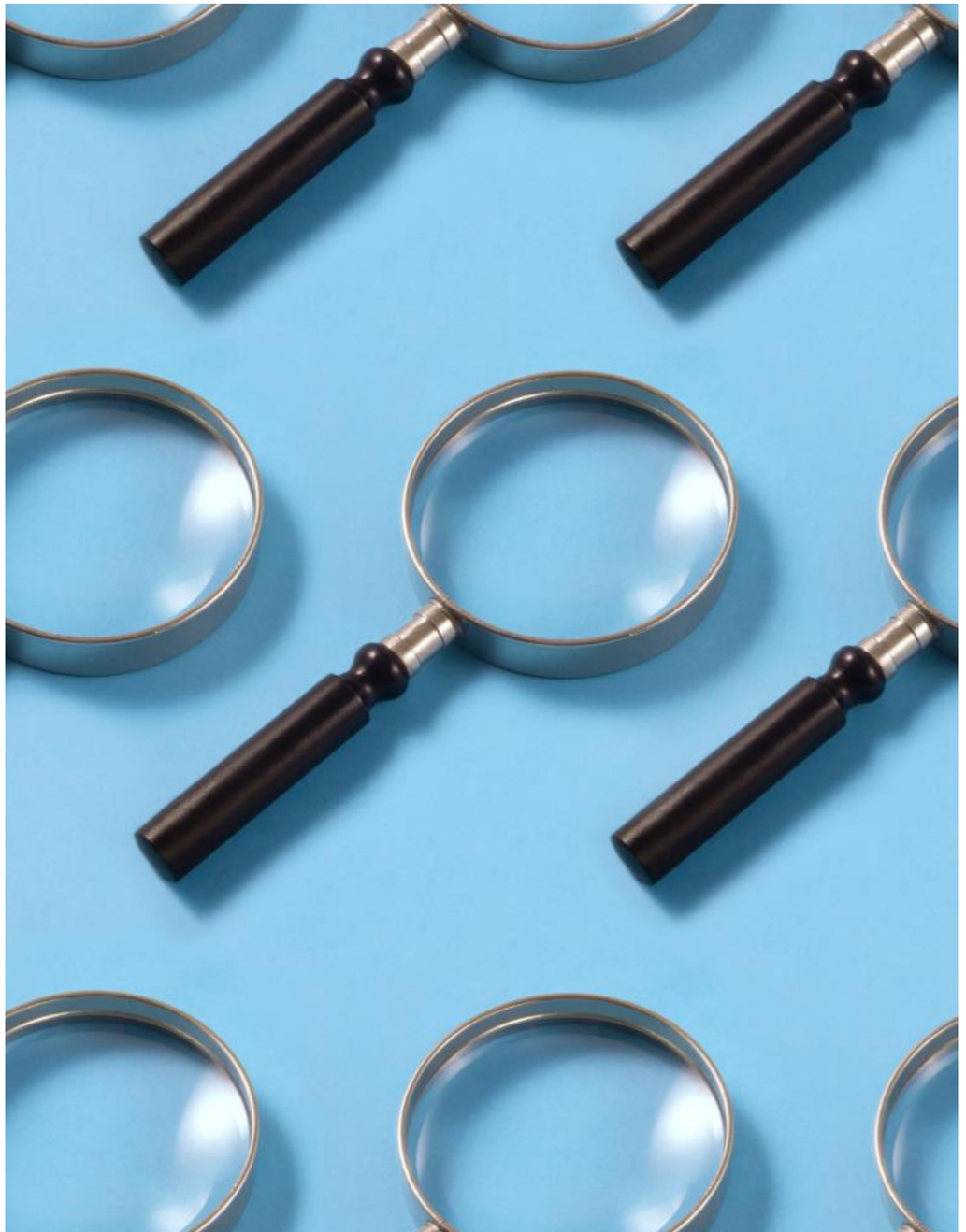
인공지능론

중간보고서

2018100910 서정민



<유명 축구 선수 얼굴 식별>

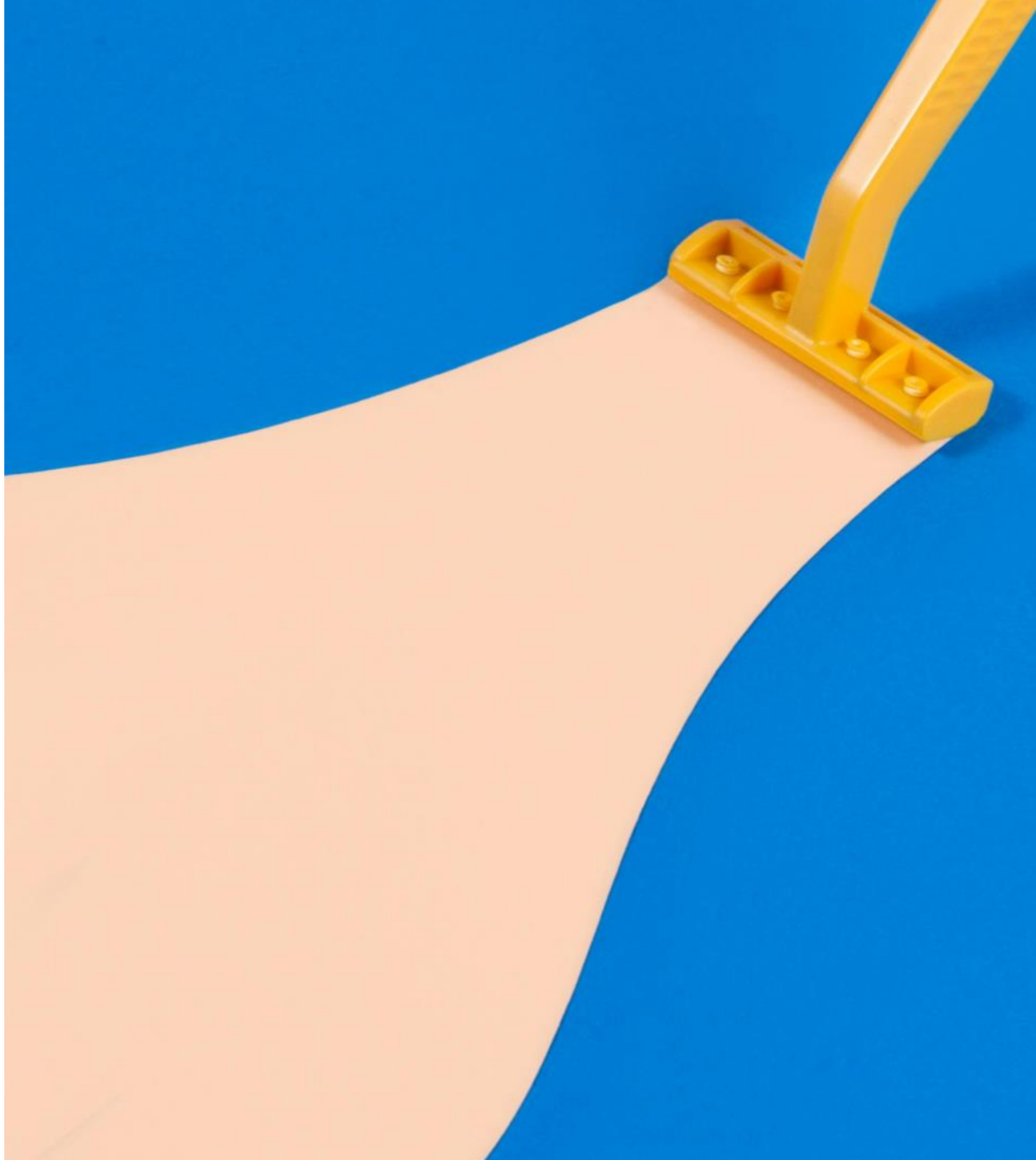
(세계적인 축구선수 5명의 얼굴 이미지 학습 후 식별)

Dataset 수집 방법 :

구글 검색 이미지 데이터 크롤링
(얼굴 위주의 사진이 나올 수 있도록 검색어를 선별하여 크롤링 진행)
- 이미지 크롤링에 있어, 깃허브의 파일을 클론하여 사용하였음.

적용 기술 : 딥러닝

적용 기법 : CNN



딥러닝 기반
이미지 인식
기술 수준

이미지 인식 (Image Recognition) 기술은 이미지의 정보를 식별하는 기술로써, 컴퓨터 비전 분야에서 오랫동안 연구된 기술 중 하나입니다. 이미지 인식은 단순히 사진 한 장을 인식하는 것 뿐만 아니라, 자율주행을 위한 무인 자동차 개발, 사람 얼굴을 인식하는 안면 인식, 의료 진단을 위한 엑스레이 사진 분석 등 다양한 산업에 적용됩니다.

1998

이미지 분류에 자주 사용되는 딥러닝의 알고리즘인 CNN의 첫 성공 사례 등장. 딥러닝의 대가 중 한 명인 Yann LeCun 교수의 연구팀이 필기 숫자 인식기를 만들어 수표 인식 자동화 시스템을 구현하였다.

2012

대규모 이미지 인식 경진대회 ILSVRC (ImageNet Large Scale Visual Recognition Challenge)에서 토론토 대학의 제프리 힌튼 교수 산하 연구진이 CNN을 활용해 AlexNet을 만들어, 압도적 성능으로 우승. 당시 정확도는 84.7%로 2위의 정확도와 10.9%가량 차이를 보임.

2015

2012년을 계기로, 딥러닝이 학계와 산업계에 널리 받아들여지게 됨. 이에 따라 딥러닝이 폭발적으로 발전하여, 대규모 이미지 인식 경진대회 ILSVRC 에서 사람의 인식률인 94.9%를 추월한 96.43% 수준까지 인식률이 올라오게 됨. (ResNet)

딥러닝 기반 이미지 인식 기술 동향

04

1

안전하고 신뢰성 있는 이미지 인식

인공신경망이 교통 표지판이나 사람을 인식하지 못하게 하는 사례 등장. 이러한 악의적 이미지 인식 교란에 대해서 올바른 판단을 할 수 있도록 적대적 학습 방법이나 노이즈를 제거, 완화하는 노이즈 감쇄기 방식 등을 연구 진행 중.

2

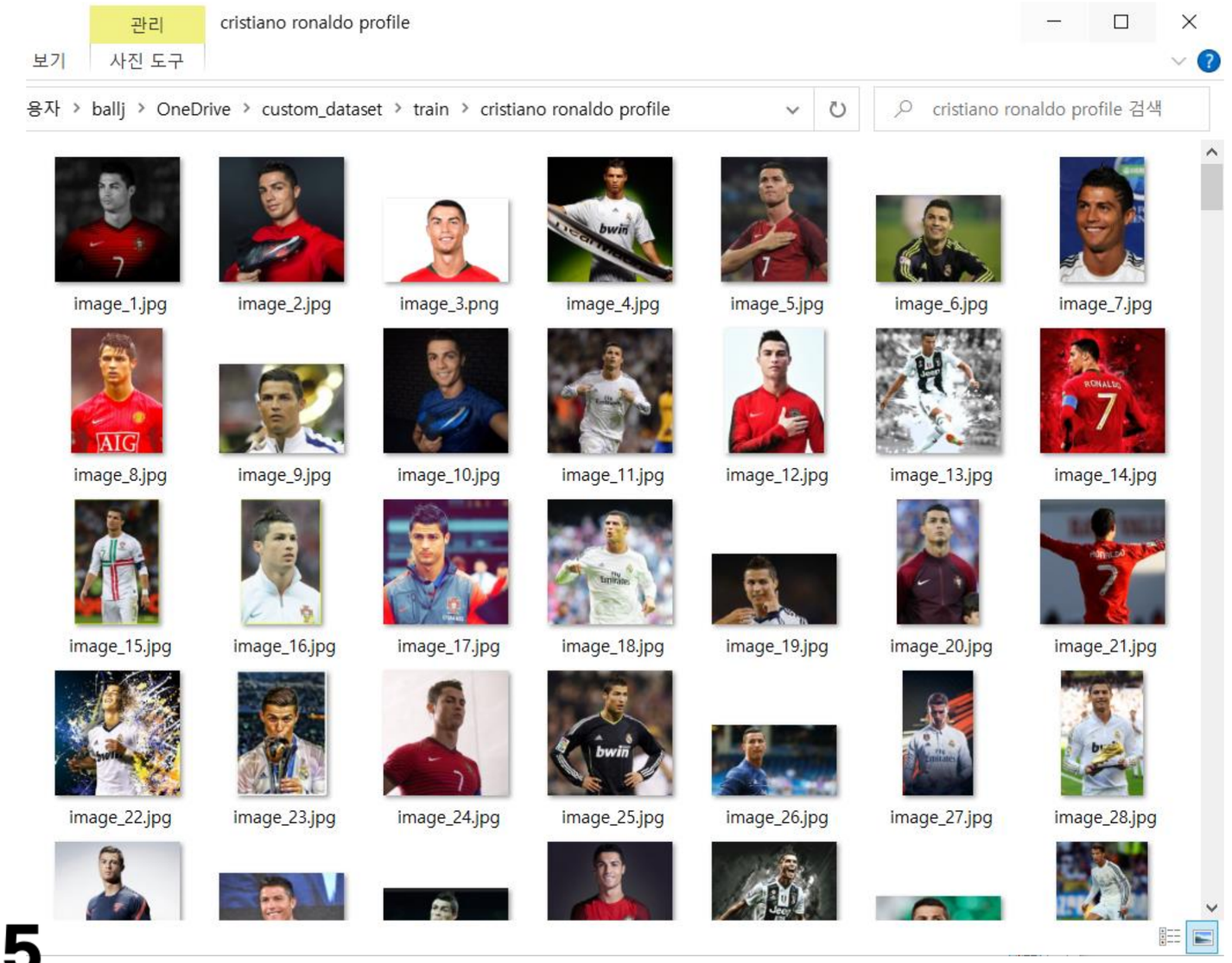
인공지능 학습의 한계 극복

자기 지도 학습을 통한 학습 데이터 절감으로 이미지 어노테이션 비용 문제 극복, 부족한 인공지능 전문가를 대신할 학습 자동화, 학습 데이터와 컴퓨팅 파워 절감을 위한 전이학습 고도화 등의 다양한 연구 진행 중.

3

온 디바이스 (On-Device) 인공지능 이미지 인식

정확도를 유지하며 모델의 크기를 줄이거나 연산을 간소화하여 작은 디바이스 등에 탑재할 수준으로 경량화 하는 경량 딥러닝 연구 진행 중. 또한 인공지능의 판정 연산 가속화를 위한 전용 칩세트 개발 중심의 하드웨어 가속화 기술 연구 진행 중.



데이터 수집 예시
Examples of Data

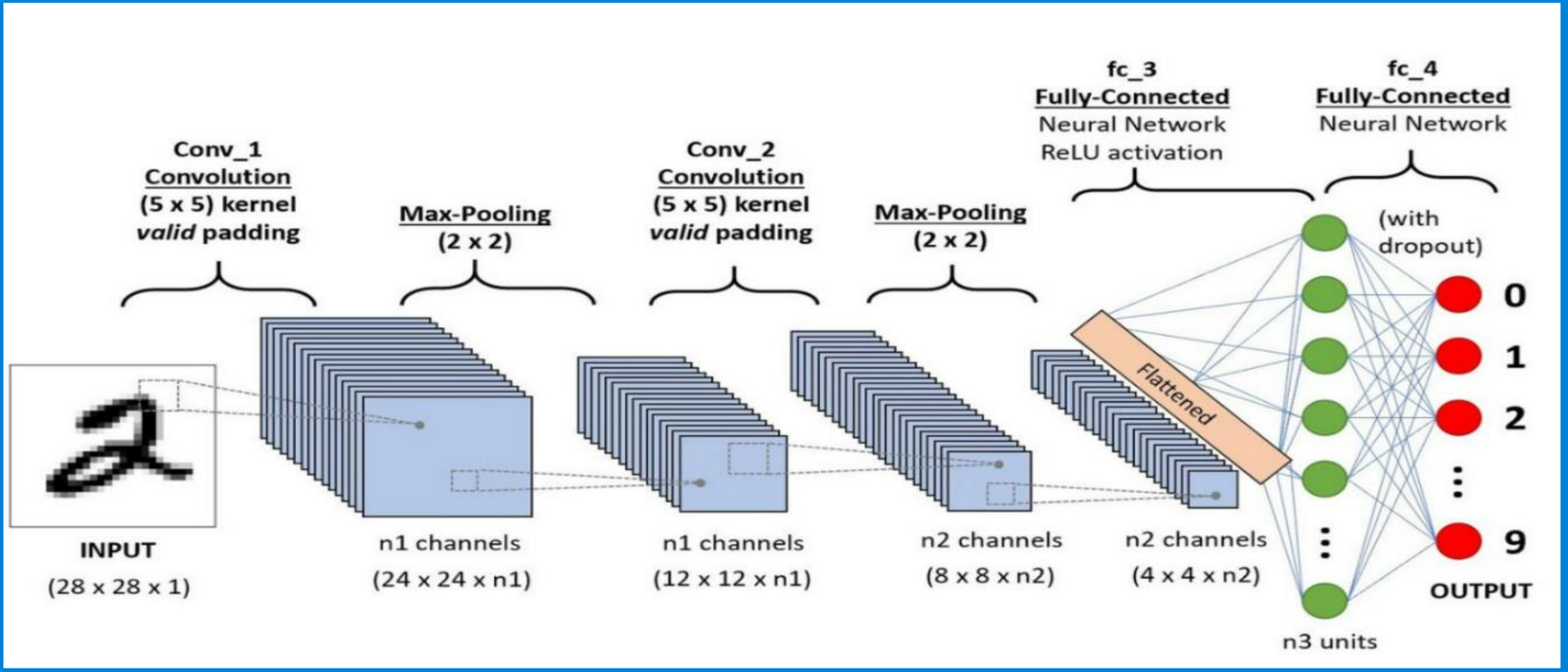
구글 검색 이미지 데이터 크롤링

(얼굴 위주의 사진이 나올 수 있도록

검색어를 선별하여 크롤링)

적용 기법 및 모델

CNN (Convolutional Neural Network)



CNN은 이미지, 비디오, 오디오 등을 분류하는 딥러닝에서 주로 사용되는 알고리즘이며, 이미지에서 얼굴, 객체 등을 인식하는 패턴을 찾는 데에 유용합니다. 또한 CNN은 자동으로 특징을 학습하거나 기존 네트워크를 기반으로 CNN을 재학습하고 새로운 이미지를 인식하도록 만들 수 있는 장점이 있습니다.

적용 기법 및 모델

CNN (Convolutional Neural Network)

```
model = Sequential()
model.add(Conv2D(16, kernel_size = [3,3], padding = 'same', activation = 'relu', input_shape = (64,64,3)))
model.add(Conv2D(32, kernel_size = [3,3], padding = 'same', activation = 'relu'))
model.add(MaxPool2D(pool_size = [3,3]))

model.add(Conv2D(32, kernel_size = [3,3], padding = 'same', activation = 'relu'))
model.add(Conv2D(64, kernel_size = [3,3], padding = 'same', activation = 'relu'))
model.add(MaxPool2D(pool_size = [3,3]))

model.add(Conv2D(128, kernel_size = [3,3], padding = 'same', activation = 'relu'))
model.add(Conv2D(256, kernel_size = [3,3], padding = 'same', activation = 'relu'))
model.add(MaxPool2D(pool_size = [3,3]))

model.add(BatchNormalization())

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation = 'relu'))
model.add(Dense(5, activation = 'softmax')) #분류  class가 5개니까 마지막 출력노드는 5개

model.compile(optimizer = 'adam', loss = keras.losses.categorical_crossentropy, metrics = ["accuracy"])
```

프로젝트 CNN 코드

CNN은 부분적인 수용 영역을 갖는 convolution 커널을 도입하여 층을 쌓아가며 입력 이미지를 학습한다.
CNN을 통해 층을 쌓아가며 입력 이미지의 정보를 학습합하고 점진적으로 특징맵 (feature map)을 학습한다.
CNN은 입력층과 convolution층, pooling층, fully connected층, flattening층 등으로 구성된다.

1) Convolutional layer

Convolution은 입력 특성 맵의 타일을 추출한 후, 이 타일에 필터를 적용하여 새로운 특성을 산출한다.
또한 컨볼루션 단계에서 타일의 크기와 동일한 크기의 행렬인 필터가 입력 특성 맵의 그리드를 한 번에 한 픽셀 씩 가로 및 세로 방향으로 이동하면서 해당 타일을 추출한다.
이렇게 CNN은 각 필터-타일 쌍마다 필터 행렬과 타일 행렬의 요소를 곱하고, 생성된 행렬의 모든 요소를 더하여 하나의 값을 얻는다. 이를 통해 CNN은 유의미한 특성을 추출하는 데에 가장 적합한 필터 행렬의 값이 어느 정도인지 학습하게 된다.
또한 convolutional layer에서는 모델에 비선형을 주기 위해 convolution이 끝날 때마다 합성곱 된 특성에 Relu를 적용시킨다.

적용 기법 및 모델

CNN (Convolutional Neural Network)

```
model = Sequential()
model.add(Conv2D(16, kernel_size = [3,3], padding = 'same', activation = 'relu', input_shape = (64,64,3)))
model.add(Conv2D(32, kernel_size = [3,3], padding = 'same', activation = 'relu'))
model.add(MaxPool2D(pool_size = [3,3]))

model.add(Conv2D(32, kernel_size = [3,3], padding = 'same', activation = 'relu'))
model.add(Conv2D(64, kernel_size = [3,3], padding = 'same', activation = 'relu'))
model.add(MaxPool2D(pool_size = [3,3]))

model.add(Conv2D(128, kernel_size = [3,3], padding = 'same', activation = 'relu'))
model.add(Conv2D(256, kernel_size = [3,3], padding = 'same', activation = 'relu'))
model.add(MaxPool2D(pool_size = [3,3]))

model.add(BatchNormalization())

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation = 'relu'))
model.add(Dense(5, activation = 'softmax')) #분류  class가 5개니까 마지막 출력노드는 5개

model.compile(optimizer = 'adam', loss = keras.losses.categorical_crossentropy, metrics = ["accuracy"])
```

프로젝트 CNN 코드

2) Pooling layer

모델의 전체 매개변수의 수를 줄여, 데이터의 크기를 줄이고자 pooling layer가 존재한다. 이 때, max-pooling과 average pooling의 방법이 존재하는데, 이미지 인식 분야에서는 주로 max-pooling을 사용한다.
또한 이론적 측면에서 계산된 특징이 이미지 내의 위치에 대한 변화에 영향을 덜 받기 때문에 pooling layer가 존재한다. 예를 들어 이미지의 우측 상단에서 눈을 찾는 특징은, 눈이 이미지의 중앙에 위치하더라도 크게 영향을 받지 않아야 한다. 그렇기 때문에 풀링을 이용하여 불변성을 찾아내서 공간적 변화를 극복할 수 있다.
결론적으로 pooling layer는 가장 중요한 특성 정보는 남겨두면서 특성 맵의 차원 수를 줄이는 역할을 한다고 볼 수 있다.

3) Fully connected layer

Fully connected layer는 CNN 끝에 하나 이상으로 존재하는 layer로, convolution을 통해 추출된 특성을 기반으로 분류를 진행한다.
일반적으로 softmax함수를 사용하는데, 이를 통해 예측하고자 하는 분류 라벨 별로 0~1 까지의 확률 값을 출력한다.

프로젝트 코드 설명

Data crawling 및 파일 디렉토리 생성

```python
import os
import shutil
from bing_image_downloader.bing_image_downloader import downloader

directory_list = [
    './custom_dataset/train/'
]

# 초기 디렉토리 만들기
for directory in directory_list:
    if not os.path.isdir(directory):
        os.makedirs(directory)

def dataset(query, train_cnt):
    #데이터셋 디렉토리 만들기
    for directory in directory_list:
        if not os.path.isdir(directory + '/' + query):
            os.makedirs(directory + '/' + query)
    #데이터셋 준비하기
    cnt = 0
    for file_name in os.listdir(query):
        print(f'[Train Dataset] {file_name}')
        shutil.move(query + '/' + file_name, './custom_dataset/train/' + query + '/' + file_name)
        cnt += 1
    shutil.rmtree(query)
```

```python
query = 'cristiano ronaldo profile'
downloader.download(query, limit=500,  output_dir='./', adult_filter_off=True, force_replace=False, timeout=60)
dataset(query, 500)
```

```
#33 이미지 다운로드 (https://avatarfiles.alphacoders.com/228/228641.jpg)
#33 파일 다운로드가 완료되었습니다.
#34 이미지 다운로드 (https://2.bp.blogspot.com/-2FP4xGtiYqw/Wbl3Gfw8m2I/AAAAAAAAEK8/nlAfqkkgrFMqmucqKq5hz06V7I6WqUVdwCLcBGAs/s1600/Cristiano%2BRonaldo%2BHigh%2BResolution%2BHD%2BPhoto.jpg)
#34 파일 다운로드가 완료되었습니다.
```

깃허브를 통해 구글 이미지 크롤링에 있어 적합한 코드 발견. 이를 클론하여 라이브러리로 사용함.

수집한 이미지를 저장하기 위한 폴더를 생성한 후, 데이터셋 디렉토리를 만들고, 데이터 셋을 준비하기.

"cristiano ronaldo profile" 이라는 검색어를 통해 구글 이미지에 검색하여, 500개의 이미지를 크롤링. 마찬가지의 방법으로 이 외의 4명의 선수의 이미지를 각각 500개씩 크롤링하였음.

선수의 이름만 검색할 시, 후면 사진, 선수 이외의 인물 사진 등이 많이 등장하여, profile 이라는 단어를 추가해 검색을 진행하였음.

프로젝트 코드 설명

CNN을 위한 library 호출 및 로컬 폴더로부터 crawling data 불러오기

```
# CNN model 구현에 필요한 library 호출
import tensorflow as tf
from tensorflow.keras import layers, utils
import numpy as np
import pandas as pd
import os
import keras
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, Dropout, BatchNormalization
from keras.preprocessing.image import ImageDataGenerator, load_img
from keras.utils import to_categorical
from keras.models import Sequential
from keras import regularizers
from sklearn.model_selection import train_test_split
import cv2
import matplotlib.pyplot as plt
import seaborn as sns
```

convolution층, pooling층, flattening층 등 CNN 모델 구현을 위한 라이브러리들 호출
train, test dataset을 추후 비율 설정 시 랜덤하게 이미지의 수를 나눠주도록 하는 train_test_split 라이브러리 또한 호출

```
# 크롤링한 폴더로부터 데이터를 불러들이고 train set 카테고리 개수를 출력해준다.
print(os.listdir("C:/Users/ballj/OneDrive/custom_dataset"))
fpath = "C:/Users/ballj/OneDrive/custom_dataset/train"
categories = os.listdir(fpath)
print("No. of categories of images in the train set = ",len(categories))
train_dir = 'C:/Users/ballj/OneDrive/custom_dataset/train'
test_dir = 'C:/Users/ballj/OneDrive/custom_dataset/test'
```

```
['test', 'train']
No. of categories of images in the train set =  5
```

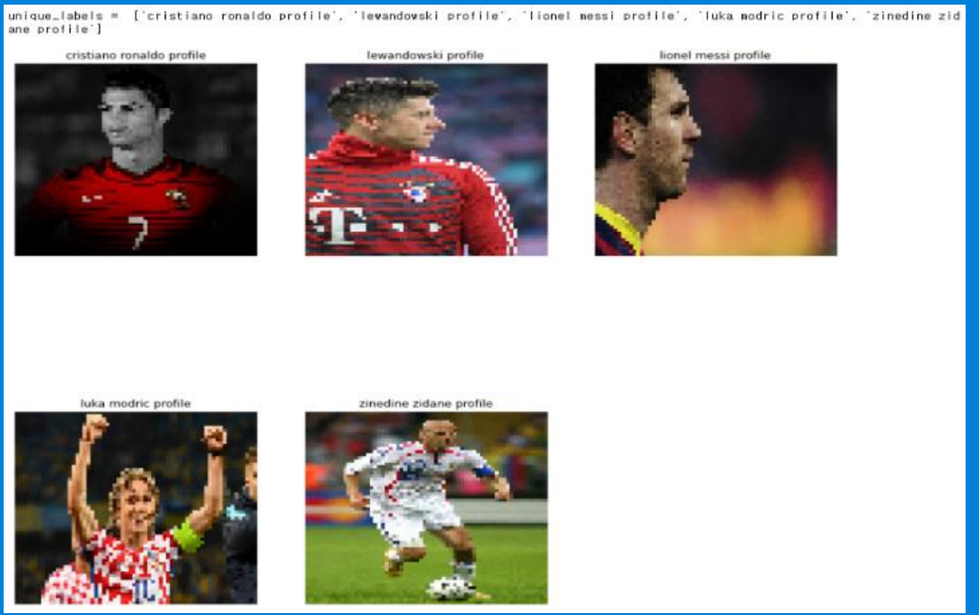
data crawling한 폴더로부터 데이터를 불러들이고, train data set에서 카테고리의 개수를 출력한다.

프로젝트 코드 설명

확인 차 카테고리별 대표 이미지 출력 & 이미지 데이터 resizing

```
# 카테고리별 대표 이미지를 한개씩 출력
# 이미지를 labelling하고 크기를 동일하게 맞춰준 후 화면에 출력한다.
def load_unique():
    size_img = 64,64
    images_for_plot = []
    labels_for_plot = []
    for folder in os.listdir(train_dir):
        for file in os.listdir(train_dir + '/' + folder):
            filepath = train_dir + '/' + folder + '/' + file
            image = cv2.imread(filepath)
            final_img = cv2.resize(image, size_img)
            final_img = cv2.cvtColor(final_img, cv2.COLOR_BGR2RGB)
            images_for_plot.append(final_img)
            labels_for_plot.append(folder)
            break
    return images_for_plot, labels_for_plot

images_for_plot, labels_for_plot = load_unique()
print("unique_labels = ", labels_for_plot)

fig = plt.figure(figsize = (15,15))
def plot_images(fig, image, label, row, col, index):
    fig.add_subplot(row, col, index)
    plt.axis('off')
    plt.imshow(image)
    plt.title(label)
    return

image_index = 0
row = 2
col = 3
for i in range(1,(row*col)):
    plot_images(fig, images_for_plot[image_index], labels_for_plot[image_index], row, col, i)
    image_index = image_index + 1
plt.show()

# 아래 사진을 통해 카테고리 별로 이미지를 잘 불러들여왔음을 확인할 수 있다.
```

이미지 data를 labeling하고, input data의 크기를 64x64 사이즈로 동일하게 맞춘 후, 화면에 예시 이미지 데이터 출력



프로젝트 코드 설명

training dataset과 test dataset을 준비한 후, 이의 비율 설정

```
# train dataset, test dataset의 비율 및 구조를 보여준다.
labels_dict = {'cristiano ronaldo profile':0,'lewandowski profile':1,'lionel messi profile':2,'luka modric profile':3,'zinedine zidane prof

def load_data():
    images = []
    labels = []
    size = 64,64
    print("LOADING DATA FROM : ",end = "")
    for folder in os.listdir(train_dir):
        print(folder, end = ' | ')
        for image in os.listdir(train_dir + "/" + folder):
            temp_img = cv2.imread(train_dir + '/' + folder + '/' + image)
            temp_img = cv2.resize(temp_img, size)
            images.append(temp_img)
            if folder == 'cristiano ronaldo profile':
                labels.append(labels_dict['cristiano ronaldo profile'])
            elif folder == 'lewandowski profile':
                labels.append(labels_dict['lewandowski profile'])
            elif folder == 'lionel messi profile':
                labels.append(labels_dict['lionel messi profile'])
            elif folder == 'luka modric profile':
                labels.append(labels_dict['luka modric profile'])
            elif folder == 'zinedine zidane profile':
                labels.append(labels_dict['zinedine zidane profile'])

    images = np.array(images)
    images = images.astype('float32')/255.0

    templabels = labels

    labels = keras.utils.to_categorical(labels)

    X_train, X_test, Y_train, Y_test = train_test_split(images, labels, test_size = 0.1)

    print()
    print('Loaded', len(X_train),'images for training,','Train data shape =',X_train.shape)
    print('Loaded', len(X_test),'images for testing','Test data shape =',X_test.shape)

    return X_train, X_test, Y_train, Y_test, templabels
```

labeling 후, train dataset 과 test dataset 준비
또한 이미지 데이터를 256 scale에 맞추기 위해
255로 나눠준다.
그리고, train dataset과 test dataset의 비율은
9:1로 설정하였다.

그 결과, training data로는 2250개의 data, test data로는 250개의 data가 사용되었으며,
train data shape = (2250, 64, 64, 3)
test data shape = (250, 64, 64, 3) 출력되었다.

```
# train data와 test data의 개수 출력
X_train, X_test, Y_train, Y_test, labels = load_data()
```

```
LOADING DATA FROM : cristiano ronaldo profile | lewandowski profile | lionel messi profile | luka modric profile | zinedine zidane profile
|
Loaded 2250 images for training, Train data shape = (2250, 64, 64, 3)
Loaded 250 images for testing Test data shape = (250, 64, 64, 3)
```

프로젝트 코드 설명

CNN 모델 구조 정의, 파라미터 설정 및 데이터 훈련 함수 정의

```python
# CNN model을 구축한다.
def create_model():

    model = Sequential()
    model.add(Conv2D(16, kernel_size = [3,3], padding = 'same', activation = 'relu', input_shape = (64,64,3)))
    model.add(Conv2D(32, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(MaxPool2D(pool_size = [3,3]))

    model.add(Conv2D(32, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(Conv2D(64, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(MaxPool2D(pool_size = [3,3]))

    model.add(Conv2D(128, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(Conv2D(256, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(MaxPool2D(pool_size = [3,3]))

    model.add(BatchNormalization())

    model.add(Flatten())
    model.add(Dropout(0.5))
    model.add(Dense(512, activation = 'relu'))
    model.add(Dense(5, activation = 'softmax')) #분류  class가 5개니까 마지막 출력노드는 5개

    model.compile(optimizer = 'adam', loss = keras.losses.categorical_crossentropy, metrics = ["accuracy"])

    print("MODEL CREATED")
    model.summary()

    return model

# 구축한 모델을 그래프로 표현하는 함수이다.
def fit_model():
    model_hist = model.fit(X_train, Y_train, batch_size = 64, epochs = 100, validation_split = 0.1)
    early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
    return model_hist
```

```
MODEL CREATED
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 64, 64, 16)        448
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 64, 64, 32)        4640
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 21, 21, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 21, 21, 32)        9248
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 21, 21, 64)        18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 7, 7, 128)         73856
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 7, 7, 256)         295168
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
batch_normalization_1 (Batch (None, 2, 2, 256)         1024
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 2565
=================================================================
Total params: 930,245
Trainable params: 929,733
Non-trainable params: 512
_________________________________________________________________
```

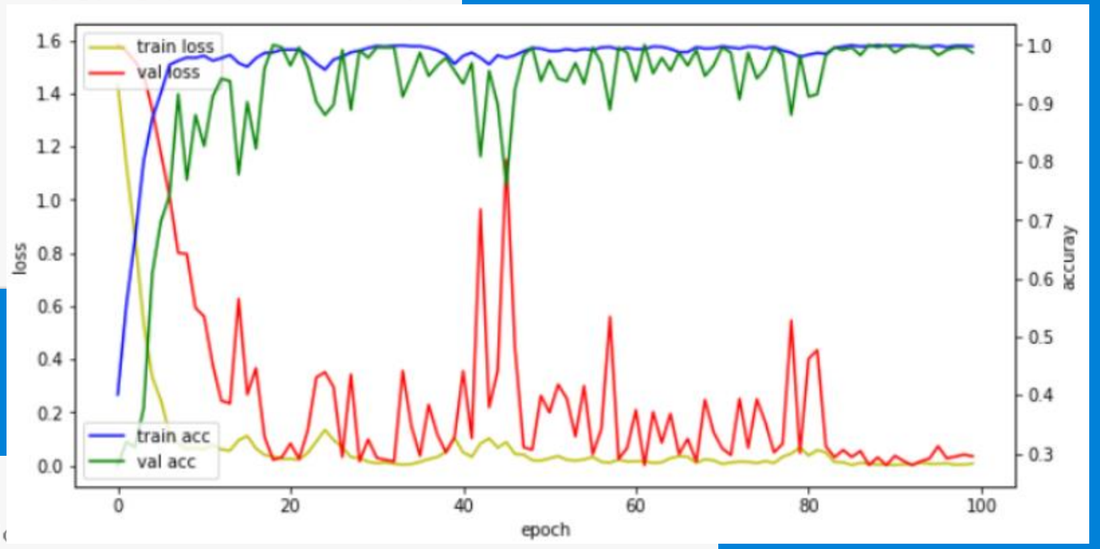
프로젝트 코드 설명

학습 결과 표현

```
# 그래프 표현
import matplotlib.pyplot as plt

fig, loss_ax = plt.subplots(figsize=(10, 5))
acc_ax = loss_ax.twinx()

loss_ax.plot(curr_model_hist.history['loss'], 'y', label='train loss')
loss_ax.plot(curr_model_hist.history['val_loss'], 'r', label='val loss')
acc_ax.plot(curr_model_hist.history['accuracy'], 'b', label='train acc')
acc_ax.plot(curr_model_hist.history['val_accuracy'], 'g', label='val acc')

loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
acc_ax.set_ylabel('accuray')
loss_ax.legend(loc='upper left')
acc_ax.legend(loc='lower left')

plt.show()
```

학습 결과 그래프로 표현

- 전반적으로 train loss와 train accuracy가 이상적인 방향의 그래프로 표현되었다.
- val_loss 또한 epoch 수가 많아질수록 점점 작게 수렴해나가는 것을 확인할 수 있다.




```
# test data set으로 모델 평가하기
evaluate_metrics = model.evaluate(X_test, Y_test)
print("₩nTest Accuracy = ", "{:.2f}%".format(evaluate_metri
```

```
8/8 [==============================] - 0s 26ms/step - loss: 0.0244 - accuracy: 0.9840

Test Accuracy =  98.40%
Test loss =  0.024416
```

학습 결과

- test accuracy는 98.4%로 높은 정확도를 보였다.

프로젝트 코드 설명

image data processing 및 카테고리 예측

```python
# test data 전처리하기
test_dir = 'C:/Users/ballj/OneDrive/custom_dataset/test'

def load_test_data():
    images = []
    names = []
    size = 64,64
    for image in os.listdir(test_dir):
        temp = cv2.imread(test_dir + '/' + image)
        temp = cv2.resize(temp, size)
        images.append(temp)
        names.append(image)
    images = np.array(images)
    images = images.astype('float32')/255.0
    return images, names

test_images, test_img_names = load_test_data()
```

```python
# 5개 카테고리별 이미지 예측하기
def give_predictions(test_data):
    predictions_classes = []
    for image in test_data:
        image = image.reshape(1,64,64,3)
        pred = model.predict_classes(image)
        predictions_classes.append(pred[0])
    return predictions_classes

predictions = give_predictions(test_images)
```

test data를 전처리하여
image data processing한다.

이를 통해 5개 카테고리별 이미지를 하나씩 test해보고, 이미지가 어떤 카테고리에 해당되는지 예측하도록 한다. 그리고 그 예측한 이미지를 표현하도록 한다.

```python
# 예측한 이미지 표현하기
def get_labels_for_plot(predictions):
    predictions_labels = []
    for i in range(len(predictions)):
        for ins in labels_dict:
            if predictions[i] == labels_dict[ins]:
                predictions_labels.append(ins)
                break
    return predictions_labels

predictions_labels_plot = get_labels_for_plot(predictions)
```

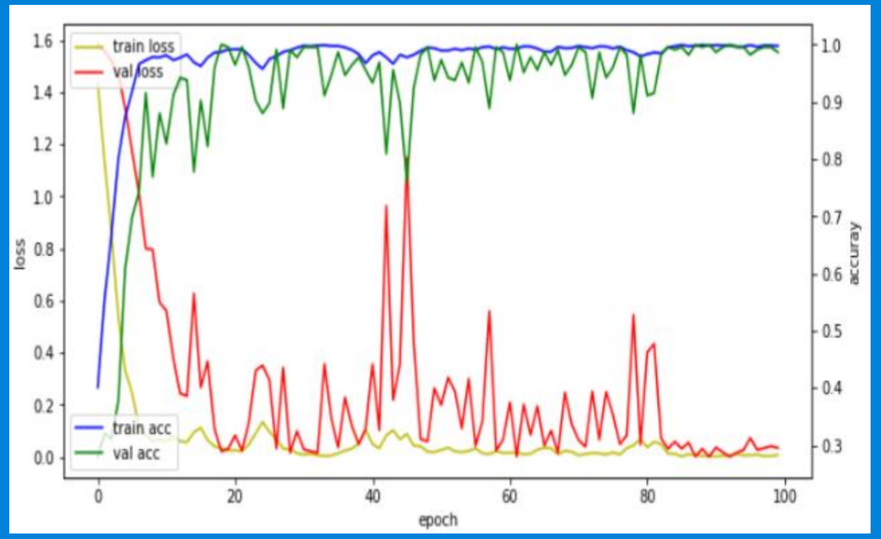
프로젝트 코드 설명

예측한 값 이미지로 표현하기 및 예측값 확인

```
# 예측한 값 이미지로 보여주기
predfigure = plt.figure(figsize = (13,13))
def plot_image_1(fig, image, label, prediction, predictions_label, row, col, index):
    fig.add_subplot(row, col, index)
    plt.axis('off')
    plt.imshow(image)
    title = "prediction : [" + str(predictions_label) + "] "+ "₩n" + label
    plt.title(title)
    return

image_index = 0
row = 2
col = 3
for i in range(1,(row*col)):
    plot_image_1(predfigure, test_images[image_index], test_img_names[image_index], p
    image_index = image_index + 1
plt.show()
```

예측한 값을 이미지로 표현한다.
예시 이미지 데이터들이 학습된 모델에 따라
모두 정확하게 분류되었음을 확인할 수 있다.

실험 결과 정리 및 분석



Test Accuracy = 98.40%
Test loss = 0.024416

MODEL CREATED
Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 64, 64, 16)	448
conv2d_7 (Conv2D)	(None, 64, 64, 32)	4640
max_pooling2d_3 (MaxPooling2	(None, 21, 21, 32)	0
conv2d_8 (Conv2D)	(None, 21, 21, 32)	9248
conv2d_9 (Conv2D)	(None, 21, 21, 64)	18496
max_pooling2d_4 (MaxPooling2	(None, 7, 7, 64)	0
conv2d_10 (Conv2D)	(None, 7, 7, 128)	73856
conv2d_11 (Conv2D)	(None, 7, 7, 256)	295168
max_pooling2d_5 (MaxPooling2	(None, 2, 2, 256)	0
batch_normalization_1 (Batch	(None, 2, 2, 256)	1024
flatten_1 (Flatten)	(None, 1024)	0
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 512)	524800
dense_3 (Dense)	(None, 5)	2565

Total params: 930,245
Trainable params: 929,733
Non-trainable params: 512

1. size를 줄이기 위해 모두 3*3 사이즈의 작은 커널을 사용하였다.
2. Fully connected layer의 weight 수를 최대한 줄이고, feature map을 뽑아내는 곳에 많이 투자하기 위해 convolution layer를 6개의 층으로 구성하였다.
3. 그 결과, test accuracy는 98.4%가 나오게 되었고, 예측을 위한 이미지를 넣었을 때 모두 예측이 옳았다는 결과가 도출되었다.

<최종 결론 및 한계점>

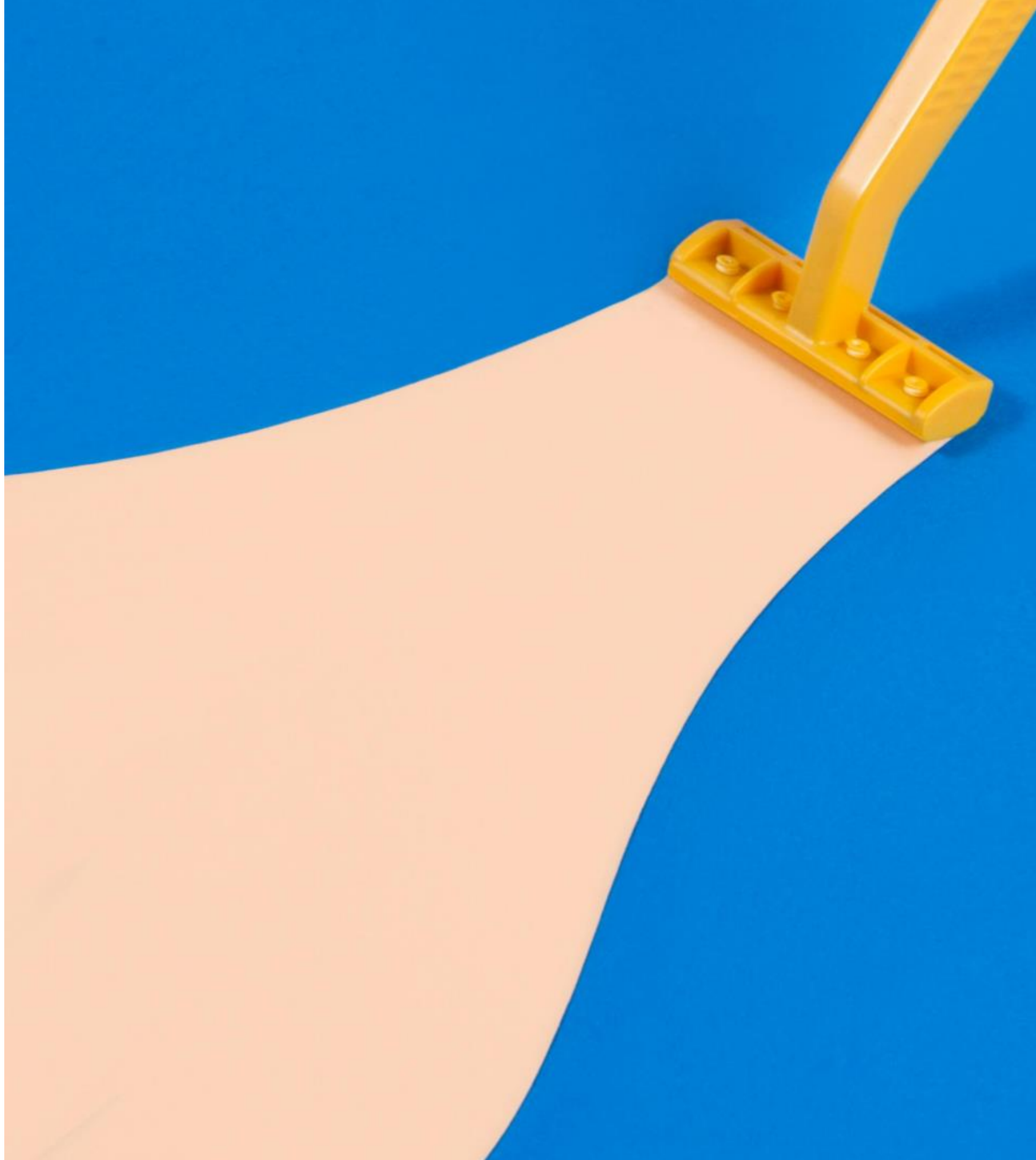
(유명 축구선수 5명의 얼굴 이미지 학습 후 식별)

최종 결론

test accuracy가 98.4%로 나왔고, 학습 과정을 보여주는 그래프의 개형이 전반적으로 이상적으로 나타남.

한계점 - 크롤링 과정에서의 한계점

- 구글 검색 이미지를 크롤링해서 사용하다 보니, 중복되는 이미지 데이터도 많았으며, 후면사진, 타인의 사진 등의 정제되지 않은 데이터들 또한 데이터셋에 포함되었음.



THANK YOU