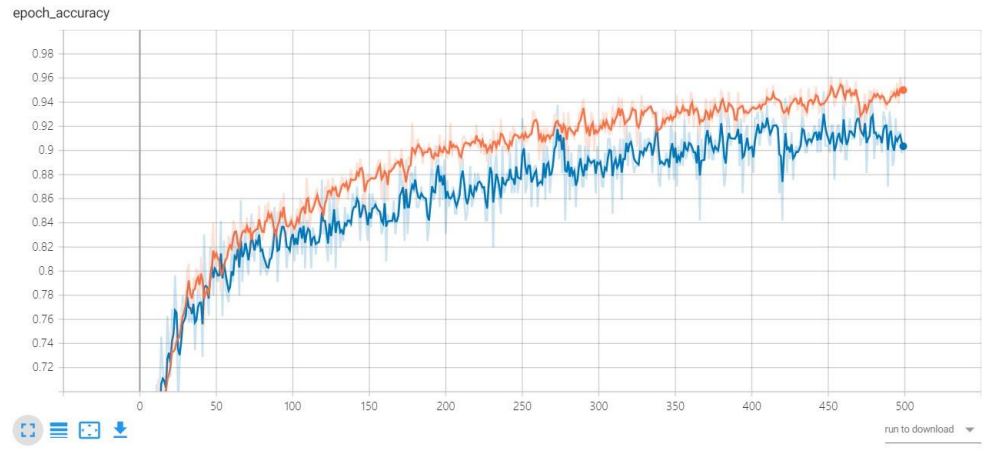
데모 시나리오 #3

〈Story〉
사용자 ‘민호’ 는 PhotoPick에서 ‘아빠’ 사진을 확인하다가 엄마 사진이 잘못 인식되어 ‘아빠‘ 폴더에 들어가 있는 것을 확인한다. 이에 수동으로 태그를 ‘아빠’ 에서 ‘엄마 ’ 로 고쳐 주려 한다.

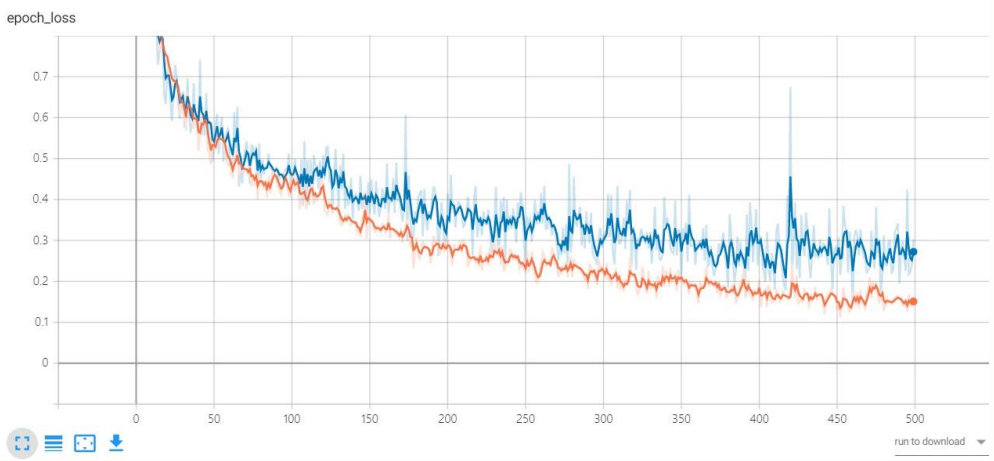
1) PhotoPick 어플리케이션을 실행한다.
2) Home 메뉴에서 ‘아빠‘ 폴더를 들어간다.
3) ‘아빠‘ 폴더의 하위 폴더인 ‘혼자‘ 폴더를 들어간다.
4) 잘못 태그되어 있는 엄마 사진을 클릭한다.
5) 수정 버튼을 누르고 아빠 태그를 해제하고 엄마 태그에 체크한다.
6) Home 탭으로 나와 ‘엄마’ 폴더에 들어간다.
7) 하위 폴더인 ‘혼자’ 폴더에 들어가 태그가 수정된 사진이 들어가 있는 것을 확인한다.

02 다양한 테스트

System Test



epoch 횟수에 따른 accuracy 추이 그래프



epoch 횟수에 따른 loss 추이 그래프

02 다양한 테스트

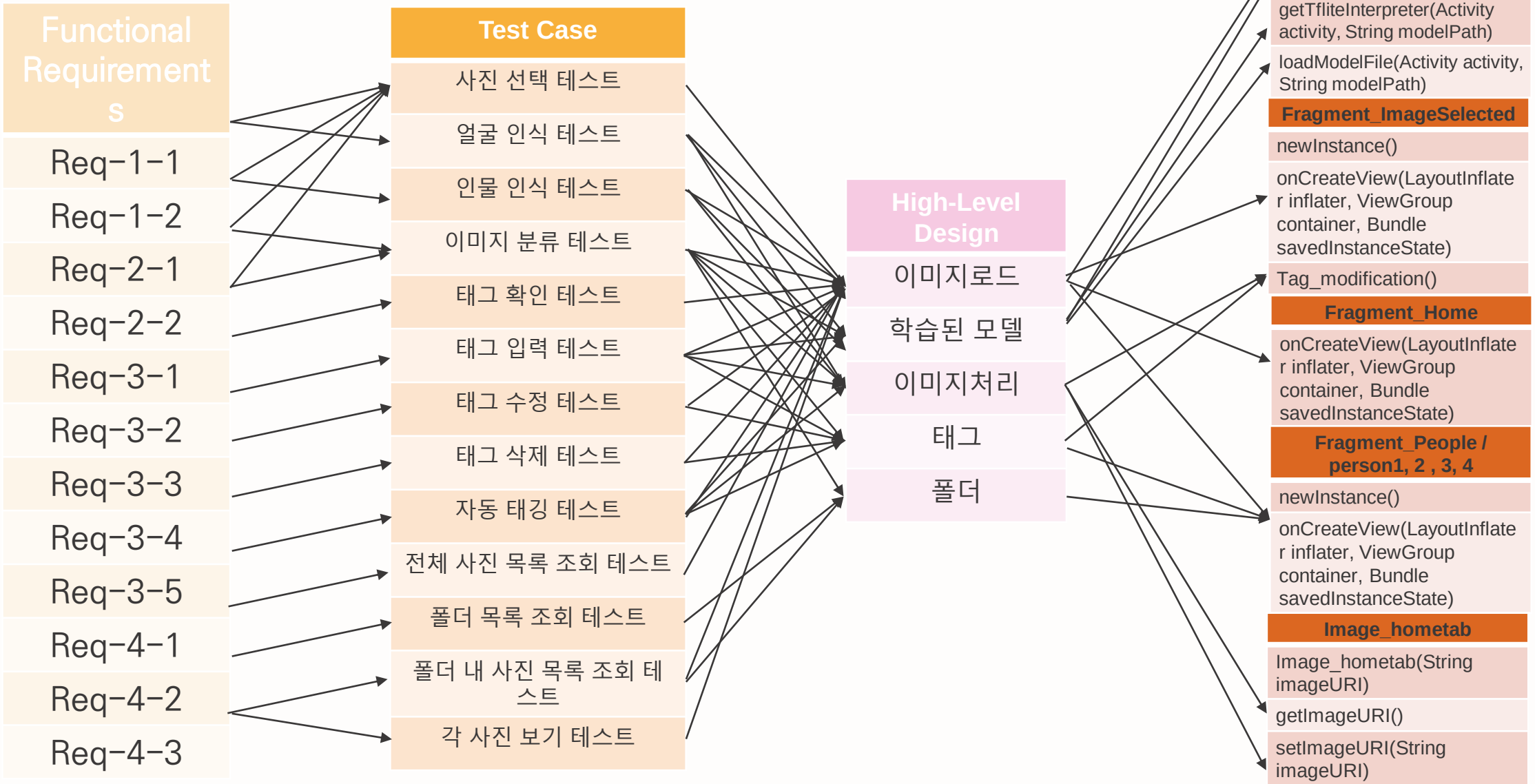
Pass/Fail Criteria

Test Case	Description	Pass /Fail
사진 선택	선택한 이미지가 화면에 표시되는지 테스트한다.	Pass
얼굴 인식 – A	이미지에서 인물의 수를 제대로 분류(혼자/같이)하는지 테스트한다.	Pass
얼굴 인식 – B	인물을 포함하지 않은 이미지를 제대로 분류(Others)하는지 테스트한다.	Pass
	정확도가 90%가 넘는지 테스트한다.	Fail
인물 인식– A, 이미지 분류 – A	사전 학습된 인물을 인식하고 제대로 분류되는지 테스트한다.	Pass
	정확도가 90%가 넘는지 테스트한다.	Fail
인물 인식 – A – 2, 이미지 분류 – B	학습되지 않은 인물과 학습된 인물이 포함된 이미지에서 각각 제대로 분류되는지 테스트한다.	Pass
	정확도가 90%가 넘는지 테스트한다.	Fail
인물 인식 – A – 3, 이미지 분류 – B	학습된 인물이 두 명 이상 포함된 이미지에서 학습된 인물을 제대로 분류하는지 테스트한다.	Pass
	정확도가 90%가 넘는지 테스트한다.	Fail

Test Case	Description	Pass /Fail
태그 확인	사진을 클릭하여 저장된 태그를 확인할 수 있는지 테스트한다.	Pass
태그 수정	기존의 태그에서 다른 태그로 변경할 수 있는지 테스트한다.	Pass
태그 입력	기존의 태그에 새로운 태그를 추가할 수 있는지 테스트한다.	Pass
자동 태깅	학습된 모델을 바탕으로 새롭게 입력된 사진에 자동으로 태그가 부착되는지 테스트한다.	Pass
전체 사진 목록 조회	‘home’ 탭을 눌렀을 때 핸드폰 내에 저장된 전체 사진 목록을 출력할 수 있는지 테스트한다.	Pass
폴더 목록 조회	‘Photopick’ 탭을 눌렀을 때 기존에 학습된 인물의 폴더 목록에 출력되는지 테스트한다.	Pass
어플리케이션 부팅 시간	실행하여 2 초 내로 실행 화면이 뜨는지 테스트한다.	Fail
폴더 선택 시 이미지 로딩 시간	선택한 인물의 폴더에 접속했을 때 3 초 이내로 이미지 로딩이 완료되는지 테스트한다.	Pass

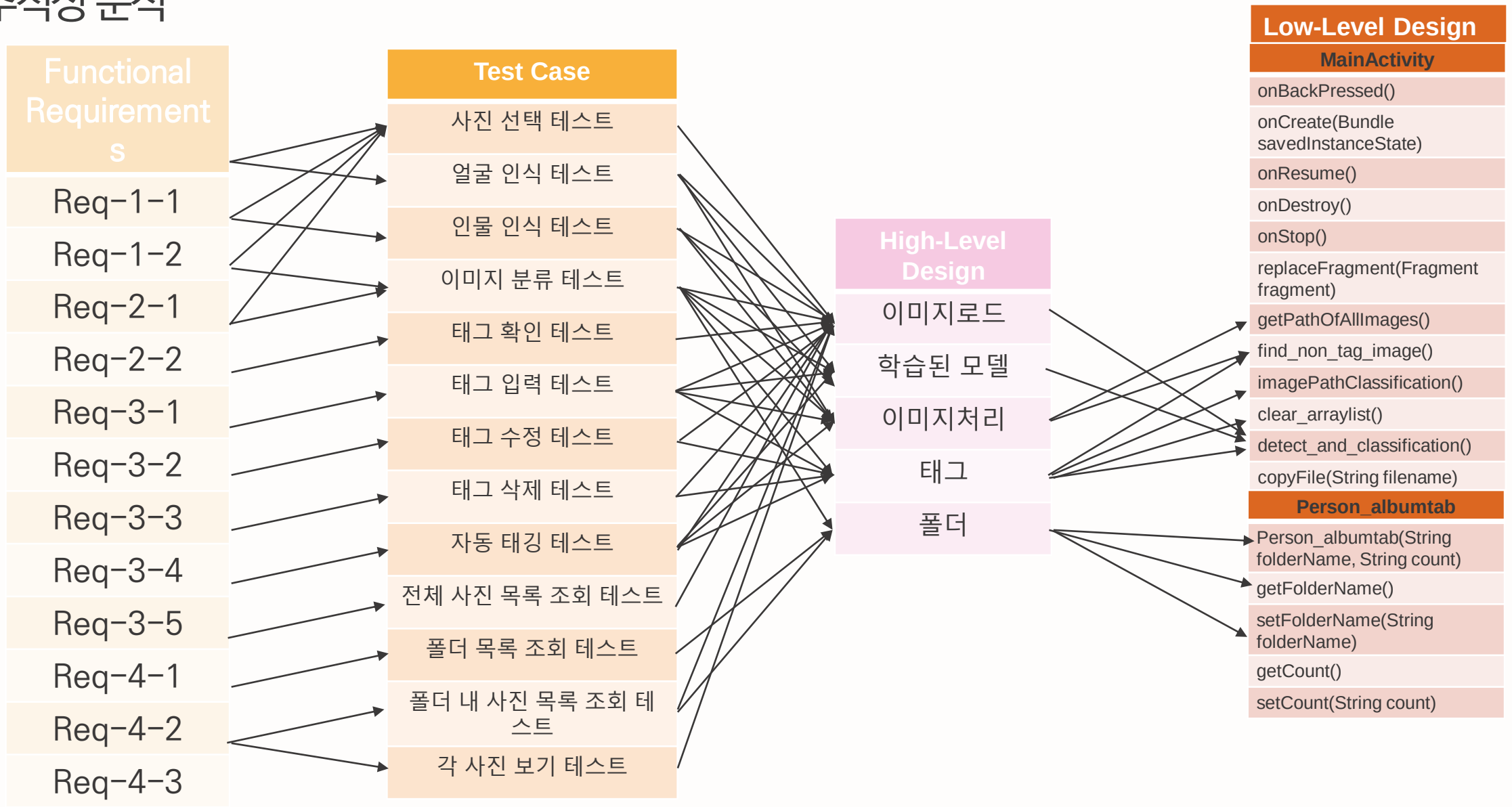
02 다양한 테스트

추적성 분석



02 다양한 테스트

추적성 분석



향후 전망

보완점

감사합니다