

**라즈베리 파이를 이용한**

**안면인식**

**1. OpenCV**

1.1 OpevCV 정의

1.2 OpenCV 활용

**2. CNN(Convolution Neural Network)**

2.1 CNN의 정의

2.2 Keras를 이용한 CNN모델 구축

2.2.1 Keras를 이용한 CNN의 구성 요소

2.2.2 Conv2D Layer

2.2.3 MaxPooling2D Layer

2.2.4 Flatten Layer

2.2.5 CNN모델 구축

2.3 CNN모델 학습

2.3.1 모델 이미지 학습

2.3.2 모델 평가

2.4 모델 사용

2.5 학습 결과

**3. 안드로이드 어플리케이션**

3.1 구현 범위

3.2 설계

3.2.1 User InterFace

3.2.2 AsyncTask

3.2.3 얼굴 촬영

**4. 라즈베리파이**

4.1 구현 범위

4.2 결과 수신 및 제어

**5. 서버와 클라이언트**

5.1 서버 구축

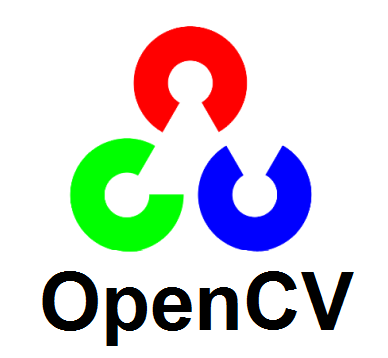
5.2 서버 구성파일

**1. OpenCV**

1.1 OpenCV의 정의

Open Source Computer Vision의 약자로, 다양한 영상 및 동영상 처리에 사용 할 수 있는 오픈소스 라이브러리를 말한다.

이 라이브러리에는 성능과 속도면에서 뛰어난 정확도를 가진 2500개 이상의 최적화된 알고리즘이 포함되어 있으며, 물체를 식별하고 얼굴을 감지 또는 인식하는등 많은 분야에서 사용되고 있다

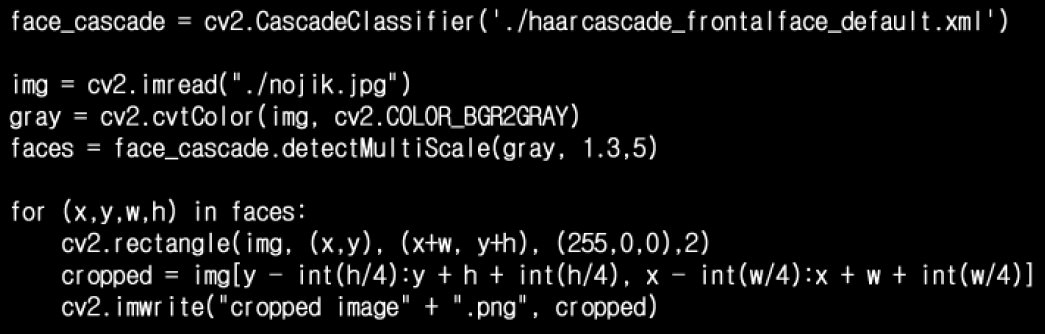


[그림 1-1] OpenCV

1.2 OpenCV 활용

본 실험에서는 안드로이드 카메라를 활용한다. 따라서 카메라가 인식한 사진을 서버에 전송해야 하는데 OpenCV를 활용하여 불필요한 정보를 소거하고 인식에 필요한 얼굴 부분만을 활용한다.

다음 코드는 서버에 전송된 사진을 OpenCV가 얼굴 부분만 잘라서 딥러닝 머신에 장착됨을 뜻한다.

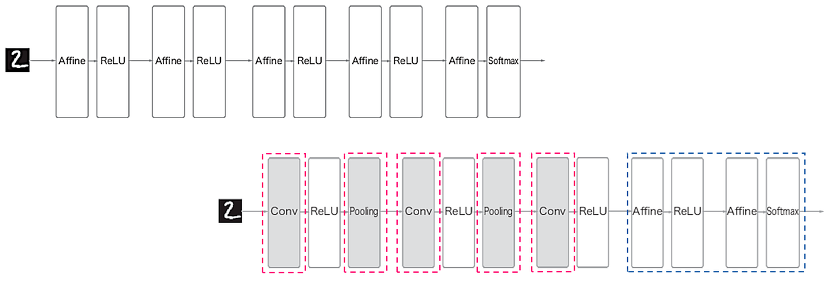


[그림 1-2] OpenCV 활용

**2. CNN(Convolution Neural Network)**

2.1 CNN의 정의

CNN은 3차원의 데이터를 다음 레이어로 보낼 수 있기 때문에 영상이나 이미지 분석에 많이 사용된다. 영상 데이터의 경우 3차원의 형상을 가지게 되는데, 이 형상은 공간적 구조를 갖고 있다. 이 공간적 구조는 공간적으로 가까운 픽셀은 값이 비슷하거나, RGB의 각 채널이 서로 밀접하게 관련되어 있거나, 거리가 멀리 떨어져 있는 픽셀은 관련이 없는 성질이다. CNN을 사용하지 않은 완전연결계층의 경우 3차원을 나타내기 위해서 1차원으로 모두 펼쳐서 연산하기 때문에 위에 설명한 공간적 구조에 대한 데이터가 사라지게 된다. 이러한 이유들로 인해서 영상이나 이미지 분석에는 CNN이 사용된다.



[그림 2-1] 완전연결계층과 CNN의 Layer

2.2 Keras를 이용한 CNN 모델구축

2.2.1 Keras를 이용한 CNN의 구성 요소

CNN의 구현은 Keras를 통해서 구현할 수 있다. 이미지 분석을 위해서 Convolution Layer와 MaxPooling Layer, Flatten layer에 대해서 설명한다.

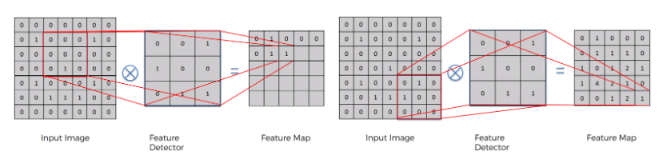
2.2.2 Conv2D Layer

Keras에서는 이미지 인식을 위해서 필터가 탑재된 Convolution Layer를 사용한다. 이는 Keras라이브러리에 포함된 Conv2D클래스를 이용하여 구현할 수 있다.



[그림 2-2] Conv2D Layer

[그림2-2]에서 볼 수 있는 Conv2D레이어의 첫 번째 인자와 두 번째 인자는 필터의 개수와 각 필터의 크기이다. 이 두 인자를 이용해서 들어온 이미지의 영역을 훑으면서 Feature를 뽑아낸다. 뽑아낸 특징은 다시 Feature Map을 만들어서 축적한다.



[그림 2-3] Feature Map 생성 과정

32개의 필터가 [그림2-3]처럼 훑으면서 Feature Map을 생성한 후 Convolution Layer를 이루게 된다.

세 번째 인자는 입력되는 이미지의 크기와 채널을 나타낸다. 현재는 가로 256, 세로 256, 3개의 채널로 구성된 이미지가 입력될 것이라는 것을 알려주고 있다.

네 번째 인자는 활성화 함수를 설정한다. 여기서 활성화 함수란, 어떠한 신호를 입력받아서 이를 적절하게 처리해주는 함수이다. 활성화 함수의 종류는 다양한데 여기서는 ReLU 함수를 사용한다. Rectified Linerar Unit의 약자로 선형함수라고 부른다. 이 함수는 0보다 작은 값은 0을 반환하고, 0보다 큰 값에 대해서는 해당하는 값을 그대로 반환한다. 즉, [그림2-2]의 결과 0보다 작은 값에 해당하는 픽셀은 모두 0으로 반환하는 결과를 보여준다.

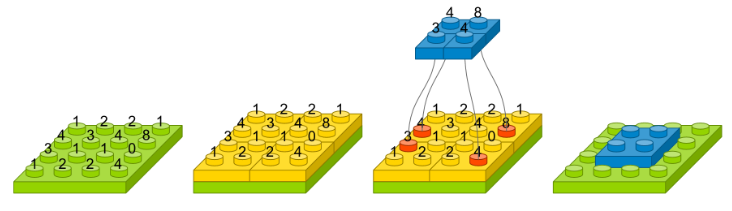
2.2.3 MaxPooling2D Layer

MaxPooling2D Layer는 Conv2D Layer의 반환 값 중에서 주요한 값만 뽑아서 이미지의 크기를 줄인다. 이러한 과정은 많은 학습 이미지 중에서 지역적으로 사소한 변화가 학습 결과에 영향을 미치지 않도록 하는 과정이다.



[그림 2-4] MaxPooling2D Layer

[그림 2-4]의 첫 번째 인자인 pool\_size는 수직, 수평 축소비율을 지정한다.



[그림 2-5] 입력 이미지가 (4,4)인 이미지의 MaxPooling2D Layer예

[그림 2-5]를 통해서 볼 수 있듯이, (2,2)인 pool을 만나면 가중치가 가장 큰 값만 뽑아서 이미지의 크기를 줄이면서 사소한 변화를 무시한다.

2.2.4 Flatten Layer

Keras CNN에서는 Conv2D Layer와 MaxPooling2D Layer를 반복적으로 거치면 이미지의 주요 특징만 추출된다. 추출된 주요 특징들을 전결합층으로 보내기 위해서는 1차원의 자료로 바꿔줘야 한다. 이 역할을 하는 것이 Flatten Layer이다.



[그림 2-6] Keras에서 Flatten Layer

[그림 2-6]에서 볼 수 있는 것처럼 Flatten Layer는 별도의 인자가 없다.

2.2.5 CNN모델 구축

모델을 정의했으니 학습시키기 위해서 엮어줘야 한다. 이는 케라스에서 compile 클래스를 사용한다.



[그림 2-7] Keras에서 compile 클래스

첫 번째 인자는 출력으로 나온 가중치중 최적의 가중치를 검색하는데 사용한다. Keras에서는 이를 경사 하강법 알고리즘 중 하나인 adam을 사용한다.

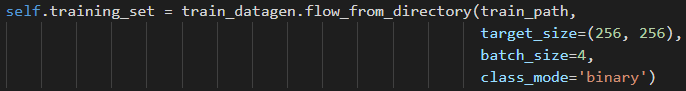
두 번째 인자는 현재의 가중치 세트를 평가하는데 사용하는 손실 함수이다. 현재는 등록된 사용자가 맞는지를 평가해야 하기 때문에 인자 값으로 이진 문제를 다루는 binary\_crossentropy를 사용한다.

세 번째 인자는 평가 척도를 나타내며 본 연구에서는 정확도가 중요하기 때문에 accuracy를 사용한다.

2.3 CNN 모델 학습

2.3.1 모델 이미지 학습

Keras에서는 이미지 학습을 위해서 ImageDataGenerator클래스를 제공한다. 본 연구에서는 사용자의 편의를 위해서 학습시키는 이미지의 수를 줄여야 하기 때문에 학습 데이터를 증강 시켜주는 기능이 포함된 ImageDataGenerator 클래스를 사용한다.



[그림 2-8] Keras에서 ImageDataGenerator클래스

ImageDataGenerator객체를 생성한 후 flow\_from\_directory()함수를 사용한다. 첫 번째 인자는 훈련할 이미지 경로를 설정한다. 이미지 경로는 미리 변수로 설정해 두었다.

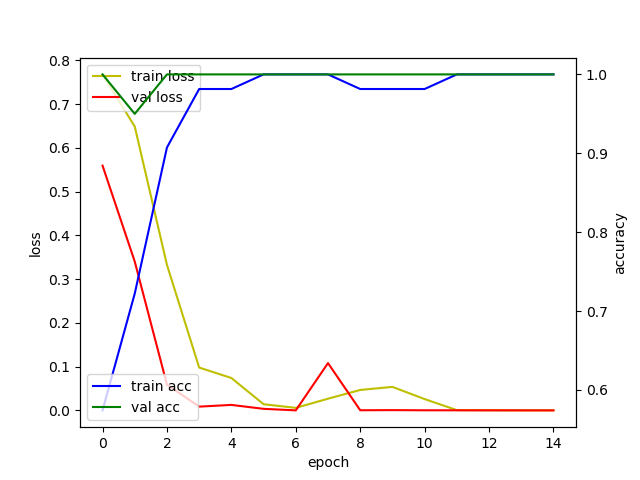
두 번째 인자는 훈련할 이미지의 크기이다. 여기서 이미지의 크기는 원본의 크기가 아닌, 모델에 처음 들어가는 입력 크기다. [그림 2-2]에서 볼 수 있듯이, input\_shape에 들어가는 값을 입력한다.

세 번째 인자는 배치 크기를 지정한다. 배치 크기는 한 epoch에서 몇 장의 사진씩 학습할 것인가이다. 본 연구에서는 30장의 학습 이미지와 10장의 테스트 이미지가 사용되기 때문에, 배치 크기를 4로 할당한다면, 한 epoch에서 4장에 사진씩 10번을 학습하게 된다.

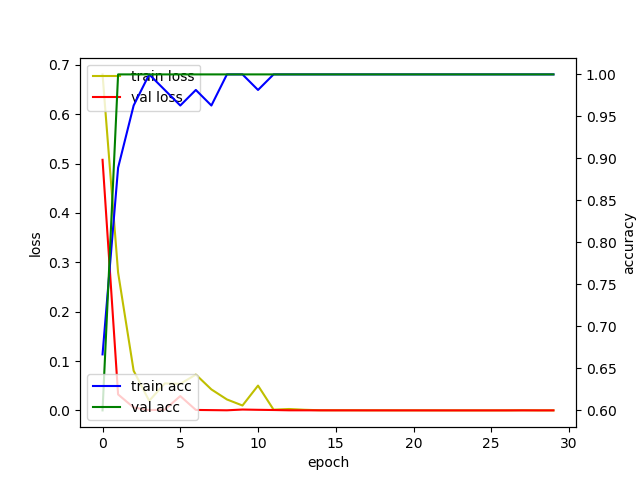
네 번째 인자는 분류 방식에 대해서 지정한다. [그림 2-7]과 같이 본 연구에서는 이진 문제를 다루기 때문에 binary로 설정한다.

2.3.2 모델 평가

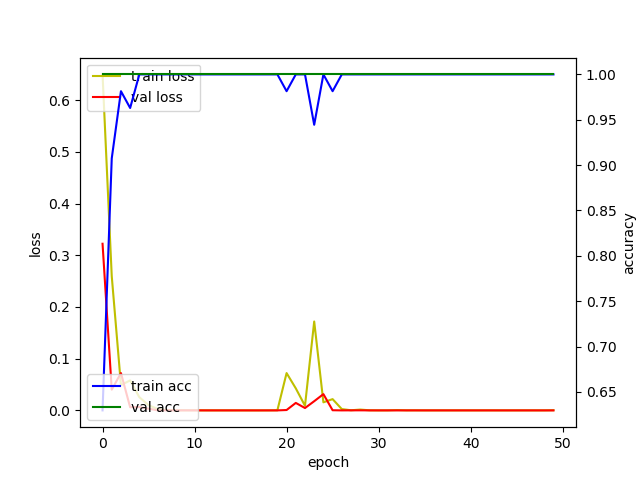
모델의 구성과 이미지 학습까지 완료되었으니 이제 최적의 epoch를 찾아야 한다. 최적의 epoch를 찾는 방법은 Keras에서 제공하는 라이브러리를 이용해서 할 수도 있지만, 학습 데이터의 수가 적기 때문에 직접 여러번의 학습을 시키면서 최적의 epoch를 찾는다. 아래 그림에서 train은 훈련, val은 검증이다. loss는 손실, acc는 정확도 이다.



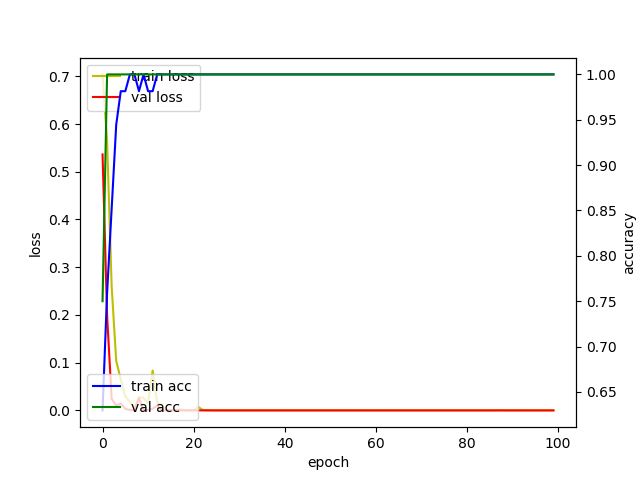
[그림 2-9] 15 epoch 결과



[그림 2-10] 30epch 결과



[그림 2-11] 50epoch 결과



[그림 2-12] 100epoch 결과

15epoch 이상부터는 공통적으로 검증 정확도(val acc)가 1.00으로 수렴하는 것을 알 수 있다. 이를 통해 epoch를 무의미하게 늘려서 훈련 시간을 증가시킬 이유가 없다는 것을 확인할 수 있다. 따라서 본 연구에서는 20epoch를 사용한다. 이렇게 학습된 모델은 model.h5라는 파일로 저장되어 사용한다.

2.4 모델 사용

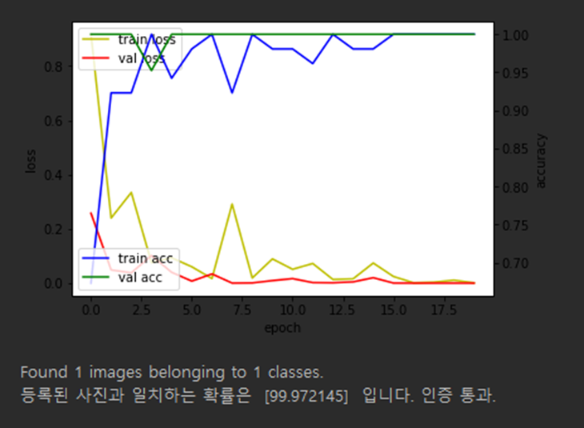
Keras에서 모델의 사용은 fit()함수를 사용해서 한다.



[그림 2-13] Keras에서 fit()함수의 사용

첫 번째 인자는 훈련 데이터의 위치, 두 번째 인자는 테스트 이미지의 위치, 세 번째 인자는 epoch의 횟수이다.

2.5 학습 결과



[그림 2-14] 20epoch 학습 결과

[그림 2-14]는 20epoch로 학습된 모델을 사용한 그림이다. 학습된 사진과는 다르지만 동일한 인물의 사진을 넣었을 때 99.97%의 확률로 같은 인물이라는 것을 예측한다.

**3. 안드로이드 어플리케이션**

3.1 구현 범위

본 연구에서는 사용자의 얼굴 사진을 촬영해서 서버로 전송하는 어플리케이션이 필요하다. 안드로이드에서 인터넷을 사용하기 위해서는 AsyncTask를 사용하여 메인스레드가 아닌 백그라운드에서 작업을 하게 설정한다.

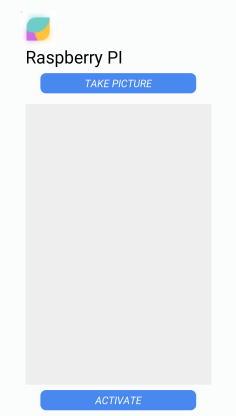
(1) 직관적으로 알기 쉬운 UI

(2) 메인 스레드의 부담을 줄이기 위해 AsyncTask 사용

(3) 사용자의 얼굴 촬영을 위해서 카메라 기능 사용

3.2 설계

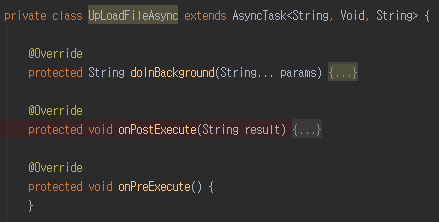
3.2.1 User Interface



[그림 3-1] 어플리케이션 UI

3.2.2 AsyncTask

안드로이드에서 인터넷을 사용할 때는 메인 스레드의 부담을 줄이기 위해서 백그라운드에서 작업하는 AsyncTask를 사용해야 한다.



[그림 3-2] AsyncTask의 문법

AsyncTask는 크게 3가지의 필드를 갖고 있다.

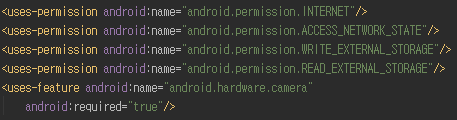
onPreExecute()는 AsyncTask를 사용하기 전 해야 할 작업에 대해서 설정한다. 사용자에게 입력 정보를 받는다거나 작업을 시작하기 전 사용자에게 알림으로 알려주는 행동을 할 수 있다.

doInBackground()는 onPreExecute()의 실행이 끝나면 바로 실행된다. 이는 메인 스레드가 아닌 백그라운드에서 서버와 통신을 하는 등의 작업을 할 수 있다.

onPostExecute()는 서버에서 받은 값 등을 사용자에게 적절하게 변환해서 알려주는 작업을 할 수 있다. 위 세가지 필드를 통해서 메인스레드의 부담을 줄여주는 AsyncTask를 사용할 수 있다.

3.2.3 얼굴 촬영

얼굴을 촬영하기 위해서 카메라 기능을 사용해야 한다. 안드로이드에서는 기본으로 카메라 사용이 꺼져있기 때문에 이를 활성화 해줘야한다. 안드로이드의 기본적인 권한을 설정하는 파일인 AndroidManifest.xml파일을 통해서 카메라사용을 허용해준다. 또한 촬영한 사진을 서버로 전송해야 하기 때문에 데이터를 전송하는 권한도 허용해 준다.



[그림 3-3] 카메라 설정과 인터넷 전송 허용

**4. 라즈베리파이**

4.1 구현 범위

본 실험에서는 서버에서 진행된 딥러닝의 결과를 받아 출력할 라즈베리파이가 필요하다. 라즈베리파이는 서버에서 머신러닝을 통해서 예측한 결과를 받아서 출력하는 역할을 수행한다.

4.3 결과 수신 및 제어

라즈베리파이에서 결과를 수신받기 위해서는 목적에 맞는 py파일이 필요하다.

[그림 4-1] 결과를 수신받는 receive.py 파일

[그림 4-1]의 코드는 주기적으로 서버에 있는 json파일을 읽는다. 읽은 파일의 timestamp가 변하게 되면 파일을 받아서 확인한다. json파일에는 머신러닝 연산 결과가 담겨 있고 그 결과를 사용하여 도어락을 제어한다.

**5. 서버와 클라이언트**

5.1 서버 구축

|  |  |
| --- | --- |
| CPU | 인텔 AVX 터보 1코어 3.3GHz |
| Memory | 1 Gib |
| Storage | EBS 전용 |
| Network | 낮음에서 중간 |

[표 5-1] AWS EC2 t2.micro 사양

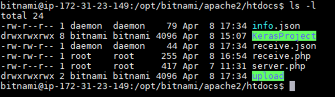
본 연구에서는 AWS(Amazon Web Service)에서 제공하는 EC2(Elastic Compute Cloud)를 사용하여 서버를 구축하였다. EC2에서 사용할 수 있는 많은 어플리케이션 중 본 연구에서는 LAMP스택을 사용한다. LAMP스택은 Linux운영체제 위에서 Apache서버를 사용하며 MySQL데이터베이스와 PHP를 사용한다. 서버의 인스턴스 유형은 t2.micro를 사용하며 자세한 항목은 [표5-1]로 확인할 수 있다.



[그림 5-1] 서버와 클라이언트의 구성 요소

5.2 서버 구성파일

서버의 역할은 안드로이드 어플리케이션에서 보낸 이미지를 저장하고 이를 바탕으로 학습된 모델을 이용하여 결과를 예측해야 한다. 서버에 업로드된 폴더와 파일은 [그림 5-2]에서 확인할 수 있다.



[그림 5-2] 서버에 업로드된 폴더와 파일

upload폴더에는 안드로이드 어플리케이션에서 보낸 사진이 저장된다. 이 사진은 계속 저장되어 쌓이지 않고, 결과를 예측한 후 삭제되어 사용자의 개인정보가 남아있지 않게 한다.

KerasProject폴더에는 2장에서 설명한 Keras를 사용한 CNN모델의 학습 결과물인 model.h5와 이를 실행시키는 Python파일이 들어있다. 사진이 새로 upload되면 info.json파일의 timestamp를 새로 작성한다. 그 후, model.h5를 사용하여 결과를 예측하기 시작하고 결과와 함께 receive.json파일에 timestamp를 작성한다.

info.json파일에는 upload폴더에 이미지가 업로드 되면 timestamp가 작성된다.

receive.json파일에는 예측 결과와 timestamp가 작성되어 있다.