2019 스터디상생플러스 결과 보고서

2019. 5. 30.

나얼팀

목 차

Ⅰ. 연구개요 ····· 3
1. 목적 및 필요성 ····· 3
2. 연구방법 ····· 4
3. (기타 연구개요 내용) ····· 5

Ⅱ. 연구 내용 ····· 6
1. 이미 학습된 openCV모듈을 이용한 얼굴 이미지분석 ····· 6
2. 컨볼루셔널 신경망을 이용한 얼굴 인식 ····· 10
3. 서버 및 클라이언트 설계 ····· 17
4. DB구축 및 설계 ····· 22

Ⅲ. 연구 결론 ····· 24

Ⅳ. 첨부자료 ····· 24
1. 참고문헌 목록 ····· 24
2. 연구 과정 증빙자료 ····· 25

Ⅰ. 제목: 연구 개요

1. 목적 및 필요성

1) 현재 출석 시스템 상황 및 고안 계기

현재 상명대학교의 출결 시스템은 스마트폰, 즉 안드로이드 어플리케이션과 블루투스를 이용해 출석이 진행 되고 있습니다. 이는 과거 호명 출석의 문제점인 출석에 필요한 시간과 대리 출석을 방지 할 수 있게 되었으나 출석처리 후 도망가는 행위 소위 “출튀”를 막을 수는 없었습니다.

이를 해결하는 방법을 고민하던 중 기술적인 결과만을 두고 고려하였을 때 인공지능을 이용한 얼굴인식 출결 시스템을 만든다면 위의 언급한 두가지 문제를 해결 할 수 있을 것으로 생각했습니다.

2) 얼굴인식 출결 시스템의 원리

이 시스템의 원리는 우선 강의실을 전부 촬영할 수 있는 카메라를 강의실 전면의 천장에 설치합니다. 카메라는 강의실의 영상을 실시간으로 촬영하며 해당 영상자료를 서버에 전송합니다. 서버는 받은 영상자료와 학생 사진 데이터셋을 비교하여 얼굴을 인식한 뒤 자동으로 출석을 처리합니다.

이러한 시스템을 이용할 경우 학생들은 출석을 위해서는 강의실에 착석해 있어야 합니다. 또한 자동으로 출석을 하기 때문에 현재 1시간 단위 출석도 큰 불편함 없이 가능하며 중간에 강의를 이탈할 경우 무단결석을 체크할 수 있고, 교수님 또한 출석에 크게 신경쓰지 않고 바로 수업을 진행하실 수 있습니다.

2. 연구방법

1) 연구 진행 단계 및 상태

상태	진행 단계
~ 04/ 16 완료	인공지능 출석 시스템을 만들기 위한 시스템 구상
~ 04/ 30 완료	멘토님께 인공지능에 대하여 배우고 어떤 모델이 적합할지에 대한 논의 및 질의응답
~ 05/ 02 완료	mnist 데이터셋을 가지고 SLP, MLP, CNN을 적용하여 mnist를 학습시켜 모델 선택
~ 05/ 09 완료	얼굴 인식 데이터셋을 어떻게 구축하고 어떻게 가지고 있을 것인지에 대한 회의
~ 05/ 14 완료	서버 - 클라이언트 - DB의 부분으로 나누어 설계를 회의
~ 05/ 21 완료	클라이언트와 서버 설계 및 구축 내용 : 클라이언트는 얼굴인식 결과를 서버로 전송 서버는 얼굴인식 결과를 클라이언트 반응에 맞춰(출석하기 클릭) DB에 저장
~ 05/ 23 완료	DB 테이블 구상 및 구축
5/28 - 진행중	CNN을 이용한 학습으로 이미지를 인식+ 인식 결과를 DB에 저장.
미정.	강의실 카메라 탑재 후 여러 얼굴을 동시에 인식해서 출석이 가능하게 만들기.

2) 연구 구성원 및 역할

구성원	역할
멘토 - 오현우	팀 구성원에게 딥러닝에 대한 기초적 지식을 알려주고 CNN의 기본 특성과 single layer perceptron 및 multi layer perceptron 에 대하여 설명함. 진행 과정 중 이러한 내용을 얼굴 인식 출석부에 어떻게 적용할 수 있을지 조언
팀장 - 정창현	팀원을 관리하고 연구 일정을 계획. 클라이언트 부분을 설계, 서버 - 클라이언트 통신 모델 설계
팀원 - 박현준	CNN을 이용한 얼굴 인식을 연구
팀원 - 임상균	출석내용을 저장하기 위한 DB를 설계
팀원 - 이동석	서버 부분을 설계, 서버 - 클라이언트 통신 모델 설계

3. (기타 연구개요 내용)

(1) openCV

CNN모델 구축에 앞서 클라이언트 - 서버- DB 모델이 제대로 작동 하는지 검증을 하기 위하여 openCV 컴퓨터 비전을 이용하여 CNN 대신에 얼굴 인식 코어으로 사용하였습니다. openCV를 이용하여 사람당 사진 1장만을 이용하여 얼굴을 학습하고, 이에대하여 인식된 얼굴이 누구인지를 반환 해주지만 학습 결과에 대한 정확도를 끌어올릴 수 없다는 점과, 한번에 여러사람이 인식이 불가능하다는 점 때문에 실제 시스템의 코어로 사용하지는 않았습니다.

(2) 서버-클라이언트 모델

서버 클라이언트 모델에서는 통신을 어떻게 할지 구상을 깊게 했습니다. 클라이언트가 서버에게 보낼 정보는 어떤게 있을까? 어떻게 하면 네트워크 트래픽의 혼잡을 줄일 수 있을까? 에 초점을 두어 구상을 했습니다.

Ⅱ. 연구 내용

가. openCV모듈을 이용한 얼굴 이미지 분석

FaceRecog는 사진에서 얼굴을 인식하고 인식된 얼굴이 현재 저장된 값들과 얼마나 비슷한가를 비교합니다. 그 중에서 제일 비슷한 얼굴에 해당하는 학번을 찾습니다. 즉 얼굴 이미지 분석기 라고 할 수 있습니다.

face_recog_server.py 중 class FaceRecog 부분

```
17  class FaceRecog():
18      def __init__(self):
19          # Using OpenCV to capture from device 0. If you have trouble capturing
20          # from a webcam, comment the line below out and use a video file
21          # instead.
22          #self.camera = camera.VideoCamera()
23
24          self.known_face_encodings = []
25          self.known_face_names = []
26
27          # Load sample pictures and learn how to recognize it.
28          dirname = 'knowns'
29          files = os.listdir(dirname)
30          for filename in files:
31              name, ext = os.path.splitext(filename)
32              if ext == '.jpg':
33                  self.known_face_names.append(name[0:9])
34                  pathname = os.path.join(dirname, filename)
35                  img = face_recognition.load_image_file(pathname)
36                  face_encoding =
```

```
    face_recognition.face_encodings(img)[0]
37  self.known_face_encodings.append(face_encoding)
38
39          # Initialize some variables
40          self.face_locations = []
41          self.face_encodings = []
42          self.face_names = []
43          self.process_this_frame =   True
44
45      def __del__(self):
46          del self.camera
47
48      def get_frame(self):
49          try:
50              # Grab a single frame   of video
51              #frame =   self.camera.get_frame()
52              frame =   cv2.imread('download/image.jpg')
53
54              # Resize frame of video    to 1/4 size for
    faster face recognition processing
55              small_frame =    cv2.resize(frame, (0, 0),
    fx=0.25, fy=0.25)
56
57              # Convert the image   from BGR color (which
    OpenCV uses) to RGB color (which face_recognition uses)
58              rgb_small_frame =   small_frame[:, :, ::-1]
59
60              # Only process every   other frame of video
    to save time
61              if   self.process_this_frame:
62                  #  Find  all  the      faces  and  face
    encodings in the current frame of video
63                  self.face_locations                   =
    face_recognition.face_locations(rgb_small_frame)
```

```
                self.face_encodings                    =
64  face_recognition.face_encodings(rgb_small_frame,
    self.face_locations)
65
66              self.face_names =   []
                for     face_encoding               in
67  self.face_encodings:
                    # See if the   face is a match for
68  the known face(s)
                    distances                          =
69  face_recognition.face_distance(self.known_face_encoding
    s, face_encoding)
70                  min_value =   min(distances)
71
                    # tolerance:    How much distance
72  between faces to consider it a match. Lower is more
    strict.
73                  # 0.6 is   typical best performance.
74                  name =   "Unknown"
75                  if min_value   < 0.6:
76                      index =   np.argmin(distances)
                        name                           =
77  self.known_face_names[index]
78                      print('[%s]   로 판별' %name)
                        #response                       =
79  tkinter.messagebox.askyesnocancel("출석", "출석취소")
80
81                        self.face_names.append(name)
82              return name
83          except Exception as e:
84              print(e)
85              name = 'Unknown'
86              return name
87
```

24-37 knowns 디렉토리에서 사진 파일을 읽습니다. 파일 이름으로부터 학번을 추출합니다. face encoding 결과를 known_face_encodings에 추가합니다.

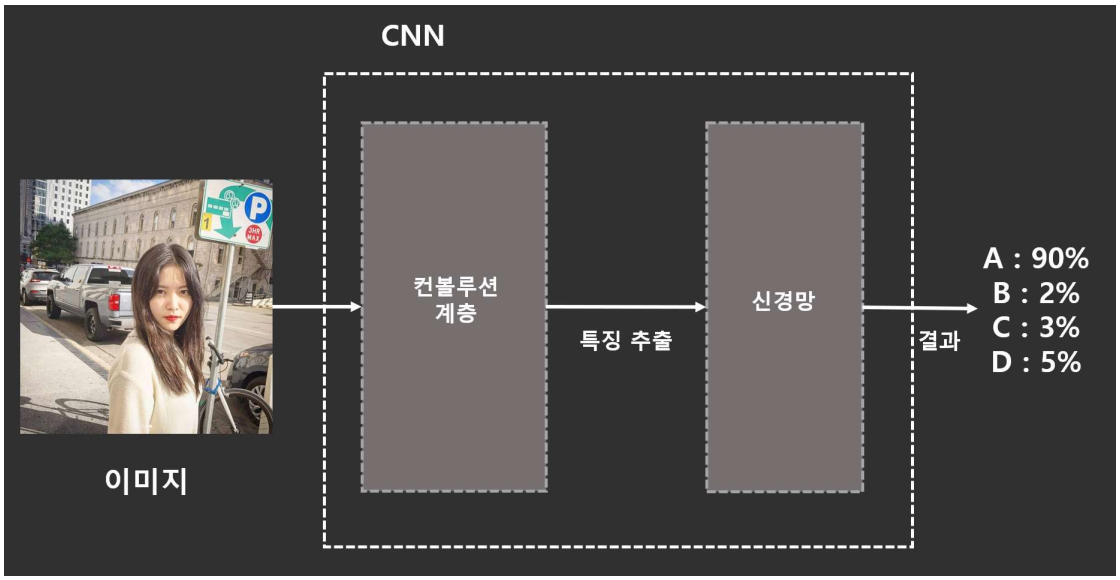
52-55 download/image.jpg로부터 frame을 읽어서 계산양을 줄이기 위해서 1/4 크기로 줄입니다.

66-86 Frame에서 추출한 얼굴 특징과 knowns에 있던 사진 얼굴의 특징을 비교하여, (얼마나 비슷한지) 거리 척도로 환산합니다. 거리(distance)가 가깝울수록 서로 비슷한 얼굴입니다.
거리가 0.6 이면 다른 사람의 얼굴입니다. 이런 경우의 이름은 Unknown입니다 거리가 0.6 이하이고, 최소값을 가진 사람의 학번을 찾습니다.
찾은 사람의 학번을 name에 저장하고 반환 합니다.

나. 컨볼루션 신경망을 이용한 얼굴 인식

1) 컨볼루션 신경망 [CNN]

컨볼루션 신경망은 사람의 신경망에서 착안해 낸 모델입니다. 입력 데이터로부터 특징을 추출하는 역할 주로 이미지와 같은 반복적인 패턴을 가지는 패턴인식에 쓰입니다. Convolution 계층과 Pooling 계층을 번갈아 수행해 가면서 Fully-connected 계층을 구성하고 이를 통해 분류하고 예측을 하는 모델입니다. 이 계층을 이용하여 입, 출력 간 오차를 최소화 하는 “오차역전파” 학습을 반복합니다. 일반적인 신경망에서는 각 픽셀을 가지고 벡터를 표현을 하기 때문에 픽셀 수에 따라 그만한 Computing Power를 요구합니다. 반면에 컨볼루션 첫 번째 층을 구성하는데 컨볼루션 신경망은 이미지 자체를 Feature 벡터로 사용함으로써 상대적으로 연산이 더 효율적이라는 장점을 가지고 있습니다.



[CNN의 구조]

2) 얼굴 인식에 이용한 도구

Tensorflow : 구글에서 공개한 딥러닝과 머신러닝에 활용하기 위한 오픈소스 소프트웨어

Tensorboard : 모델을 시각화하기 위한 도구

Anaconda + Python 3.5 : 코드를 수행하고 적용하기 위한 도구

OpenCV2 : 실시간 이미지 프로세싱에 중점을 둔 라이브러리 (얼굴 인식을 위한 도구)

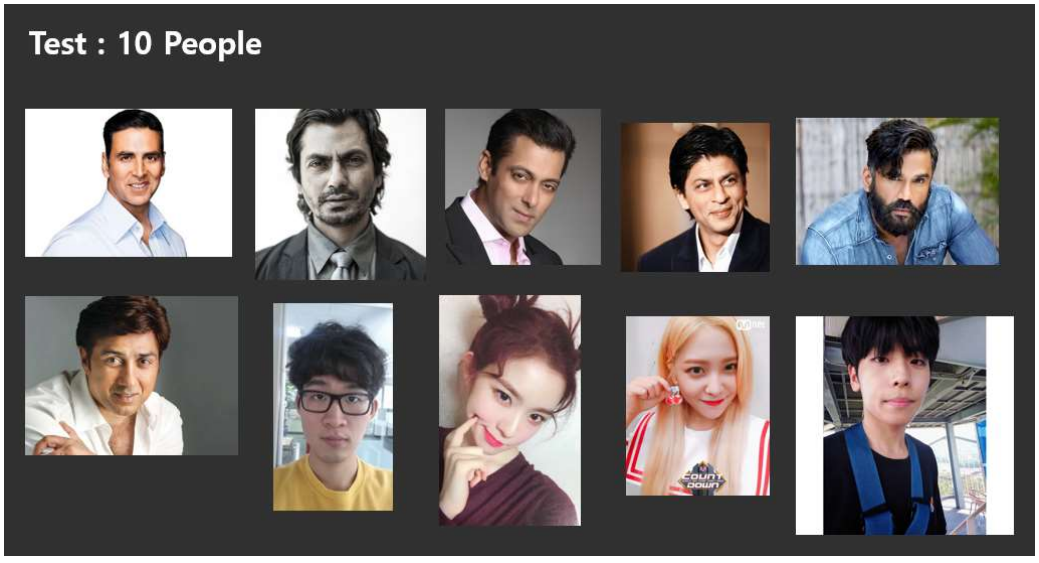
FaceNet : 얼굴 사진을 인식하기 위한 신경망을 구성하기 위한 Training Set

3) 인식을 하기 위한 과정 [파이프라인 프로세스]

데이터 수집 -> 추출 및 가공 -> 데이터 전처리 -> 학습 -> 예측

- 이미지 데이터 수집

예측 정확도를 높이는 데 많은 사진 데이터를 필요로 합니다. 그러나 수많은 사진 데이터를 수집하는 데 어려움이 있어 예측 테스트에 적용하기 위해서 2가지의 방법을 썼습니다. 첫 번째는 FaceNet에서 제공하는 예시 이미지와 얼굴만 추출한 몇 사람의 사진을 이용하였고 두 번째는 직접 긁어오거나 본인의 사진을 이용하였습니다.



[얼굴인식의 대상 10명 - FaceNet 제공 이미지 + 수집한 이미지]

- 이미지에서 얼굴만 추출 및 가공

다음으로 openCV를 이용하여 얼굴만 추출하고 잘라내었습니다. 지면을 위해 일부 주요 코드만 가져왔습니다.

Crop_Face.py 중 주요코드

```
for image in images:
    detector = FaceDetector("haarcascade_frontalface_default.xml")
    faces_coord = detector.detect(image, True)
    faces = normalize_faces(image ,faces_coord)
    for i, face in enumerate(faces):
            cv2.imwrite('%s.jpeg' % (count), faces[i])
            count += 1
```

openCV를 이용하여 사람의 얼굴을 인식하였습니다. 인식 되면 normalize과정을 통해 얼굴만 크롭한 후에 일정한 사이즈로 resize를 하였다. 이후 나온 resize된 이미지는 번호를 붙여 저장을 하였습니다.

[이미지 데이터로부터 얼굴만 크롭 후 32개의 학습시킬 이미지]

- 데이터 전처리

data_preprocess.py

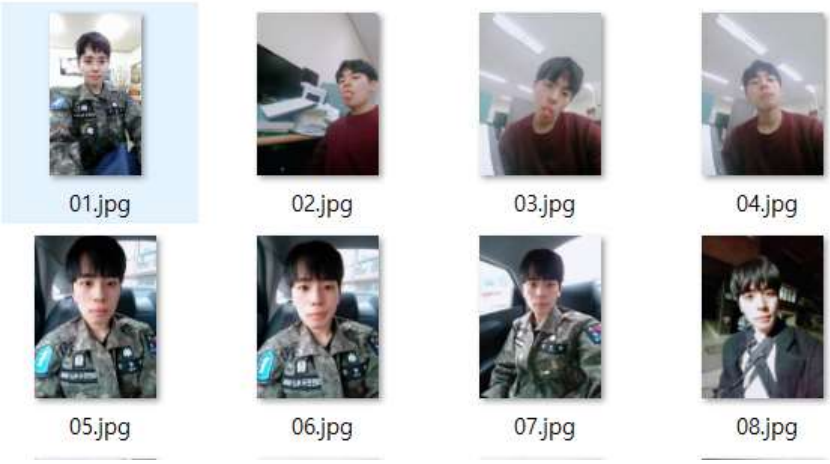
```
from preprocess import preprocesses
import os
import tensorflow as tf

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

old_v = tf.logging.get_verbosity()
tf.logging.set_verbosity(tf.logging.ERROR)

input_datadir = './train_img'  # 전처리 할 이미지를 입력 값으로 설정
output_datadir = './pre_img'  # 가공된 이미지를 출력 값으로 설정

obj=preprocesses(input_datadir, output_datadir) # 입력, 출력 디렉토리 설정, 전처리
nrof_images_total,nrof_successfully_aligned=obj.collect_data()

print('Total number of images: %d' % nrof_images_total)
print('Number of successfully align image: %d' % nrof_successfully_aligned)
```

전처리 할 이미지를 입력 값으로 넣어서 전처리하여 분류를 합니다.

```
./pre_img₩hyunjun₩01.png 352 405 690 869
./pre_img₩hyunjun₩02.png 647 425 887 734
./pre_img₩hyunjun₩03.png
./pre_img₩hyunjun₩04.png 352 423 628 739
./pre_img₩hyunjun₩05.png 311 209 689 710
./pre_img₩hyunjun₩06.png 272 228 664 746
./pre_img₩hyunjun₩07.png 356 270 627 611
./pre_img₩hyunjun₩08.png 297 329 621 761
./pre_img₩hyunjun₩09.png 261 317 590 737
./pre_img₩hyunjun₩10.png 406 325 657 653
./pre_img₩hyunjun₩11.png 335 338 572 657
./pre_img₩hyunjun₩12.png 311 278 648 716
./pre_img₩hyunjun₩13.png 326 239 668 692
./pre_img₩hyunjun₩14.png
./pre_img₩hyunjun₩15.png 254 203 668 709
./pre_img₩hyunjun₩16.png 281 268 762 881
./pre_img₩hyunjun₩17.png 330 339 602 687
./pre_img₩hyunjun₩18.png 358 439 659 842
```

전처리 과정이 끝나고 bouncing box가 생성된 결과입니다. 전처리 과정도 마찬가지로 FaceNet의 MTCNN(Multi-CNN)을 이용하였습니다.

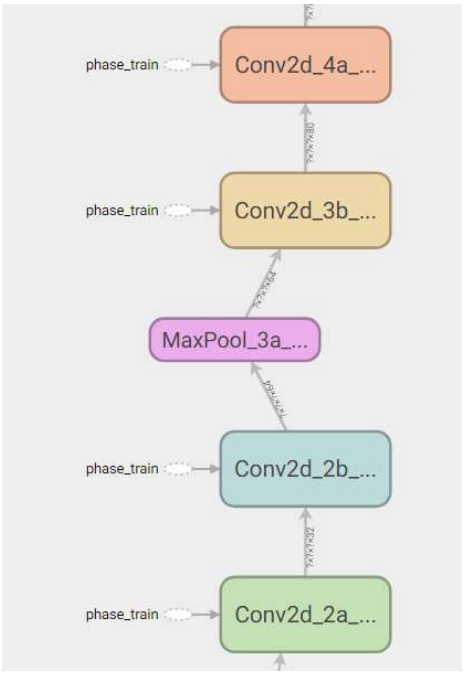
전처리 과정을 거친 후에 새로운 이미지를 이용하여 학습을 진행했습니다.



[학습할 여러 이미지 중 일부]

- 학습

학습 모델은 사전에 정의된 모델을 사용하였습니다. 구글의 FaceNet을 사용하였으며 아키텍쳐로 Inception ResNet v1, WebFace Dataset으로 Pre-trained된 모델을 사용하였다. 모델의 내부는 컨볼루션 계층 연산과 풀링 계층 연산의 반복으로 이루어져 있습니다.

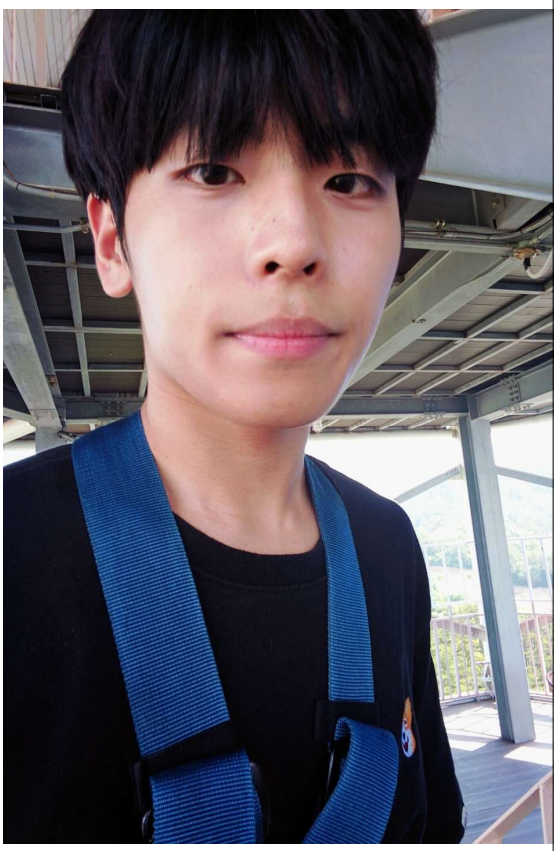


FaceNet 모델의 구조 (일부)

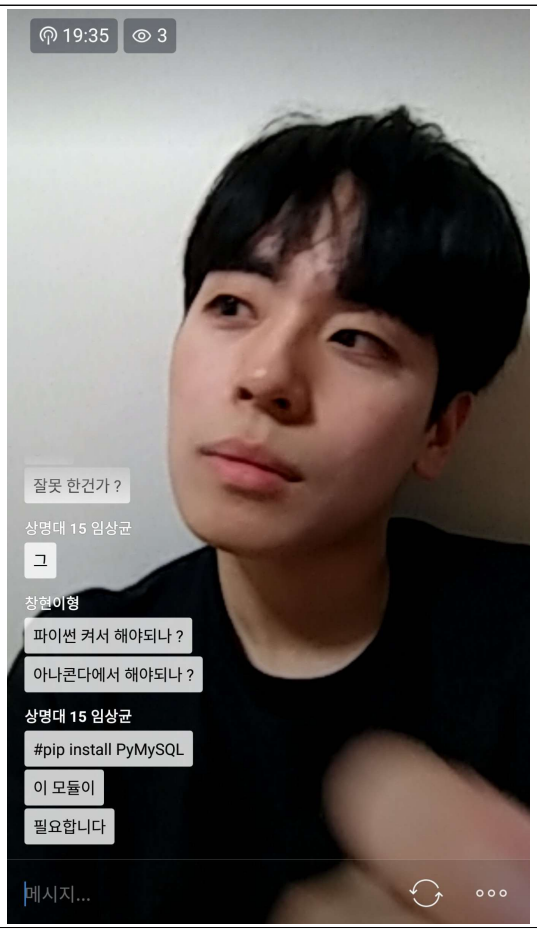
여러 번의 연산 후에 각 사람(대상=객체)에 대하여 최종적으로 저장을 한다. pkl 확장자의 pickle 파일이며 이는 각 객체 정보를 담고 있습니다. (인식할 사람들을 담고 있음) 이는 앞으로 새로운 얼굴 이미지에 대하여 예측하는 데에 이용됩니다.

- 예측

모델과 전처리된 이미지를 이용하여 학습을 하고 예측을 하였습니다. 예측하는 사진은 다음 같은 사람의 두 사진을 이용하였습니다. 한 장은 모바일기기에서 스크린캡처를 하였고 다른 한 장은 직접 촬영을 한 사진입니다.



[테스트 사진 1 - hyunjun.jpg]

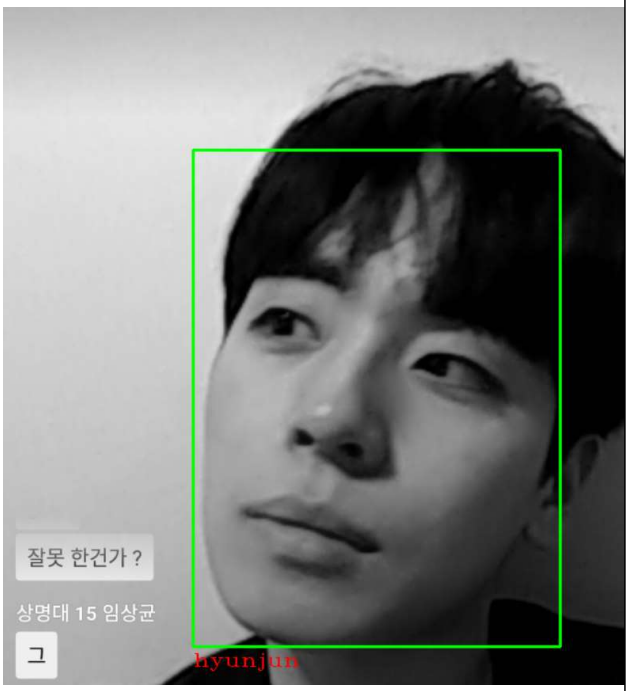


[테스트 사진 2 - hyunjun.png]

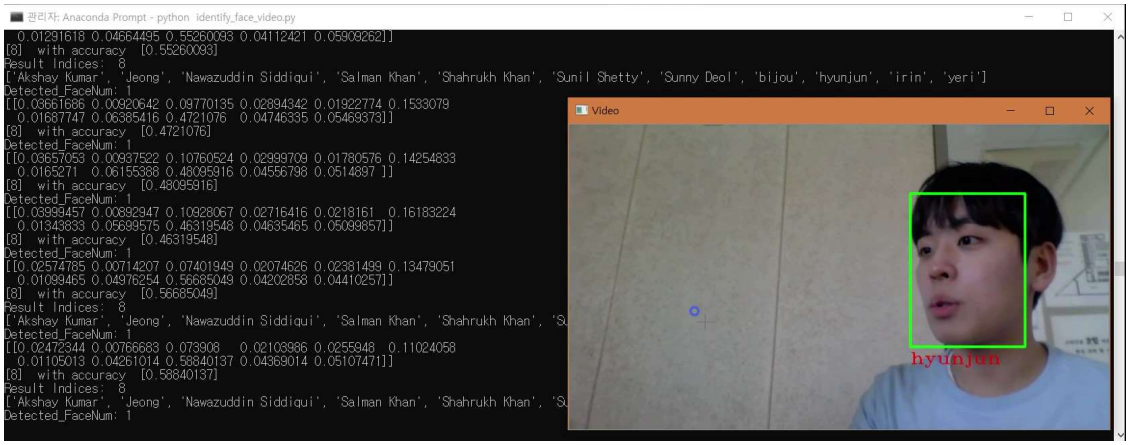
```
Instructions for updating:
Deprecated in favor of operator or tf.math.divide.
Loading feature extraction model
Model filename: ./model/20170511-185253.pb
WARNING:tensorflow:From C:₩myCode₩PHJ₩anaconda₩Facenet-Real-time-face-recognition-using-deep-learning-Tensorflow-master₩facenet.py:378: FastGFile.__init__ (from tensorflow.python.platform.gfile) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.gfile.GFile.
Start Recognition!
Face Detected: 1
[[0.01208639 0.00764806 0.10846825 0.0225165  0.02495475 0.05551878
  0.01613914 0.07025826 0.62720553 0.02540145 0.0298019 ]]
[0.62720553]
Result Indices:  8
['Akshay Kumar', 'Jeong', 'Nawazuddin Siddiqui', 'Salman Khan', 'Shahrukh Khan', 'Sunil Shetty', 'Sunny Deol', 'bijou', 'hyunjun', 'irin', 'yeri']
```

예측 결과 정확도는 높지 않지만 얼굴 인식은 물론 예측 결과도 두 사진 모두 'hyunjun'으로 예측하였습니다.

[예측 결과 1 - hyunjun.jpg]

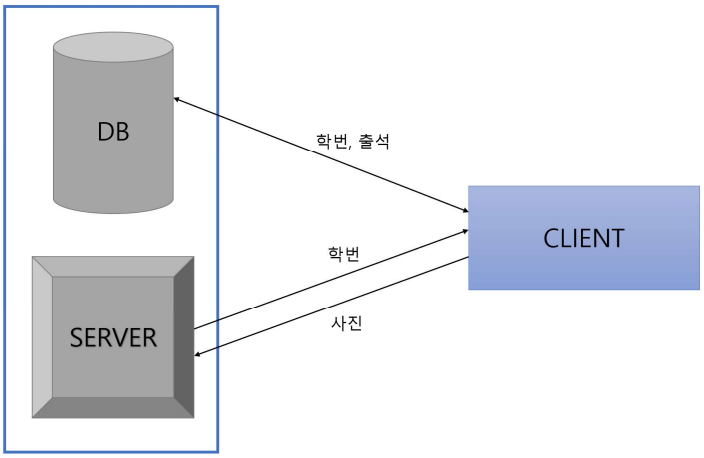


[예측 결과 2 - hyunjun.png]



영상의 경우 실시간으로 인식을 할 수 있음을 알 수 있었습니다.

다. 서버 및 클라이언트 설계



처음 구상에서는 클라이언트가 서버에게 사진을 보내면 학번이 반환되게, 클라이언트가 서버에게 학번을 주면 서버가 DB에 그 인원에 대하여 출석을 하는 쿼리문을 보내고, 클라이언트에 성공 여부를 반환도게 구상했습니다. 이 모델은 클라이언트가 인식한 얼굴을 DB에 저장을 할때는 의미가 있으나 그렇지 않으면 만약 서버나 데이터베이스를 옮길 때 수정해야 하는 부분이 많이 생길 수 있기 때문에 지금의 방법, 클라이언트는 서버에게 사진만 보내주고, 사진속 인물에 해

당하는 학번을 반환받고, 클라이언트가 DB와 쿼리문을 주고받는 형태가 되도록 만들었습니다. 데이터를 전송 할때는 TCP 소켓 통신을 통해 서버와 통신을 합니다.

(1) 서버설계

■ 이 부분은 서버와 클라이언트가 데이터를 주고 받는 것을 묘사하였습니다.

즉, 카메라로 인해 사진이 찍히고, 출석여부를 누르거나 출석정보를 가져오고 자 하는 3가지 경우에 대한 서버의 동작을 보였습니다.

클라이언트는 3가지 종류의 메시지를 보냅니다.

1 학번+강의코드

2 강의코드

3 사진 데이터

1번이 오면 서버는 해당 강의코드 테이블의 학번을 출석으로 합니다.

2번이 오면 서버는 강의코드 테이블의 학생명단을 클라이언트로 보냅니다.

3번이 오면 서버는 해당 사진 데이터와 비슷한 얼굴을 가진 학생의 학번을 클라이언트로 보냅니다.

face_recog_server.py 중 class MyTcphandler 부분 및 Server 및 DB 부분

```
186  class  MyTcpHandler(socketserver.BaseRequestHandler):
187      def handle(self):
188
189          self.request.settimeout(1)
190          data_transferred = 0
191          print('[%s] 연결됨' %  self.client_address[0])
192          data   =          self.request.recv(1024)      # 클라이언트로부터   무언가를 전달받음
193
194          try:
195              message =   data.decode()[0:9]
196              print(message)
```

```
            if(200000000 <= int(message) and
197 int(message) <= 201999999):  # 파일이름 이진 바이트
    스트림 데이터를 일반 문자열로 변환
198                 print('학번 이구나!')
                      findStudentCode(self,   int(message),
199 data.decode()[9:17])
200                 return
201         except Exception as e:
202             print(e)
203
204         try:
205             message =  data.decode()[0:8]
206             print('강의코드구나!')
207               findAttentionsAndSendData(self, message)
208             return
209         except Exception as e:
210             print(e)
211
212         downloadImage(self, data)
213         name =  face_recog.get_frame()
            print('[%s   을   클라이언트로   보내기_코드는
214 여기에]'%name)
215
216         message = name
217           self.request.send(message.encode())
218
219
220 def runServer():
221
222     print('++++++파일 서버를 시작++++++')
        print("+++파일  서버를  끝내려면     'Ctrl  +  C'를
223 누르세요.")
224
225     try:
            server =   socketserver.TCPServer((HOST, PORT),
226 MyTcpHandler)
227         server.socket.settimeout(1)
228         server.serve_forever()
229         #server.handle_timeout()
```

```
230
231       except KeyboardInterrupt:
232           print('++++++파일 서버를   종료합니다.++++++')
233
234   try:
235
236
237       conn =   pymysql.connect(host='localhost',
238                                user='connectuser',
239                                password='connect123',
240                                db='helloworld',
241                                charset='utf8mb4')
242   except Exception:
243       print("Error in MySQL   connexion")
244   else:
245       print("Success to connect   database")
246
247   face_recog = FaceRecog()
248   print(face_recog.known_face_names)
248   runServer()
```

187-191 server의 request timeout을 1로 설정. 연결을 기다리고 연결되면 연결된 클라이언트 주소를 출력함.

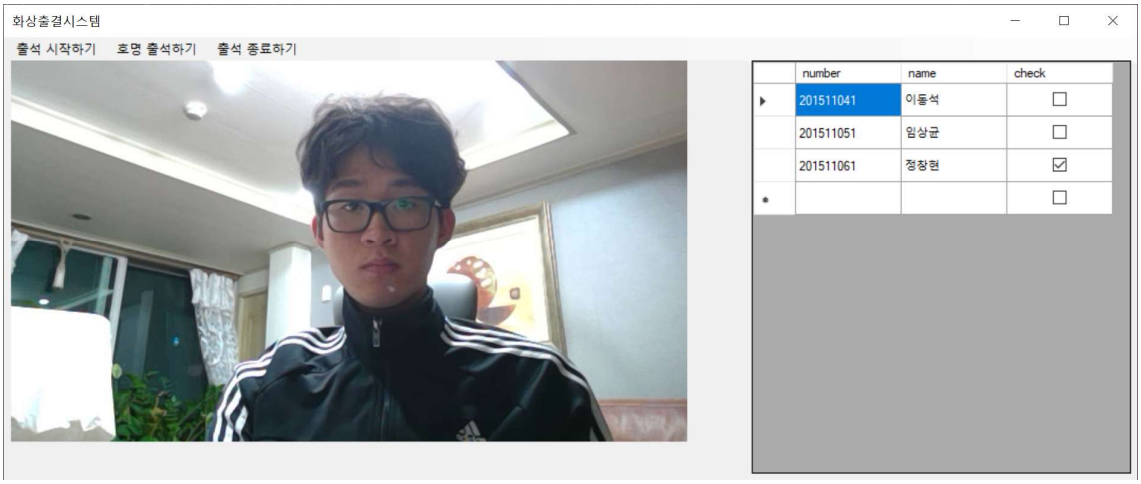
192-200 클라이언트로부터 데이터를 받으면 디코딩을 하고 int 형 변환시 학번에 해당하면 해당학번과 강의코드를 기반으로 테이블에 데이터를 삽입(findStudentCode(self, int(message), decode()[9:17])) 하고 성공 여부를 client로 send.

204-210 앞 코드 부분에서 Exception이 발생하면 이 부분을 수행. data decode에 성공했다면 해당 강의 코드에 해당하는 모든 학생내역을 client로 send.

212-217 위 코드에서 Exception이 났다면 사진 data로 인식. 그 이미지를 다운로드하고(downloadImage(self, data)) face_recog.get_frame()으로 인식결과를 name에 저장한 후 그 학번을 client로 send.

220-232 server를 구동하는 부분
237-245 DB에 연결하는 부분. 성공 실패 여부를 출력한다.
247-249 knowns 에 들어 있는 이미지를 encoding하고 저장된 학번을 출력.
그리고 서버를 구동하는 함수 호출.

(2) 클라이언트 설계



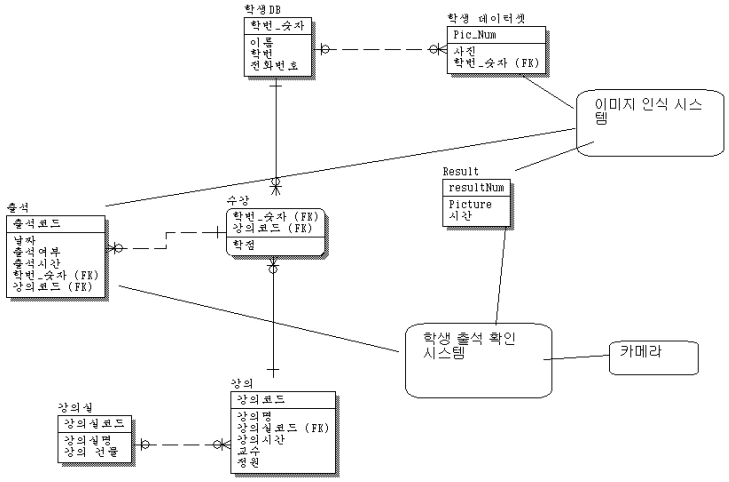
클라이언트에서는 출결 시작 버튼을 누르면 카메라가 켜지게 되고, openCV를 이용하여 사람이 있는지 검출을 하고 만약 사람이 있으면 그때 찍은 사진을 서버에 보내게 됩니다. 그에대해 결과를 받고, 사용자가 그 학번에 대하여 확인을 눌러주게 되면 DB에 출석을 해달라는 요청문을 전송하게 됩니다.

가끔 교수님에 따라 호명으로 출석을 하시는 경우도 있습니다. 이

경우에 대해서 현재 학교에서 수행하는 방법은 학생 한명에 대해 출석을 클릭하면 데이터베이스에 그 학생에 대하여 출석을 요청하게 되는데 이때 1명 호명하는데 걸리는 시간이 T 라고 하고, 쿼리문을 보내고 응답 받는데 걸리는 시간을 RTT , 학생수가 N 명이라고 하면 처음 학생 목록 받아오는데 RTT, 학생 N명에 대해 출석을 하는데 (T+RTT)*N 만큼의 시간이 걸리므로 총 걸리는 시간은 TN + (N+1)RTT 입니다. 하지만, 먼저 출석을 체크를 끝내놓고 한꺼번에 쿼리문을 보내게 된다면 걸리는 시간은 목록 받아오는데 RTT만큼 걸리고, 학생 N명 호명하는데 N*T, 마지막으로 한꺼번에 출석을 요청하는데 걸리는 시간 RTT(파이프라이닝으로 구성)이기 때문에 TN + 2RTT 이므로 (N-1)RTT 만큼 소요 시간이 줄어들게 됩니다. 따라서 호명 출석시에는 모든 학생의 출석에 대한 정보를 출석 종료 버튼을 누르면 한꺼번에 보내는 방식으로 설계를 하였습니다.

라. DB구축 및 설계

(1) Database Diagram



대략적인 테스트를 위한 데이터베이스 다이어그램을 구성하였습니다.

테스트에 필요한 DB항목만 간소화하여 테이블을 작성하였으므로 실제 운용하기 위한 시스템에 적용할 수 없습니다.

(2) Database

DB는 무료버전인 MySQL 8.0 버전을 이용합니다. DB 및 서버는 개인 컴퓨터에 구축하였으며 보안을 위해 포트는 20050번으로 설정하였습니다.

이미지 전송을 위한 소켓통신은 서버포트 20060번으로 설정하였습니다.

- 서버주소 : [IP비공개]:20050
- Database 명 : AI
- 인코딩 : UTF8
- 접속자용 계정 : (“aitester”, “[비밀번호 비공개]”)
- 소켓통신 : [IP비공개]:20060

(3) Database <-> python 연동

얼굴 인식을 위한 CNN은 tensorflow를 이용하며 이를 위해서는 python언어를 사용합니다. 따라서 python언어를 이용하여 DB접속이 가능한가에 대한 여부를 테스트 해야 합니다.

```
# DB연동을 위한 python 코드
import pymysql as sql

pw = input("Password : ")
conn = sql.connect(host='IP', port=20050, user='aitester', password=pw, db='ai', charset='utf8')
cur = conn.cursor()
cur.execute("SELECT * FROM test;")
for row in cur.fetchall() :
    print(row)
conn.close()
```

Ⅲ. 연구 결론

본 연구는 이미지 인식에 최적화 된 컨볼루션 신경망(CNN)을 이용하여 pre-train된 모델을 이용하여 학습을 하였다. 단순히 얼굴을 인식하는 데에만 끝내는 것이 아닌, 예측을 해 보고 이를 실제로 특정 분야에 적용을 해 보았다. 연구에서는 CPU를 사용하였으며 데이터 수집에 한계가 있어 많은 연산을 할 수 없어서 일부만 가져와서 학습하는 Mini-batch 방법을 사용하였다. 시행착오를 겪으며 얼굴을 인식하고 예측하는 데 정확도를 높이는 방법에는 좋은 모델이 필요하는 것을 확인했다. 그 뿐만 아니라 CPU가 아닌 여러 층의 컨볼루션 연산을 빠르게 수행하기 위한 병렬 연산이 가능한 GPU를 사용하고 Dataset을 구축하는 데 최대한 많은 이미지, 그리고 질적으로 우수한 이미지로 구성된 Dataset이 예측 정확도를 높인다는 것을 확인할 수 있었다. 이 외에도 DB연동, 팀원들 간 의사소통도 중요한 요소임을 본 연구를 통해 볼 수 있었다.

Ⅳ. 첨부자료

1. 참고문헌 목록

■ single layer perceptron, multi layer perceptron을 이용하여 mnist 판별하기

https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/tutorials/mnist/beginners/

■ MySQL 테이블의 생성 -

https://opentutorials.org/course/3161/19537

■ openCV를 이용한 얼굴 인식 -

https://ukayzm.github.io/python-face-recognition/

■ 점프 투 파이썬 04-3 파일 읽고 쓰기 -

https://wikidocs.net/26

■ openCV를 이용한 얼굴 검출

https://076923.github.io/posts/C-opencv-29/

■ CNN을 활용한 얼굴 인식

http://krasserm.github.io/2018/02/07/deep-face-recognition/

2. 연구 과정 증빙 자료

연구 결과 깃 허브 주소

https://github.com/ckdgus0505/Naul

활동 보고서 (11개)

스터디_상생플러스_활동보고서_04_16.hwp

스터디_상생플러스_활동보고서_04_23.hwp

스터디_상생플러스_활동보고서_04_30.hwp

스터디_상생플러스_활동보고서_05_02.hwp

스터디_상생플러스_활동보고서_05_07.hwp

스터디_상생플러스_활동보고서_05_09.hwp

스터디_상생플러스_활동보고서_05_14.hwp

스터디_상생플러스_활동보고서_05_17.hwp

스터디_상생플러스_활동보고서_05_21.hwp

스터디_상생플러스_활동보고서_05_23.hwp

스터디_상생플러스_활동보고서_05_28.hwp