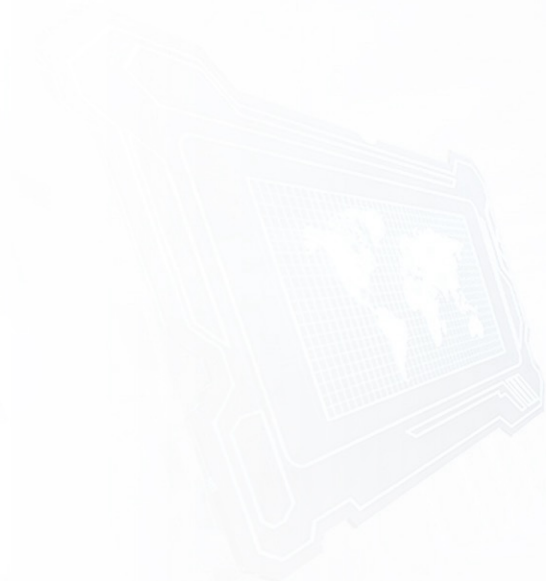


인공지능 세미 프로젝트

Part 5



Contents



1. Keras를 이용한 사진분류 3

Q&A 104

Chapter

1

Keras를 이용한 사진분류

- Python + Google Custom Search API
- Python + OpenCV
- Python + Keras
- Python + Django

←

→

↺

127.0.0.1:8000

앱

MyEtherWallet.com

Free Stock Charts,...

IntelliJ IDEA を使っ...

【5分でわかる】heroku...

동아비즈니스리뷰

Facebook Messeng...

keras를 이용한 딥러닝

좋아하는 연예인을 분류해 보자.

판정할 이미지를 선택해 주세요.

파일 선택

선택된 파일 없음

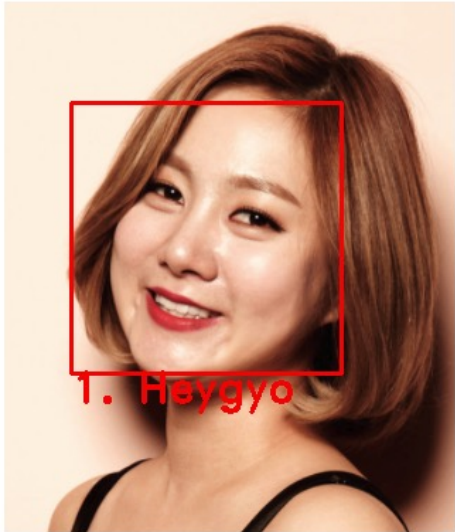
분석

판정 이미지

- predict_2.jpg

판정 결과

1. 전지현일 가능성:0.000% / 송혜교일 가능성:100.000%



1. Heygyo

Python과 Google Custom Search API

이미지 다운로드



Google Search API 사용

- Google Cloud Platform (<https://console.cloud.google.com>)
 - 신규 프로젝트 생성 (or 프로젝트 만들기)
 - API 및 서비스 (or API 개요로 이동)
 - 라이브러리 → Custom Search API 추가
 - 인증 등록 (API Key)

The screenshot displays the Google Cloud Platform console interface. On the left, the 'API 및 서비스' (APIs & Services) section is active, showing a list of services including '대시보드' (Dashboard), '라이브러리' (Library), '사용자 인증 정보' (Credentials), 'OAuth 동의 화면' (OAuth Consent Screen), '도메인 확인' (Domain Verification), and '페이지 사용 동의' (Page Usage Consent). The '사용자 인증 정보' (Credentials) section is highlighted, and a red box around the '+ 사용자 인증 정보 만들기' (Create New Credentials) button is connected by a red arrow to a detailed modal window.

The modal window, titled '사용자 인증 정보' (Credentials), shows the 'API 키' (API Key) creation process. A red box highlights the 'API 키' field, which is currently empty. The text next to it reads: '할당량과 액세스 권한을 확인하기 위해 간단한 API 키로 프로젝트를 확인합니다.' (Use a simple API key to verify your project, so you can check your quota and access permissions.) Other options visible in the modal include 'OAuth 클라이언트 ID' (OAuth Client ID) and '서비스 계정' (Service Account).

Below the modal, the 'API 키' table is visible, showing a list of API keys. The table has columns for '이름' (Name), '생성일' (Created), and '제한사항' (Limits). The first row shows 'API 키 1' created on '2020. 1. 20' with no limits.

이름	생성일	제한사항
API 키 1	2020. 1. 20.	없음

Google Search API 사용

- Custom Search Engine 추가 (<https://cse.google.com/cse/all>)
 - 검색 엔진 추가
 - 검색 할 사이트 등록 → www.google.co.kr
 - 검색 엔진 ID(Search engine ID)
 - 이미지 검색(Image search), 전체 웹 검색(Search the entire web) → ON

Custom Search

New search engine

▼ Edit search engine

All

► Help

Visit Help Forum
(Ask a question)

Send Feedback

Edit search engines

Add Delete

<input type="checkbox"/>	Search engines	Edition	Is owner?	Public URL
<input checked="" type="checkbox"/>	Google	Free	Yes	

환경 설정

```
> python -m venv venv > .\venv\Scripts\activate.ps1
(venv)> python -m pip install --upgrade pip
```

```
(venv)> pip install pylint
(venv)> pip install requests
(venv)> pip install python-dotenv
(venv)> pip install google-api-python-client
```

```
$ conda create -n venv
$ conda activate venv
```

```
(venv)$ conda install pylint
(venv)$ conda install requests
(venv)$ pip install python-dotenv
(venv)$ pip install google-api-python-client
```

- 설정 파일 작성 → settings.env

API_KEY= [REDACTED]

CUSTOM_SEARCH_ENGINE= [REDACTED]

이미지 다운로드 프로그램

- img_download_gcs.py

```
# 지정한 키워드로 검색한 이미지의 URL 얻기
def get_image_urls(keyword, total_num):
```

```
# 이미지 URL을 기본으로 해서 이미지 다운로드
def get_image_files(dir_path, keyword_count, image_urls):
```

```
# 이미지 다운로드
def download_image(url):
```

```
# 이미지를 파일로 저장
def save_image(filename, image):
```

```
# 디렉토리, 디렉토리 내 파일 삭제
def delete_dir(dir_path, is_delete_top_dir=True):
```

```
def main():
    print("=====")
    print("Image Downloader - Google Customr Search API")
    print("지정한 키워드로 검색된 이미지 파일을 다운로드")
    print("=====")
```

이미지 다운로드 프로그램

- settings.py

```
import os
from os.path import join, dirname
from dotenv import load_dotenv

dotenv_path = join(dirname(__file__), 'settings_1.env')
load_dotenv(dotenv_path)

API_KEY = os.environ.get("API_KEY")
CUSTOM_SEARCH_ENGINE = os.environ.get("CUSTOM_SEARCH_ENGINE")
```

- 프로그램 실행

```
(venv)> python img_download_gcs.py "다운로드할 이미지"
```

Python과 OpenCV

이미지파일에서 얼굴검출



얼굴검출 프로그램

- 라이브러리 설치

```
(venv)> pip install python-dotenv
(venv)> pip install opencv-python
```

```
(venv)$ pip install opencv-python
```

- 설정 파일 추가 → settings.env

CASCADE_FILE_PATH="{작업경로}\haarcascades\haarcascade_frontalface_default.xml"

얼굴검출 프로그램

- ***Haar Cascade***

- 머신러닝 기반 오브젝트 검출 알고리즘
- 2001년 논문 "Rapid Object Detection using a Boosted Cascade of Simple Features"에서 Paul Viola와 Michael Jones가 제안한 특징(feature)을 기반으로 비디오 또는 이미지에서 오브젝트를 검출하기 위해 사용되는 알고리즘
- 사각형 영역으로 구성되는 특징을 사용하기 때문에 픽셀을 직접 사용할 때 보다 동작 속도가 빠름

- 1) Haar Feature Selection
- 2) Creating Integral Images
- 3) Adaboost Training
- 4) Cascading Classifiers

얼굴검출 프로그램

- img_face_detect.py

```
def load_name_images(image_path_pattern):
```

```
def detect_image_face(file_path, image, cascade_filepath):
```

```
def delete_dir(dir_path, is_delete_top_dir=True):
```

```
def main():  
    print("=====")  
    print("이미지 얼굴인식 OpenCV 이용")  
    print("지정한 이미지 파일의 정면얼굴을 인식하고, 64x64 사이즈로 변경")  
    print("=====")
```

얼굴검출 프로그램

- settings.py

```
import os
from os.path import join, dirname
from dotenv import load_dotenv

dotenv_path = join(dirname(__file__), 'settings_2.env')
load_dotenv(dotenv_path)

CASCADE_FILE_PATH = os.environ.get("CASCADE_FILE_PATH")
```

- 프로그램 실행

```
(venv)> python img_face_detect.py
```

Python과 OpenCV

이미지파일 수 증가



얼굴검출 프로그램

- img_data_generator.py

```
def load_name_images(image_path_pattern):
```

```
def scratch_image(image, use_flip=True, use_threshold=True, use_filter=True):
```

```
def delete_dir(dir_path, is_delete_top_dir=True):
```

```
def main():  
    print("=====")  
    print("이미지 증가를 위한 OpenCV 이용")  
    print("지정한 이미지 파일의 수를 증가 (Flip, 임계값 등의 작업으로 8배 증가)")  
    print("=====")
```

- 프로그램 실행

```
(venv)> python img_data_generator.py
```

Python과 Keras

얼굴인식 모델 작성



얼굴인식 모델 프로그램

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 64, 64, 32)	896
max_pooling2d_1 (MaxPooling2)	(None, 32, 32, 32)	0
conv2d_2 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_2 (MaxPooling2)	(None, 16, 16, 32)	0
dropout_1 (Dropout)	(None, 16, 16, 32)	0
conv2d_3 (Conv2D)	(None, 16, 16, 64)	18496
max_pooling2d_3 (MaxPooling2)	(None, 8, 8, 64)	0
dropout_2 (Dropout)	(None, 8, 8, 64)	0
flatten_1 (Flatten)	(None, 4096)	0
dense_1 (Dense)	(None, 512)	2097664
dense_2 (Dense)	(None, 128)	65664
dense_3 (Dense)	(None, 2)	258

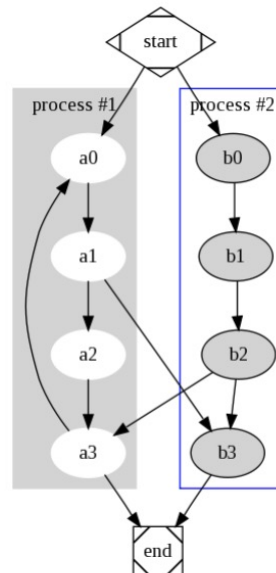
정답은 파일명에

Graphviz

- <http://www.graphviz.org/>

What is Graphviz?

Graphviz is open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. It has important applications in networking, bioinformatics, software engineering, database and web design, machine learning, and in visual interfaces for other technical domains.



Graphviz

- 라이브러리 설치

```
(venv)> pip install tensorflow  
(venv)> pip install keras  
(venv)> pip install graphviz  
(venv)> pip install pydotplus
```

```
(venv)$ pip install tensorflow  
(venv)$ pip install keras  
(venv)$ pip install graphviz  
(venv)$ pip install pydotplus  
(venv)$ pip install matplotlib
```

Graphviz

- img_model_generator.py

```
def load_images(image_directory): ...
```

```
def labeling_images(image_file_list): ...
```

```
def delete_dir(dir_path, is_delete_top_dir=True): ...
```

```
def main():
```

```
    print("=====
```

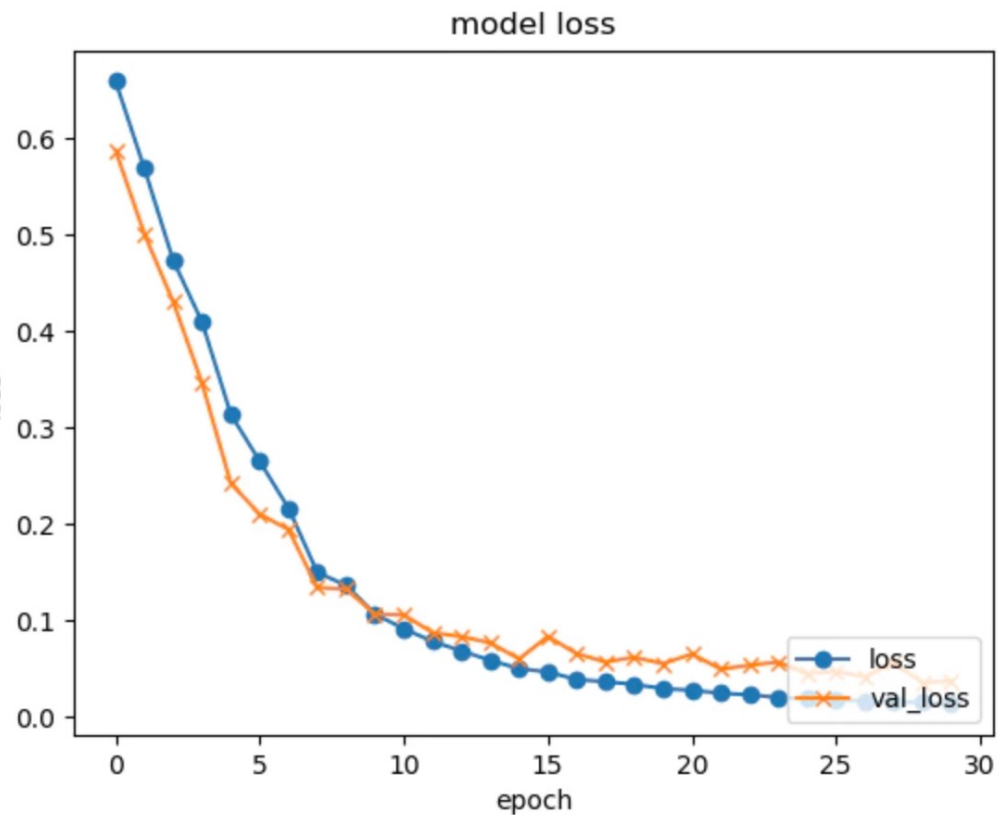
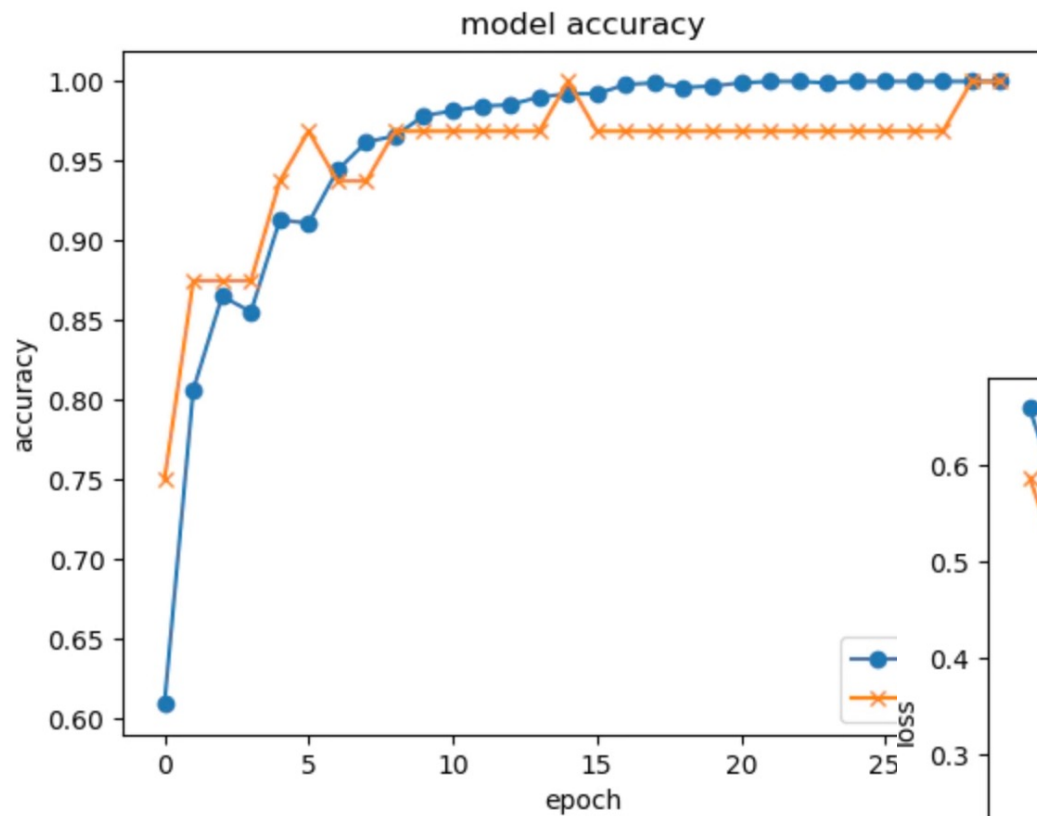
```
    print("Keras를 이용한 모델 학습 ")
```

```
    print("지정한 이미지 파일을 학습하는 모델 생성")
```

```
    print("=====
```

Graphviz

- 모델 학습



Python과 Keras

연예인 얼굴 분류



연예인 얼굴 분류

- 라이브러리 설치

```
(venv)> pip install pylint
(venv)> pip install opencv-python
(venv)> pip install matplotlib
(venv)> pip install tensorflow
(venv)> pip install keras
(venv)> pip install python-dotenv
```

```
(venv)$ pip install pylint
(venv)$ pip install opencv-python
(venv)$ pip install tensorflow
(venv)$ pip install keras
(venv)$ pip install matplotlib
(venv)$ pip install python-dotenv
```

- 설정 파일 추가 → settings.env

CASCADE_FILE_PATH="{작업경로}\haarcascades\haarcascade_frontalface_default.xml"

연예인 얼굴 분류

- img_face_judgement.py

```
def detect_face(model, cascade_filepath, image):
```

```
def detect_who(model, face_image):
```

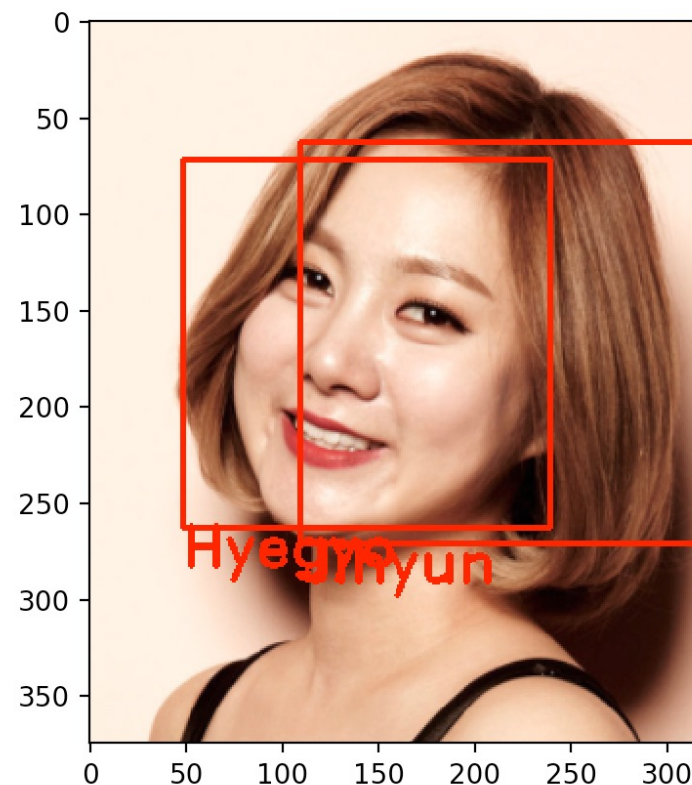
```
def main():  
    print("=====")  
    print("Keras를 이용한 얼굴인식")  
    print("학습 모델과 지정한 이미지 파일을 기본으로 연예인 구분하기")  
    print("=====")
```

연예인 얼굴 분류

- 프로그램 실행

```
(venv)> python img_face_judgement.py c:\temp\test.jpg
```

```
인식 한 얼굴의 수 :2  
인식 한 얼굴의 사이즈 : (191, 191, 3)  
전지현일 가능성 :5.992%  
송혜교일 가능성 :94.008%  
인식 한 얼굴의 사이즈 : (208, 208, 3)  
전지현일 가능성 :91.831%  
송혜교일 가능성 :8.169%
```



Python + Django + Keras

연예인 얼굴인식 App



연예인 얼굴인식 App

- 라이브러리 설치

```
(venv)> pip install pylint
(venv)> pip install opencv-python
(venv)> pip install Django
(venv)> pip install tensorflow
(venv)> pip install keras
(venv)> pip install Pillow
```

- requirements.txt

```
Django~=3.0.5
pylint
opencv-python
Pillow
tensorflow
keras
```

연예인 얼굴인식 App

- Django project 생성

```
(venv)> django-admin startproject mysite .
```

- mysite/settings.py 수정

```
import os

TIME_ZONE = 'Asia/Seoul' # 수정
LANGUAGE_CODE = 'en-us' # 수정
# 추가
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

- Migration

```
(venv)> python manage.py migrate
```

- Web server 실행

```
(venv)> python manage.py runserver
```

연예인 얼굴인식 App

- Django application 생성

```
(venv)> python manage.py startapp pred
```

- mysite/settings.py 수정

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'pred', # 추가  
]
```

연예인 얼굴인식 App

- 프로젝트 루트에 Model 파일과 Haar Cascade 파일 복사

```
img_pred_web
├─mysite
│   ├─settings.py
│   ├─urls.py
│   └─wsgi.py
├─pred
│   ├─migrations
│   ├─admin.py
│   ├─apps.py
│   ├─models.py
│   ├─tests.py
│   ├─urls.py
│   └─views.py
├─venv
├─db.sqlite3
├─haarcascade_frontalface_default.xml ← 복사
├─manage.py
├─model.h5 ← model폴더로 이동
└─requirements.txt
```


연예인 얼굴인식 App

- mysite/settings.py 수정

```
CASCADE_FILE_PATH = os.path.join(BASE_DIR, 'haarcascade_frontalface_default.xml')  
MODEL_FILE_PATH = os.path.join(BASE_DIR, './model/model.h5')
```

- mysite/urls.py 수정

```
from django.contrib import admin  
from django.urls import include, path  
  
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('', include('pred.urls')),  
]
```

- pred/urls.py 수정

```
from django.urls import path  
from .views import PredView  
  
urlpatterns = [  
    path('', PredView.as_view(), name='index'),  
]
```

연예인 얼굴인식 App

- pred/forms.py 생성

```
from django import forms

class ImageForm(forms.Form):
    image = forms.ImageField(label="판정할 이미지를 선택해 주세요.",
                             error_messages={'missing' : '이미지 파일이 선택되지 않았습니다.',
                                                'invalid' : '분류할 이미지 파일을 선택해 주세요.',
                                                'invalid_image' : '이미지 파일이 아닙니다.'})
```

- pred/views.py 수정
- pred/main.py 작성

연예인 얼굴인식 App

- static/css/style.css 작성
- template/pred/base.html 작성
- template/pred/index.html 작성

```
img_pred_web
├─mysite 프로젝트
│  ├──__init__.py
│  ├──settings.py ← 수정
│  ├──urls.py ← 수정
│  └─wsgi.py
├─pred 애플리케이션
│  ├──migrations
│  ├──static
│  │  └─css
│  │     └─style.css ← 생성
│  ├──templates
│  │  └─pred
│  │     ├──base.html ← 생성
│  │     └─index.html ← 생성
│  ├──__init__.py
│  ├──admin.py
│  ├──apps.py
│  ├──forms.py ← 생성
│  ├──main.py ← 생성
│  ├──models.py
│  ├──tests.py
│  ├──urls.py ← 수정
│  └─views.py ← 수정
├─venv
├─db.sqlite3
├─haarcascade_frontalface_default.xml ← 복사
├─manage.py
├─model.h5 ← 복사
└─requirements.txt ← 생성
```

연예인 얼굴인식 App

- 프로그램 실행

