

VIP Detection Sensor

딥러닝을 활용한 영상에서의 특정 인물 인식

Agenda

01

주제 소개

주제 선정 이유, VIP Detection Sensor 소개, 프로젝트 일정

02

모델 구현

데이터 수집, 데이터 정제, 모델 훈련, 최적화, 각 연구원 별 모델 4가지

03

시뮬레이션

테스트 영상 구현

04

결론

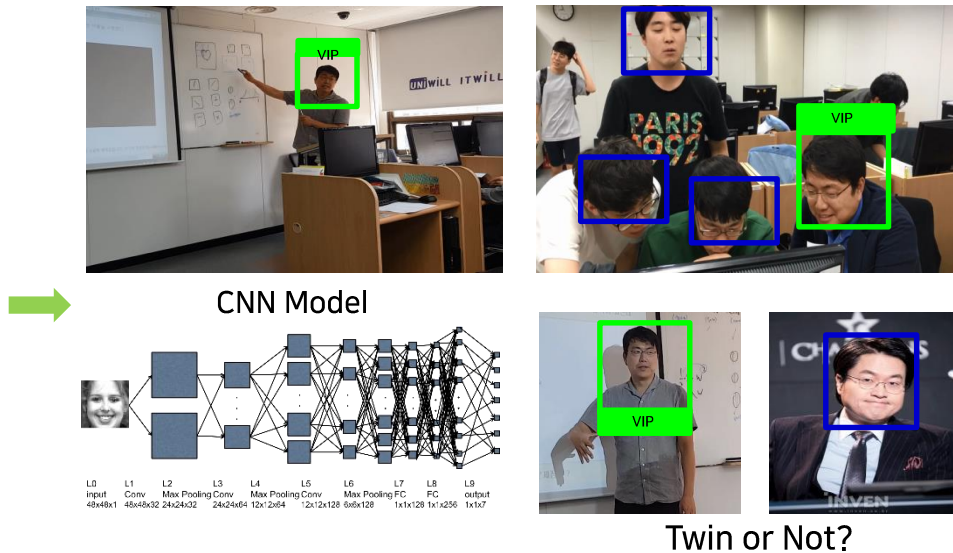
프로젝트 의의, 활용방안

주제 선정

01 주제설명



지난 2017년 Apple의 IPHONE X 가 출시되고, 얼굴인식으로 관리하는 Face ID가 **쌍둥이를 구분해낼 수 있는지에** 대한 이슈가 있었습니다.

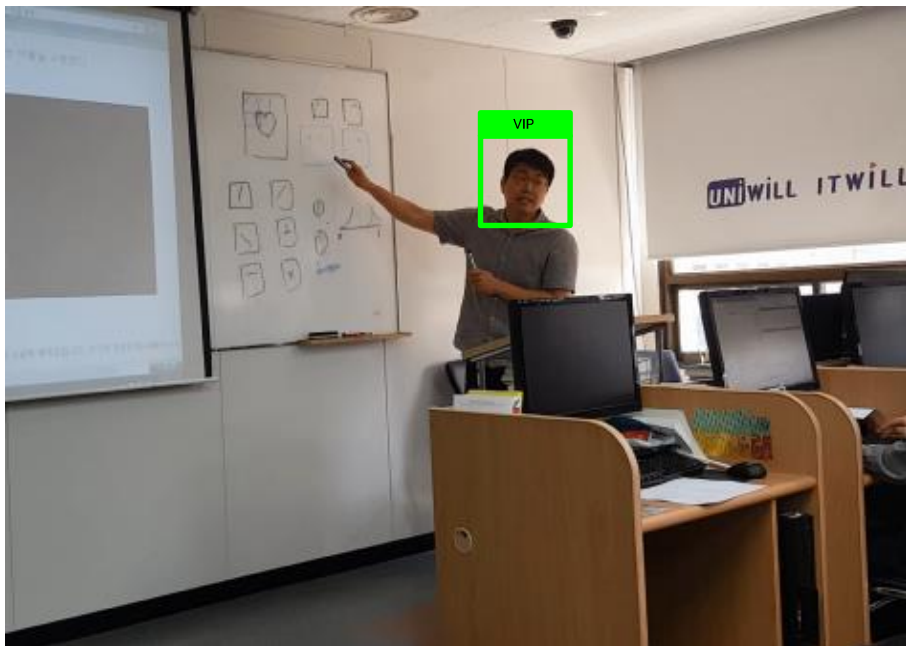


과연 실시간 영상에서 특정인물을 인식하여 다른 사람과 분류하고, Face ID의 쌍둥이 이슈 처럼 유사인물과 **정확히 구분이 가능한지** 구현해 보고 싶었습니다.

VIP Detection Sensor 소개

01 주제설명

VIP Detection Sensor란,
VIP로 라벨링한 특정인물을 다른 사람과 구별하여 영상에서 인식하는 모델을 말합니다.



VIP Detection Sensor 특징

- **실시간** 타겟 감지가 가능합니다. (휴대용 웹캠 이용)
- 인식하고자 하는 **타겟을 변경**하여도 라벨 학습이 가능하도록 모델을 설계하였습니다.
- Face_recognition-OpenCV library를 기반으로 합니다.
: **앞, 옆모습에 상관없이** 얼굴을 잘 인식합니다.
- 모델의 **정확도가 높고**, GPU 기반으로 빠른 처리가 가능합니다.

시간 자원을 적절히 분배하기 위해 프로젝트 일정을 만들어 진행하였습니다.
프로젝트 일정에 대해 최적/최악 데드라인을 설정하여 기한을 지키려 하였습니다.

데이터 수집

- 예상 소요 기간: 1일
- 최적 데드라인 9/4
- 최악 데드라인 9/5

데이터 정제

- 예상 소요 기간: 3일
- 최적 데드라인 9/7
- 최악 데드라인 9/11

모델 설계 및 구현

- 예상 소요 기간: 2일
- 최적 데드라인 9/9
- 최악 데드라인 9/15

최적화

- 마지막까지 계속 진행

PPT 제작 착수

- 예상 소요 기간: 3일
- D-3

모델 활용방안 구상

- D-2

마무리

- D-1

최종 완성

- 9/20(목)

Agenda

01

주제 소개

주제 선정 이유, VIP Detection Sensor 소개, 프로젝트 일정

02

모델 구현

데이터 수집, 데이터 정제, 모델 훈련, 최적화, 각 연구원 별 모델 4가지

03

시뮬레이션

테스트 영상 구현

04

결론

프로젝트 의의, 활용방안

모델 제작 과정

02 모델 구현

데이터 수집 : OpenCV-Face_recognition, Google

데이터 정제 : Python

모델 설계 및 최적화 프레임워크 : Keras



데이터 수집



데이터 정제



모델 설계 및 훈련



모델 최적화

- OpenCV
- Google
- 라벨 이미지(VIP)는 동영상 녹화 후 **OpenCV-Face_recognition**을 사용했으며, 비라벨 이미지는 **Google**의 이미지를 웹 스크롤링 하여 수집하였습니다.

- 자체 제작 코드 (Python) 라벨/비라벨 이미지를 정형화 시키기 위해 모두 **64 x 64**의 크기로 맞춰주고, 비라벨 이미지의 경우 **비정상 파일을 제거**해주는 코드를 Python을 통해 제작하였습니다.

- Keras
모델 설계시 Keras를 기반으로 연구원 4명이 각기 다른 **CNN 모델**을 설계하였습니다. Keras는 딥러닝 프레임워크 중에서 가장 간결하고 직관적이기 때문에 선택하였습니다.

- Keras
Keras 내장 함수인 **evaluate**를 사용하여 최종적으로 모델의 정확성을 평가하였습니다.
또한, 파라미터를 조절하면서 **최적의 모델**을 찾고자 하였습니다.

데이터 수집

02 모델 구현

데이터 수집은 라벨/비라벨로 나누어 진행되었습니다.

라벨 데이터 : VIP 동영상을 촬영 후 수집

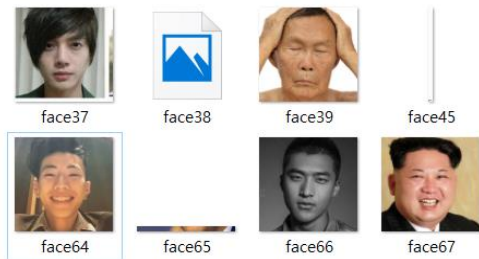
비라벨 데이터 : Google 이미지 웹 스크롤링을 통한 수집

라벨 이미지



라벨 이미지는 **OpenCV-Face_recognition**을 통해 수집되었습니다. 그 결과 손상된 파일은 없지만, 픽셀의 크기가 다른 이미지들의 집합이 형성되었습니다. 따라서 **픽셀 크기를 정제하는** 작업이 요구되었습니다.

비라벨 이미지

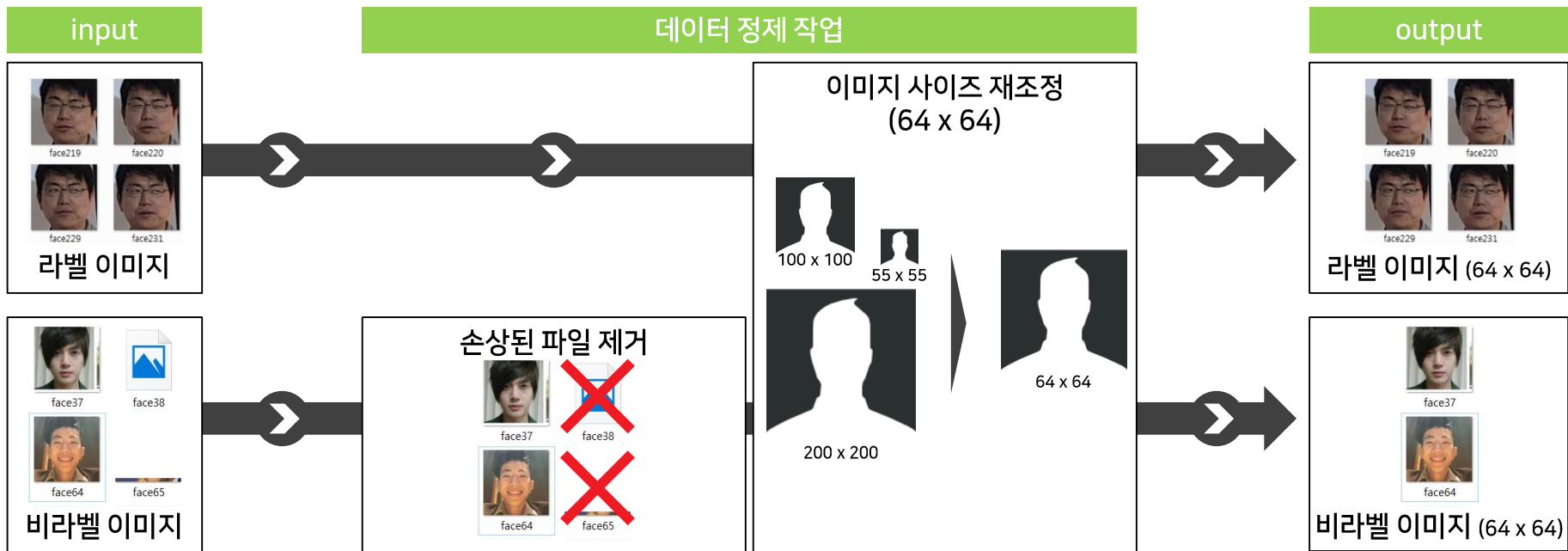


비라벨 이미지는 **Google** 웹스크롤링을 통해 수집되었습니다. 그 결과 손상된 이미지와 정상 이미지가 모두 수집되었고, 또한 픽셀의 크기가 다른 이미지 집합이 형성되었습니다. 따라서 **손상된 파일을 걸러내고 픽셀 사이즈를 통일하는** 작업이 요구되었습니다.

데이터 정제









02 모델 구현

Python 코드를 통해 비라벨 이미지의 손상된 파일을 제거하고, **OpenCV**의 **resize** 기능을 사용해, 라벨/비라벨 이미지의 크기를 일괄적으로 조정하였습니다.



VIP Detection Sensor 를 구현하기 위한 운영환경을 구축하였습니다.

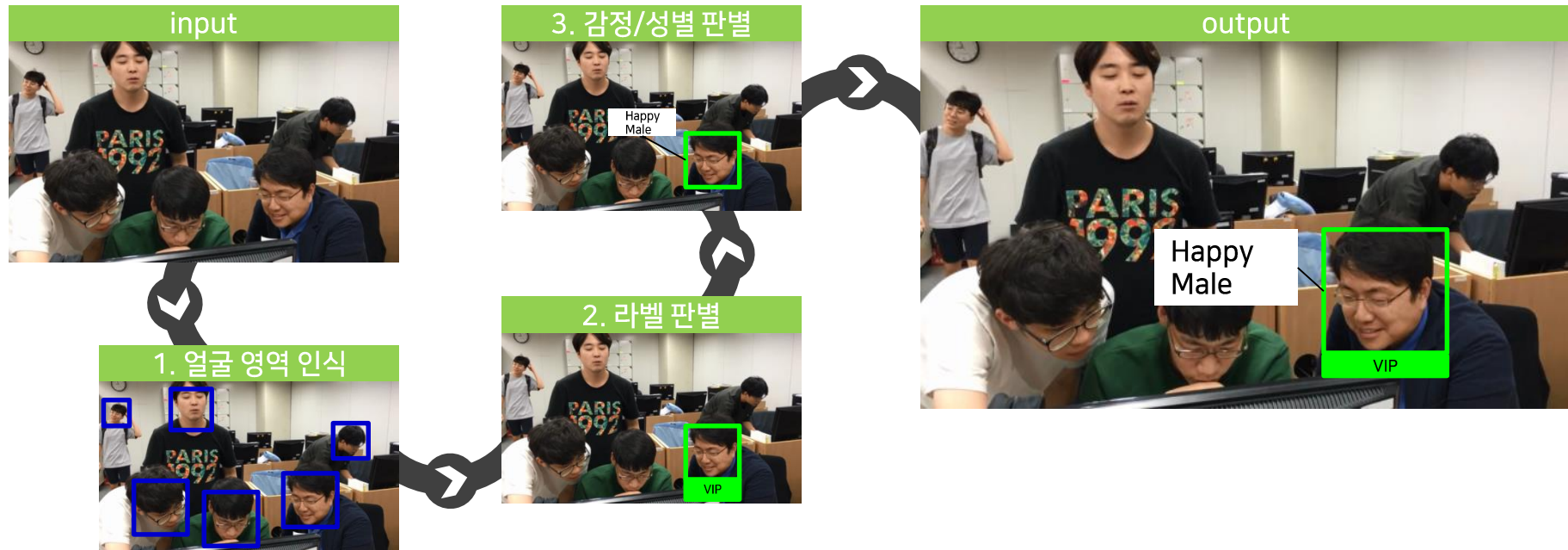
수많은 시행착오와 트러블 슈팅 끝에, **가장 호환이 잘 되는 버전**으로 설치하였습니다.

운영체제	GPU 환경	언어 환경	플랫폼 / 소프트웨어
 Ubuntu 16.04	  NVIDIA CUDA toolkit9.0 	 Python 3.5.6	 Anaconda Python 3.5 ver.  OpenCV 3.43.18  19.15.99

모델 흐름도

02 모델 구현

얼굴 영역 인식 후, **3가지 프로세스**가 동시에 진행됩니다.
먼저 VIP인지 판별하며, 동시에 감정/성별을 판별하는 프로세스가 진행됩니다.

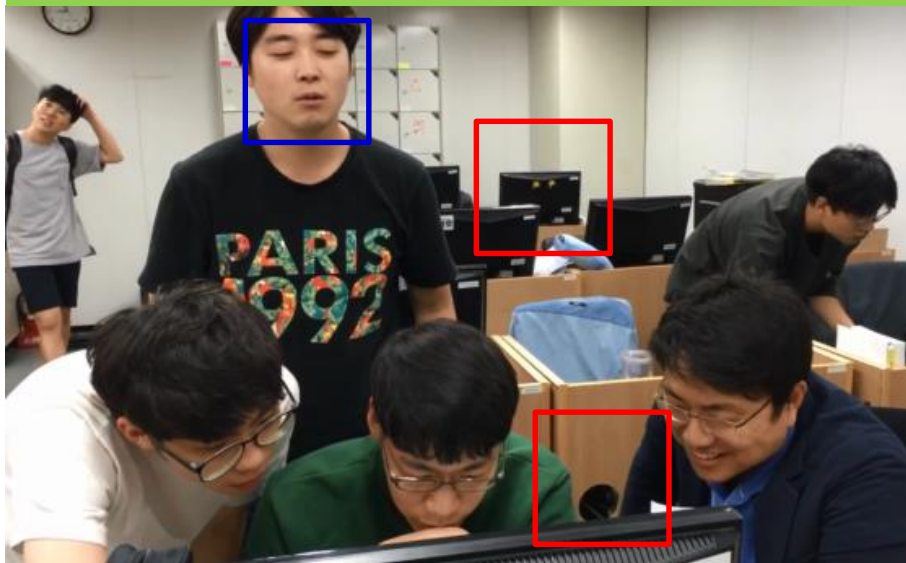


모델 흐름도_1.얼굴영역인식-오류

02 모델 구현

얼굴 영역을 인식하기 위해 **OpenCV**와 **haarcascade frontalface XML** 파일을 사용하였는데, 얼굴의 모양과 비슷한 사물을 인식하고 정면 얼굴만 인식하는 문제를 발견하였습니다.

VIDEO



CODE

```
Import cv2

if __name__ == '__main__':
    cap = cv2.VideoCapture(0) cascade_path
    ="/usr/local/opt/opencv/share/OpenCV/haarcascades/
    haarcascade_frontalface_default.xml"
```

모델 흐름도_1.얼굴영역인식-오류

02 모델 구현

다른 각도의 얼굴 영역을 인식하기 위해 **haarcascade profileface XML** 파일을 추가 사용 하였으나, 얼굴 영역에 **중복 detecting** 되는 또 다른 문제점을 발견하였습니다.

VIDEO



CODE

```
Import cv2

if __name__ == '__main__':
    cap = cv2.VideoCapture(0) cascade_path
    ="/usr/local/opt/opencv/share/OpenCV/haarcascades/
haarcascade_frontalface_default.xml"
    cap = cv2.VideoCapture(0) cascade_path
    ="/usr/local/opt/opencv/share/OpenCV/haarcascades/
haarcascade_profileface_default.xml"
```

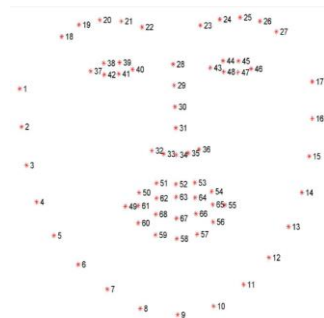
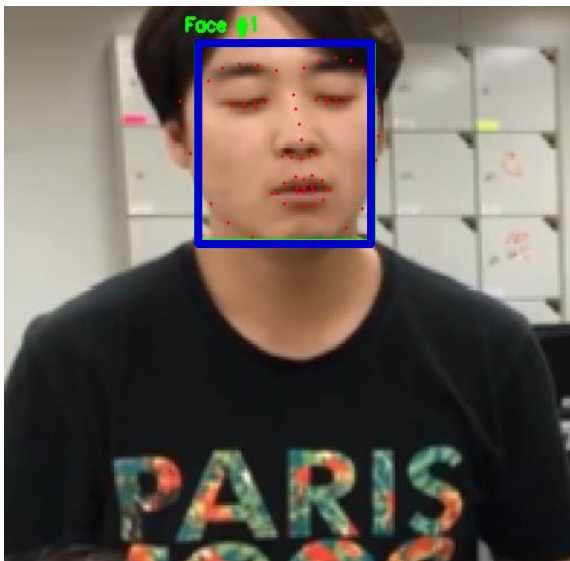
모델 흐름도_1.얼굴영역인식-오류해결

02 모델 구현

얼굴 영역은 **Face_recognition** 라이브러리를 통해 포착하였습니다.

Face_recognition은 얼굴의 **눈썹,눈,코,입,턱**을 **Landmark**하여 얼굴을 인식합니다.

VIDEO



얼굴에는 총 **68개**의
랜드마크가 존재하는데,
어떤 얼굴에서든 랜드마크
크를 찾기 때문에 **어떤
각도에서도 얼굴을 인식
할 수** 있습니다.

CODE

```
rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
rgb = imutils.resize(frame, width=400)
r = frame.shape[1] / float(rgb.shape[1])
```

```
# detect the (x, y)-coordinates of the bounding boxes
# corresponding to each face in the input frame, then
compute
```

```
# the facial embeddings for each face
```

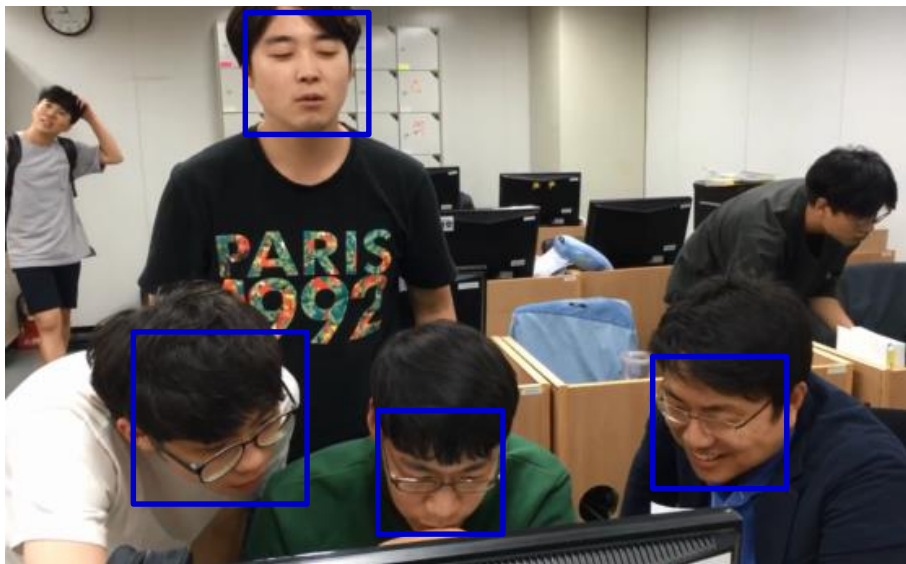
```
boxes = face_recognition.face_locations
      (rgb, model=args["detection_method"])
```

모델 흐름도_1.얼굴영역인식-성능개선

02 모델 구현

Face_recognition 라이브러리를 통해 **98%**이상의 확률로 얼굴 영역을 인식하고 Cropping 되어 저장된 이미지 데이터는 라벨 판별을 위한 프로세스에 활용하였습니다.

FACE RECOGNITION



LABEL DECISION



모델 흐름도_2.성별/감정 판별

02 모델 구현

성별/감정 판별은 오픈소스를 활용하여 적용하였습니다.
이를 통해 얼굴로 인식한 모든 오브젝트에 대하여 **성별**, **감정**이 표시되도록 하였습니다.

VIDEO



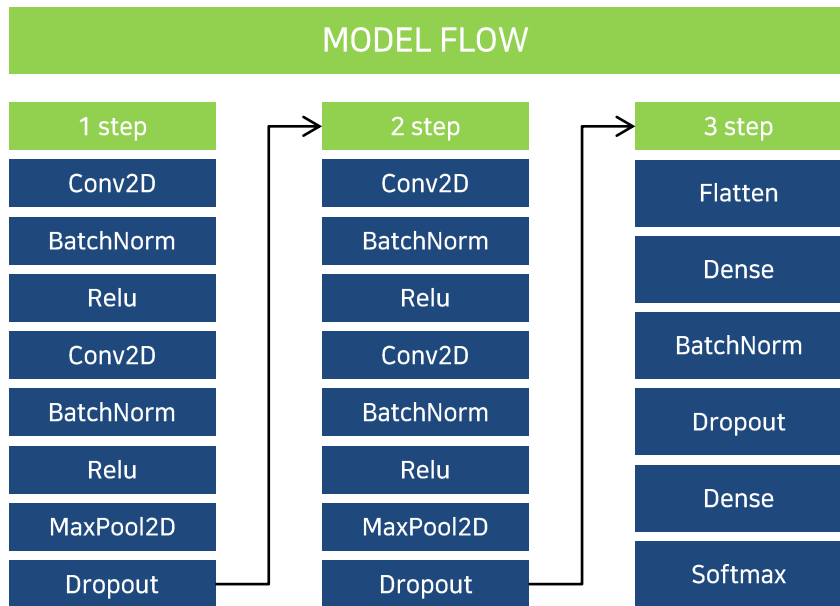
CODE

```
emotion_classifier  
= load_model(emotion_model_path, compile=False)  
  
gender_classifier  
= load_model(gender_model_path, compile=False)  
  
emotion_target_size  
= emotion_classifier.input_shape[1:3]  
  
gender_target_size  
= gender_classifier.input_shape[1:3]  
  
gender_offsets = (30, 60)  
emotion_offsets = (20, 40)
```


CNN 모델 설계를 위해 프레임워크는 **Keras**를 선택하였고, Facebook의 **Deep Face**와 **VGG-Face** 사례를 참고하여 모델을 구상하였습니다.

K Keras

- Keras는 딥러닝 프레임워크 중에서 **가장 간결하고 직관적**이며 사용하기 쉽기 때문에 사용이 편리하였습니다.
- 모듈화가 잘 되어있기 때문에 빠르게 모델을 구현하거나 레이어를 교체할 수 있었습니다.
- 내부적으로 tensorflow를 backend로 두고 있기 때문에, **tensorflow의 기능도 사용**하면서 동시에 간결한 모델을 구성할 수 있었습니다.



모델 구상 및 훈련 요약

02 모델 구현

팀원들이 모델의 학습률을 높이기 위해 Optimizer를 기준으로 각각 모델을 구현하였습니다.
결과적으로 Adam을 사용하였을 때, **94%**라는 가장 높은 정확도를 기록했습니다.

백광흠

- Optimizer : Momentum
- 활성화 함수 : relu
- 배치 정규화 : o
- 가중치 초기값 설정 : he
- 손실률/정확도 : 0.88

김건태

- Optimizer : AdaGrad
- 활성화 함수 : relu
- 배치 정규화 : o
- 가중치 초기값 설정 : he
- 손실률/정확도 : 0.90

차호성

- Optimizer : RMSProp
- 활성화 함수 : relu
- 배치 정규화 : o
- 가중치 초기값 설정 : he
- 손실률/정확도 : 0.91

은해찬

- Optimizer : **Adam**
- 활성화 함수 : relu
- 배치 정규화 : o
- 가중치 초기값 설정 : he
- 손실률/정확도 : **0.94**

Agenda

01

주제 소개

주제 선정 이유, VIP Detection Sensor 소개, 프로젝트 일정

02

모델 구현

데이터 수집, 데이터 정제, 모델 훈련, 최적화, 각 연구원 별 모델 4가지

03

시뮬레이션

테스트 영상 구현

04

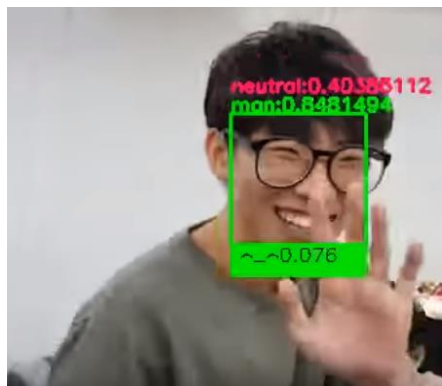
결론

프로젝트 의의, 활용방안

영상 시뮬레이션

03 시뮬레이션

타겟을 VIP(Oracle YU)로 잘 인식하는지 영상을 통해 시뮬레이션 하였습니다.
단독출연 시 라벨 인식의 정확도는 최대 **98%**였고, 공동출연 시 **88%**를 기록했습니다.



단독출연

라벨 인식 정확도 : **98%**

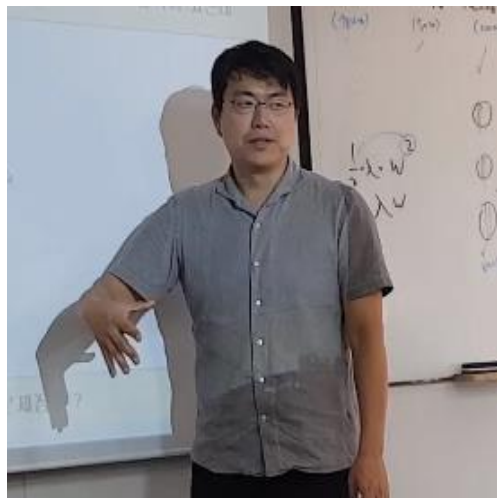
공동출연

라벨 인식 정확도 : **88%**

영상 시뮬레이션

03 시뮬레이션

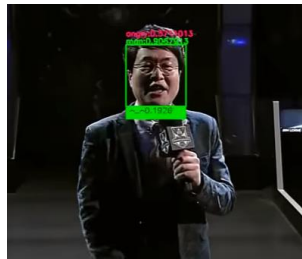
타겟과 유사인물을 대상으로, 라벨 판별 시뮬레이션을 실시하였습니다.
유사인물에 대해 98%의 비율로 라벨이 아니라고 분류하였습니다.



VIP



유사인물



98%의 확률로 비라벨 값으로 분류.

Agenda

01

주제 소개

주제 선정 이유, VIP Detection Sensor 소개, 프로젝트 일정, 자체 평가리스트

02

모델 구현

데이터 수집, 데이터 정제, 모델 훈련, 최적화, 각 연구원 별 모델 4가지

03

시뮬레이션

테스트 영상 시청

04

결론

프로젝트 의의, 활용방안

01

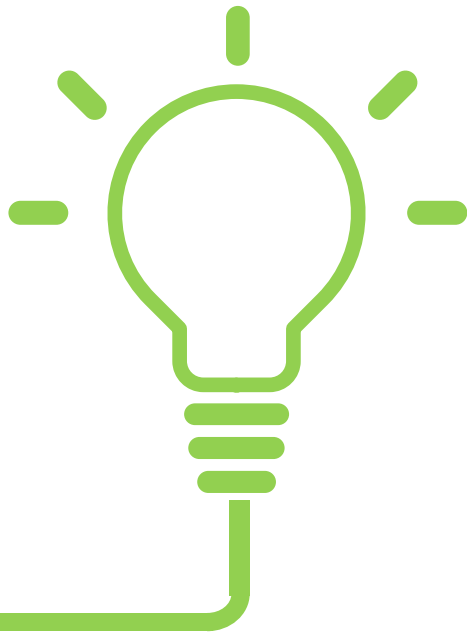
GPU 운영환경 구축
수많은 시도 끝에 최적의 운영환경을 구축하여
GPU 환경에서 모델을 구현하였습니다.

02

학습을 통한 모델 구현
Keras를 기반으로 OpenCV, face-recognition
등의 라이브러리와 Deep face, VGG-face 모델
을 학습하여 모델을 구현하였습니다.

03

오류해결과 성능개선
초기에 발생한 오류를 인지하여 해결하였고, 성능
개선을 통해 98%의 정확도를 구현하였습니다.



VIP Detection Sensor 는 얼굴인식을 기반으로 특정인물을 분류하기 때문에 다른 분야에서도 활용될 수 있습니다.



<얼굴 인식 및 감정 판별>

- 심리 상담 또는 정신 상담 시 **환자의 얼굴을 인식하고 감정 분석**하는데 활용할 수 있습니다.
- 보다 정확한 진단을 위해 **정확도를 높이는 참고 자료**로써 사용될 수 있습니다.



<타겟 위치 알림>

- **미아**를 찾거나 행방 불명된 **취객 찾기** 등에 사용이 가능합니다.
- 얼굴인식을 통한 **출석체크, 출입문 통과** 등에 사용될 수 있습니다.



<백화점 VIP 고객 찾기>

- 백화점 특정 코너를 자주 방문하는 고객일 경우, 얼굴을 인식하여 **맞춤 상품 추천** 합니다.
- VIP 고객 방문 정보를 직원에게 전달하여 **빠른 상황대처**가 가능하게 합니다.

THANK YOU