별지서식3

졸업작품 결과 보고서

제목 : 보안성을 갖춘 Face ID Doorlock

Secure Face ID Doorlock

지도 교수 : 한창희 교수 (서명)

제출일 : 2022년 11월 30일

조장 : 17101278 김석희

조원 : 17101282 김시형

16100804 이승연

16101391 이예빈

서울과학기술대학교 전기정보공학과

목차

1. 서론

1.1 연구 배경

저희는 출입 허가자 가장 방지 및 데이터 보호 기능을 도입한 얼굴 인식 (Face-ID) 도어락을 제작했습니다. 이러한 프로젝트를 진행하게 된 배경에는 크게 두 가지 이유가 있습니다.

첫 번째로 출입 허가자 가장 방지 기능 탑재 배경입니다. 2020년 11월 28일, 삼성 스마트폰 관련 공식 커뮤니티에, 갤럭시 시리즈의 최신 모델의 잠금이 사용자의 사진을 통한 얼굴인식 잠금 해제 가능 여부에 관한 게시글이 올라왔습니다.(그림 1) 이에 대해, 생체인증 담당자는 아직 그들의 기술 현황상, 사진 혹은 닮은 얼굴에 의한 허가자의 의도와 무관한 잠금 해제 가능성을 배제할 수 없다고 답변하고 있습니다.(그림 2)

비슷한 이슈가, 수많은 기종의 핸드폰에서 나타났습니다. 한 네덜란드의 소비자 단체의 검증 테스트 결과에 따르면, 같은 이슈가 테스트에 사용된 38%에 달하는 스마트폰에서 발생되었습니다.(그림 3, 그림 4, 그림 5) 안드로이드의 얼굴 인식 기능의 설명에서도, 다음과 같이 해당 이슈를 인정하고 있습니다. (그림 6)

반면, A사의 아이폰의 경우 적외선 프로젝터를 통해 얼굴의 3D map을 생성, 비교하는 과정을 포함해 사진을 통한 잠금 해제로부터 안전합니다.(그림 7) 또한, 주시 지각 기능을 추가해 눈을 뜨고 기기를 바라보는지 인식하는 기능을 탑재해, 3D map을 악용해 수면 중인 사용자의 얼굴을 인식해 제 3자가 잠금 해제하는 것을 방지합니다.(그림 8)

이처럼, 2D 카메라를 사용하는 Face ID 기능의 주요 취약점인 사진을 통한 잠금 해제, 그리고 수면 중인 사용자의 얼굴을 사용한 인식과 같은 부가 취약점을 고려해, 기존의 2D 카메라 사용 Face ID의 한계를 극복하고자 했습니다. 저희는 특정 행동을 취하면서 얼굴 인식에 응해야 잠금 해제가 가능하도록 제작해, 해당 아이디어를 구현했습니다.

두 번째는, 데이터 보호 기능 탑재 배경입니다. 보안 유지는 모든 기업에서 필수로 해결해야 하며, 가장 민감한 요소 중 하나입니다. 적절하게 보안 장치가 이루어질 경우, 왼쪽 사례처럼 해킹 등으로부터 정보 탈취를 방어할 수 있습니다.(그림 9, 그림 10)

애플의 경우에도, 보안을 강화하기 위해 원격 서버가 아닌 단말기 내에 보안 장치를 마련해, 얼굴 데이터를 저장합니다.(그림 11)

이처럼, 저희는 저장된 얼굴 데이터에 대해 보안 프로토콜을 적용해, 민감한 생체 정보 탈취 최소화를 목표로 프로젝트를 진행했습니다.

그림 1 [1]

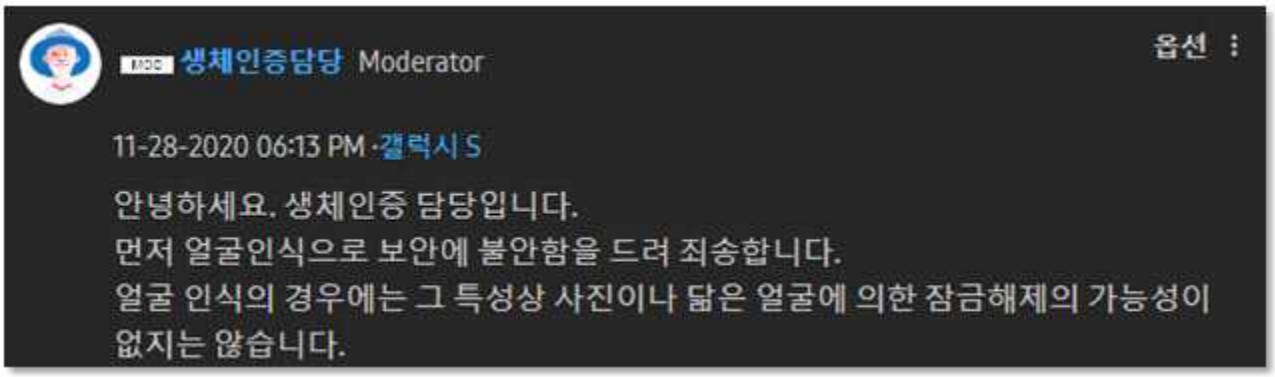


그림 2. 사진 혹은 닮은 얼굴에 대한 생체인증 담당자의 답변이다. 허가자의 의도와 무관한 잠금해제 가능성을 배제할 수 없음을 인정하고 있다.

그림 3 . 스마트론의 얼굴 인식 기능에 의한 단말 보안 유효성에 관한 네덜란드의 소비자 단체의 검증 테스트 결과 [2]

이 소비자 단체 "Consumertenbond" (영어 명칭 : Consumers' League) 보고서에 의하면, 스마트폰 110대를 사용하여 테스트 실시. 실제 소유자의 포트레이트로 얼굴 인식을 실행한 결과 화면 잠금 해제가 된 단말이 42대(38%)에 달했다. 테스트에는 스마트폰으로 자신을 찍은 "셀카"도 사용되었다.

그림 4. 실제 사용자의 얼굴이 아닌 사진을 통해 많은 안드로이드 스마트폰의 잠금 해제가 가능하다는 내용을 담고 있다. 슬로바키아 Security 기업인 ESET 사의 공식 블로그에 업로드 되었다.

포트레이트로 화면 잠금이 해제된 스마트폰은 주로 중저가 기종으로 제조사는 다양하다.
다만, 소니의 "Xperia XZ2 Premium", "Xperia XZ3", 화웨이의 "P20 Pro" 등 몇 종류의 고가 기종에서도 이러한 간단한 방법으로 얼굴 인증을 통과할 수 있었다.

ASUS, 화웨이, 레노버, 모토롤라, 노키아, 삼성, 소니, 샤오미 등의 스마트폰은 이 테스트에서 최소 두 대가 얼굴 인식 잠금이 해제되었다.

그림 5. 그림 4의 결과 테스트에 사용된 38%에 달하는 스마트폰이 소비자의 사진만을 통해 매우 간단하게 잠금 해제 됐다. [2]

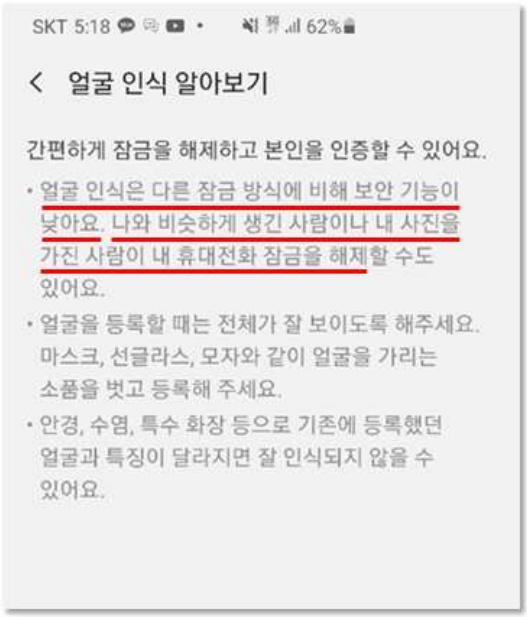


그림 6. 안드로이드 스마트폰에 안내된 Face-ID 관련 사항 [3]

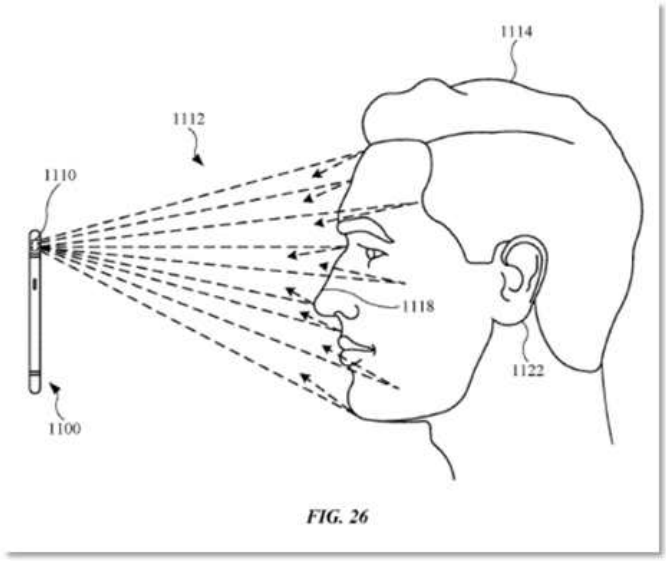


그림 7. 아이폰의 경우 적외선 프로젝터를 통해 얼굴의 3D map을 생성 & 비교하는 과정을 포함하고 있어 사진을 통한 잠금 해제로부터 안전하다.

Face ID는 주시 지각 기능까지 갖추고 있으며, 마스크를 쓴 상태로 Face ID를 사용하는 기능은 화면 주시 여부를 항상 확인합니다. Face ID는 사용자가 눈을 뜨고 기기를 바라보고 있는지 인식합니다. 이를 통해 잠을 자고 있는 경우와 같이 사용자가 모르는 사이에 다른 사람이 기기의 잠금을 해제하지 못하도록 합니다.

그림 8. 아이폰의 경우, 주시 지각 기능을 추가해 사용가자 눈을 뜨고 기기를 바라 보는지 인식하는 기능을 갖춰, 3D map을 통해 수면 중인 사람의 얼굴을 사용해 제 3자가 잠금 해제하는 것을 방지한다. [4]

삼성전자 또 해킹 당해...美 고객정보 유출

| 올해만 두번째...일부 소비자 이름·연락처 등 유출 "보안 강화"

산업일반 | 입력 :2022/09/05 16:53 수정: 2022/09/05 20:57

그림 9. 보안 유지는 모든 기업에서 필수로 해결하는 이슈임. [5]

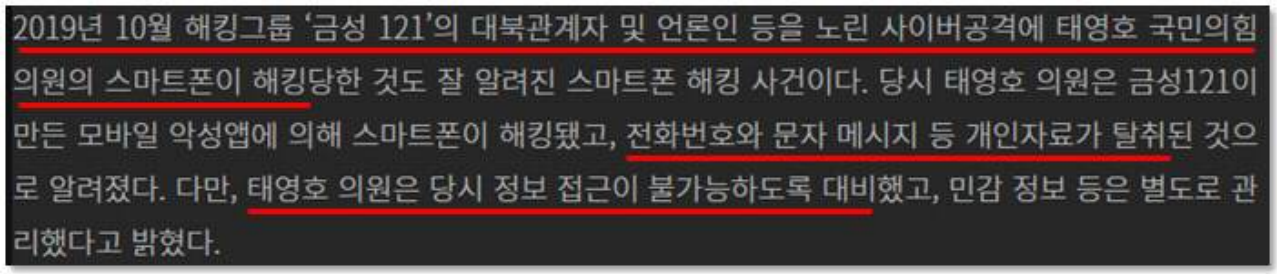


그림 10. 보안 유지 실패 사례2 [6]

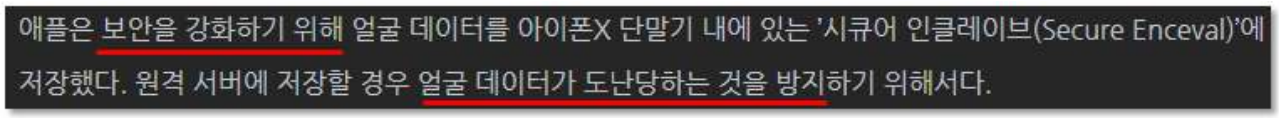


그림 11. 저장된 얼굴 데이터에 대해 보안 프로토콜을 적용, 민감한 생체 정보 탈취 최소화를 목표로 한다.

1.2 연구 목표

전체 시스템 구조는 세 개의 큰 블록으로 나눌 수 있습니다. 사전 얼굴 학습, 출입 시도, 도어락 해제가 그 구성입니다. 여러 방법론을 사용해 출입 허가자 얼굴 학습 데이터 암호화 얼굴 탐지 및 얼굴, 눈, 깜빡임 n회 인식 도어락 해제 및 자동 잠금 데이터 복호화를 구현하였으며, 도어락 해제 및 자동 잠금, 즉 전체 블록도(그림12)의 초록색 상자의 경우 앞서 구현한 소프트웨어 프로세스, 즉 주황색 상자와 연계해 하드웨어로 구현하였습니다.

먼저 얼굴 학습 과정은 크게 두 가지 과정으로 나눌 수 있습니다.

1) Face Detection & Training : Haar Cascade 알고리즘을 통해 이미지에서 얼굴 부분을 탐지하여 Haar feature 값을 추출합니다. 그 후 Adaboost training 알고리즘을 통해 중요 feature 값을 matrix형태로 yml 파일에 저장합니다.
2) 데이터 암호화 : 사용자 등록을 위해 찍은 사진을 AES 알고리즘을 통해 암호화합니다. 이 과정에서 학습에 사용된 이미지들은 암호화되어 내부 저장소에 저장되므로 외부 공격으로부터 보안성을 확보할 수 있습니다.

출입 과정은 다음과 같습니다.

Haarcascade와 LBPH 알고리즘을 통해 기존 등록된 feature값들과의 데이터 비교를 통해 등록된 사용자에 대한 인증을 진행합니다. 라즈베리파이에서 기존 사용자의 데이터와 70% 이상의 정확도를 보이는 user에 대해 눈 깜빡임 인식 과정을 진행합니다. 사용자가 출입을 시도할 때에, 사진으로도 잠금이 해제되는 여러 얼굴인식 도어락의 단점을 극복하기 위해 도입한 눈 깜빡임 인지 기술입니다. 해당 과정이 모두 끝나 인증이 완료되면, 솔레노이드 도어락으로 전기적 신호를 보내 잠금을 해제한 뒤 5초 후 문이 잠깁니다.

요약하면 라즈베리파이 개발환경에서 얼굴인식 프로그램을 구현하여 AES 알고리즘을 통해 보안성을 갖춘 도어락 기술을 구현하는 것이 저희 프로젝트의 목표입니다.

그림 12. 프로젝트 개요

1.3 시스템 설계 사양

품명	규 격	수량	단가	금액
라즈베리파이4 키트		1	199,650 원	199,650 원
카메라 모듈	25mm x 23mm x 9mm	1	32,780 원	32,780 원
아두이노 고급키트		1	42,900 원	42,900 원
아두이노 전자 도어락	149mm x 27mm x 34mm		58,000 원	58,000 원
아두이노 잠금장치	55mm x 23mm x 28mm	1	10,800 원	10,800 원
합계				344,130 원

그림 13. 재료 및 장비

1.4 연구 계획 및 시행

• 3월

- Haar cascade의 다양한 library를 통해 Linux 환경에서 웹캠을 통해 얼굴 인식 모델 구현 예정 : 60% 구현
- 재료 신청(Raspberry Pi 4 등)

• 4월

- Haarcascade library를 활용한 얼굴 인식 모델과 직접 구현한 CNN 모델 비교 (얼굴 인식 속도의 문제로 인해 Haar cascade 모델 채용) : 100 % 구현

• 5월

- LBPH 알고리즘을 통해 feature 값 줄여 연산 속도 확보 : 100% 구현

• 6월

- 1초 안에 얼굴을 판별하여 결과 값을 도출하는 코드 구현 : 100% 구현

• 여름방학 : AES 알고리즘 도입을 통해 data 암호화 및 복호화 시스템 확보 : 40% 구현

• 9월

- AES 알고리즘 issue 해결 (특정 운영체제 내부에서 AES를 통해 암호화하는 코드를 랜섬웨어로 인식하여 암호화 불가) : 100% 구현

• 10월

- 도어락, 문 등 전체적인 H/W 구현 및 시각화 : 100% 구현

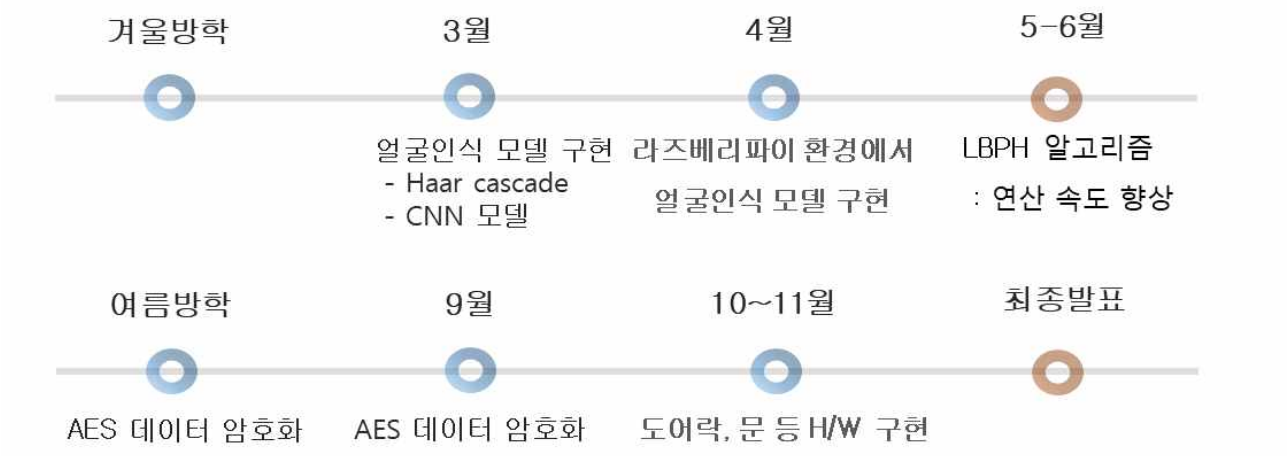


그림 14. 프로젝트 시행 계획

	학번	이름	역할 분담
조장	17101278	김석희	암호화 및 프로그래밍
조원	17101282	김시형	하드웨어 구현/구상 및 얼굴 인식
	16100804	이승연	하드웨어 구현/구상 및 눈 깜빡임
	16101391	이예빈	얼굴 인식 및 눈 깜빡임 인식
공통		공통	PPT템플릿 제작, 발표, 재료 구매

그림 15. 팀원 편성 및 역할 분담

세부계획 및 담당자 \ 연구기간(월)		1	2	3	4	5	6	7	8	9	10	11	비고
자료조사	공통	→	→	→	→	→	→	→	→				
얼굴 탐지/인식	공통		→	→									
눈 깜빡임, 암호화	김석희 이예빈				→	→							
프로그래밍 및 안정화	공통						→	→	→	→	→		
하드웨어 구현	김시형 이승연								→	→	→		
작품 마무리 및 최종발표 준비	공통										→	→	

그림 16. 팀원 별 추진 일정

2. 얼굴인식

2.1 얼굴검출 모델

2.1.1 Haar cascade 알고리즘

Haar Cascade는 머신러닝기반의 오브젝트 검출 알고리즘입니다. 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 오브젝트(물체)를 검출하기 위해 사용됩니다.[7] 직사각형 영역으로 구성되는 특징을 사용하기 때문에 픽셀을 직접 사용할 때 보다 동작 속도가 빠릅니다. 찾으려는 object(얼굴)가 포함된 이미지와 object가 없는 이미지를 사용하여 Haar Cascade Classifier(하르 특징 분류기)를 학습시킵니다. 그 후 분류기를 사용하여 object를 검출합니다.

2.1.2 Haar Cascade의 4단계

1)Haar Feature Selection [8]

첫번째 단계는 이미지에서 하르 특징(Haar Features)을 계산하는 것입니다. 모든 가능한 크기의 커널을 가지고 이미지 전체를 스캔하여 하르 특징을 계산합니다. 예를 들어 24 x 24 크기의 윈도우를 사용시 16,0000개 이상의 Haar Feature를 통해 연산하게 됩니다.

하르 특징은 이미지를 스캔하면서 위치를 이동시키는 인접한 직사각형들의 영역내에 있는 픽셀의 합의 차이를 이용하는 겁니다. 사각 영역 내부의 픽셀들을 빨리 더하기 위해 다음 장에서 소개하는 적분 이미지(integral image)를 사용합니다.

하르 특징에는 다음 세 가지가 있습니다.

1) Edge Features : 두 개의 사각형으로 구성된 하르 특징의 값은 두 사각 영역 내부에 있는 픽셀들의 합하여 검은색 영역의 합에서 흰색 영역의 합을 빼서 구한 것입니다. 두 사각형의 크기와 모양은 동일합니다.

2) Line Features : 세 개의 사각형으로 구성된 Haar 특징은 중앙에 있는 검은색 사각 영역 내부의 픽셀 합에서 바깥에 있는 두 개의 흰색 사각 영역 내부의 픽셀 합을 뺀 것입니다.

3) Four-rectangle Features : 4개의 사각형으로 구성된 하르 특징은 대각선에 위치한 영역 간의 차이를 구합니다.

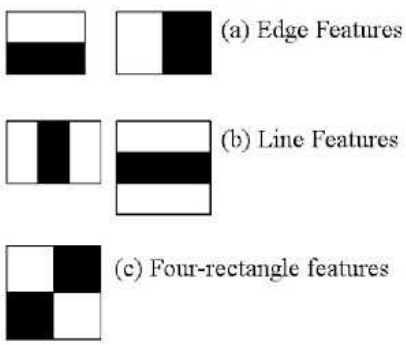


그림 17. Haar feature selection

2) Creating Integral Images

하르 특징을 계산하려면 검은색 사각형과 흰색 사각형 아래에 있는 픽셀의 합을 구해야 합니다. 픽셀의 합을 구하는 과정을 빠르게 실행하기 위해서 적분 이미지(integral image)를 사용합니다.

적분 이미지는 다음처럼 생성합니다. 기존 이미지의 너비와 높이에 1씩 더해서 더 큰 이미지를 만든 후 맨 왼쪽과 맨 위쪽은 0으로 채웁니다. 이후, 기존 이미지에 영역을 지정하여 내부 값들의 합을 구한 후, 적분 이미지에서 원본에 지정한 영역의 오른쪽 아래 픽셀에 대응하는 위치에 합을 입력해줍니다.

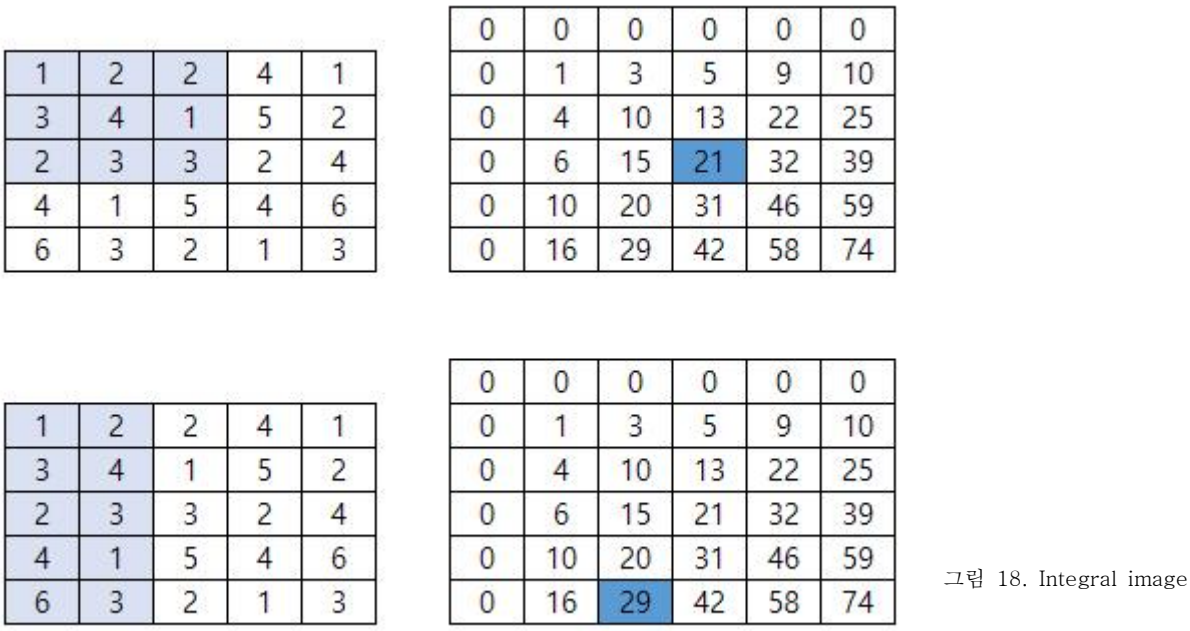


그림 18. Integral image

3)Adaboost training

우선 이미지에서 앞에서 선택한 하르 특징을 사용하여 특징을 계산합니다.

160,000개 이상의 특징이 검출되기에, 유의미한 특징을 선별해야 합니다. 이때 처리 성능을 향상하기 위해 Adaboost training을 사용합니다. 앞에서 구한 특징 중 대부분은 무의미한 특징입니다. 예를 들어 다음 그림(그림19)의 두 가지 하르 특징은 눈 주위에서만 유의미한 특징입니다.

즉, 이 두 개의 하르 특징은 눈 근처에서만 의미 있는 값을 내놓습니다. 가로 방향으로 검은색 사각 영역과 흰색 사각 영역이 있는 특징의 경우에는 코와 빰보다 눈 부분이 더 어둡다는 특성을 사용합니다.

세로 방향으로 흰색 사각 영역이 있고 좌우에 검은색 사각 영역이 있는 특징의 경우에는 중앙에 있는 코보다 양쪽에 있는 눈 부분이 더 어둡다는 특성을 사용합니다.

Adaboost는 다음 과정을 통해 이루어집니다.

최적의 특징을 선택하기 위해 모든 학습 이미지에 특징을 적용합니다. 각 특징에 대해 얼굴이 포함된 이미지와 얼굴이 없는 이미지를 분류하기 위한 최적의 임계값(threshold)을 찾습니다. 이때, 잘못 분류될 가능성이 있기에 에러률이 낮은 특징을 선택해야 합니다.

즉 얼굴이 포함된 이미지와 얼굴이 없는 이미지를 정확하게 분류할 수 있는 특징을 선택합니다. 적용 과정은 다음과 같습니다.

1) 처음에는 모든 하르 특징이 똑같은 가중치를 갖습니다.
2) 모든 하르 특징에 대해 학습 데이터 세트를 사용하여 분류한 결과 각 하르 특징의 에러률을 계산하고 잘못 분류하는 하르 특징의 가중치를 증가시킵니다. 결과적으로 성능 좋은 하르 특징은 낮은 에러률을 갖게 됩니다. 낮은 에러률을 보이는 하르 특징을 선택하게 됩니다.
3) 다음을 만족할 때까지 앞선 과정을 반복합니다.

요구하는 정확성 또는 요구하는 에러률 획득 또는 요구하는 개수의 특징을 발견 최종 분류자는 약 분류자들에 대한 가중치의 합입니다. 개별 특징으로는 이미지를 분류할 수 없어 약 분류자라고 부릅니다.

하지만 약 분류자를 여러 개 묶으면 이미지를 분류할 수 있게 됩니다. Adaboost를 통해 160,000개의 특징은 6,000개의 특징으로 줄어들게 됩니다.

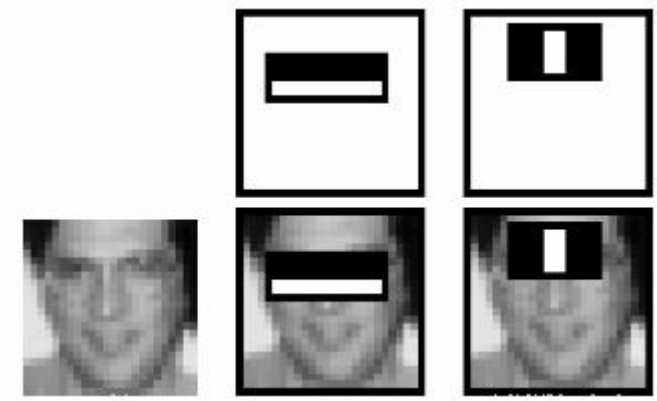


그림 19. AdaBoost training

4)Cascade Classifier

이제 이미지를 준비하여 24 x 24 크기의 윈도우를 사용하여 6,000개의 하르 특징을 적용하여 얼굴을 검출하면 됩니다. 하지만 이렇게 하면 계산량이 너무 많기 때문에 비효율적입니다. 이미지의 대부분의 공간은 얼굴이 없는 영역입니다. 그래서 현재 윈도우가 있는 영역이 얼굴 영역인지를 단계별로 체크하는 방법을 사용합니다. 낮은 단계에서는 짧은 시간에 얼굴 영역인지 판단하게 되며 상위 단계로 갈수록 좀 더 시간이 오래 걸리는 연산을 수행합니다. 이 방식을 Cascade Classifier라고 합니다. 윈도우가 이미지 위를 이동할 때마다 6,000개의 특징을 모두 적용하지 않고 여러 단계의 그룹으로 묶어 사용하는 겁니다. 첫번째 단계의 특징에서 얼굴 영역이라는 판정이 내려지면 현재 윈도우가 위치한 곳에 다음 단계의 특징을 적용합니다.

2.1.3 코드 분석

1)

opencv에서 제공해주는 haar cascade 얼굴인식 알고리즘을 가져와줍니다.

```
face_detector = cv2.CascadeClassifier('C:/Users/home/Desktop/capstone/cascades/data/haarcascade_frontalface_default.xml')
```

그림 20. 얼굴인식 알고리즘 불러오기

2)

얼굴을 감지한후 검출된 얼굴에 사각형을 만들어줍니다.

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_detector.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
    cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
```

그림 21 Image crop 과정

그림 22. Cropped Image

2.2 얼굴인식 모델

2.2.1 LBPH모델 [9]

(1) LBP

Local binary patterns(LBP)는 이미지의 질감(texture) 표현 및 얼굴 인식 등에 활용되는 아주 간단하면서도 효율적인 방법입니다. LBP는 초기에 그림1과 같은 과정을 거쳐 발전해왔습니다.

(2) 원시 LBP (3x3 사각 LBP)

가장 처음에 제안된 LBP의 작동 원리[1]에 대해 살펴보면 LBP 연산자는 말 그대로 지역적인(local) 이진(binary) 패턴(pattern)을 계산하는 방식입니다.

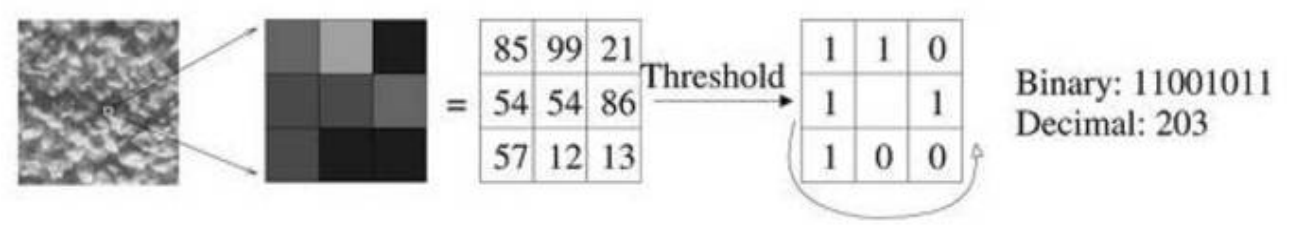


그림 23. Local Binary Patterns(LBP) 의 원리

3x3셀 내에서 중심에 위치하는 픽셀과 이웃하는 8개의 픽셀끼리 서로 크기를 비교합니다. 위 그림(그림 23)의 경우 중심 픽셀의 값이 54이므로 이웃하는 픽셀값이 해당 값보다 크거나 같으면 1로, 이것보다 작으면 0으로 threshold 해줍니다. 순서대로 나열하면 11001011과 같은 이진값을 얻게 되고, 이 값을 십진수 값으로 변환해보면 203으로 같은 값을 얻을 수 있습니다. 가능한 값으로는 0부터 255까지 총 256개의 경우의 수를 가집니다. 모든 픽셀에 대해서 이것을 계산한 후 히스토그램을 작성합니다. 해당 과정은 픽셀값의 히스토그램에 한정되지 않고, LBP 값들의 히스토그램을 만드는 과정입니다. 따라서 총 256개의 bin이 채워져, 하나의 영상의 질감을 256개의 숫자로 표현하게 됩니다.

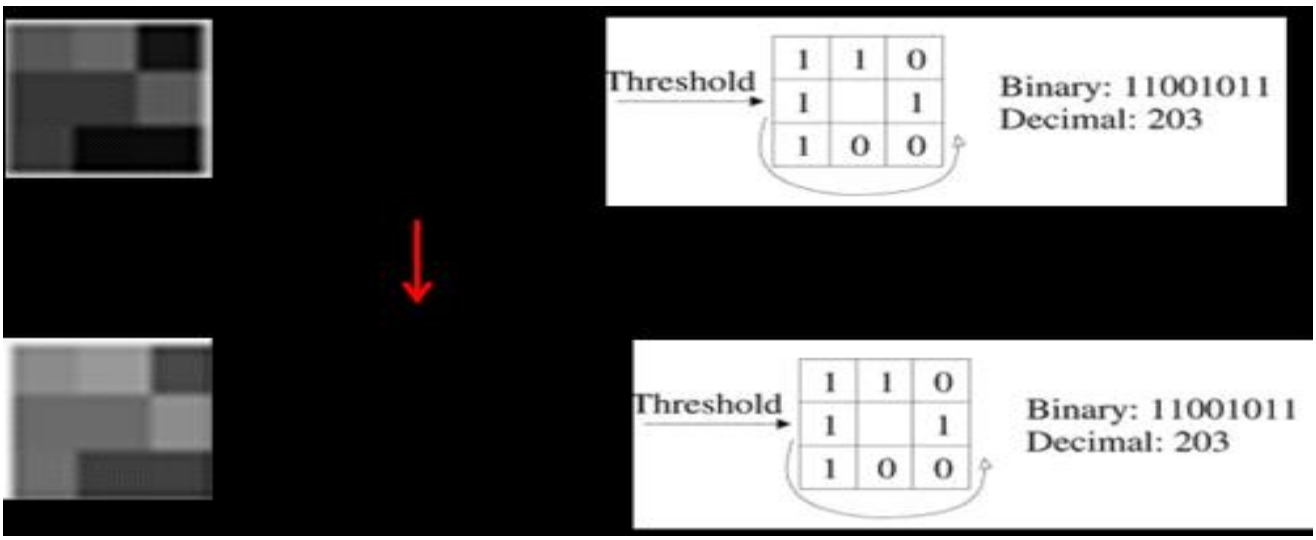


그림 24. 밝기에 덜 의존적인 모델 구현을 위한 LBPH 알고리즘 도입

이러한 원시 LBP는 영상의 밝기가 변해도 robust한 특징을 가지고 있습니다. 위 그림(그림 24)처럼, 단순한 밝기의 변화는 LBP 연산에 영향을 미치지 않습니다.

(3)LBP를 이용해서 얼굴 인식(face recognition)하기

데이터셋에 포함된 사진을 이미지를 여러 개의 블록으로 분할합니다. 각 블록에서 LBP 히스토그램을 구한 값을 연동하여, 전반적인 위치 정보도 유지됩니다.

코드분석)

1)

LBPH 얼굴 학습 모델을 recognizer 변수에 넣어줍니다.

```
recognizer = cv2.face_LBPHFaceRecognizer.create()
```
그림 25. LBPH 알고리즘 사용 코드

2)

얼굴 사진 데이터에서 얻은 이미지 값과 얼굴 id 값을 LBPH 알고리즘을 사용해서 학습시켜줍니다.

```
faces,ids = getImagesAndLabels(path)
recognizer.train(faces, np.array(ids))
```
그림 26. LBPH 알고리즘을 통해 얼굴 학습

3)

학습된 결과값을 바탕으로 trainer라는 파일을 만들어 저장해줍니다.

```
recognizer.write('C:/Users/home/Desktop/capstone/trainer/trainer.yml')
```
그림 27. trainer.yml 파일에 feature matrix 저장

4)

recognizer를 통해 LBPH를 통해 학습된 결과를 저장하는 trainer 파일을 불러옵니다

```
recognizer.read('C:/Users/home/Desktop/capstone/trainer/trainer.yml')
```
그림 28. read 함수를 통해 불러오기

5)

실제로 측정되고 있는 현재 얼굴값과 학습된 결과를 실시간 비교합니다.

일치되고 있는 얼굴값 confidence가 일치하는 비율이 높을 경우 동일인물이라고 인식합니다.

```
id, confidence = recognizer.predict(gray[y:y+h,x:x+w])
```
그림 29. predict 함수를 통해 정확도 예측

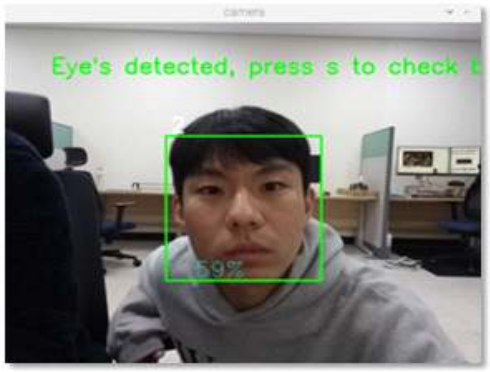


그림 30. LBPH 알고리즘을 통한 얼굴 인식 사진

2.2 얼굴인식모델

2.2.2 CNN

(1) CNN

합성곱 신경망(CNN)은 이미지 인식 분야에서 주로 사용되고 있는 신경망입니다. 생물의 시각 처리 과정을 모방한 것으로, 패턴의 크기와 위치가 바뀌어도 인식할 수 있는 장점이 있습니다.

또한, 이미지를 원시적인 상태로 받음으로써 공간적/지역적 정보를 그대로 유지하며 특성들의 계층을 차근차근 쌓아 올라갈 수 있습니다.

(2) 모델 및 코드 구현

1) 데이터 수집

```
for folder in os.listdir(dataset_folder):
    files = os.listdir(os.path.join(dataset_folder, folder))[:150]
    if len(files) < 50 :
        continue
    for i, name in enumerate(files):
        if name.find(".jpg") > -1 :
            img = cv2.imread(os.path.join(dataset_folder + folder, name))
            img = detect_face(img) # detect face using mtcnn and crop to 100x100
            if img is not None :
                images.append(img)
                names.append(folder)

                print_progress(i, len(files), folder)
```

그림 31. 데이터 수집 코드

위 코드를 이용해 파일 내부에 있는 데이터셋을 불러옵니다.

```
[######################] (150 samples)     label : hyung
[######################] (101 samples)     label : seung
```

그림 32. 데이터 수집 완료

2)image augmentation 진행 [10]

적은 수의 특정 이미지 데이터로 Neural Network를 훈련할 때 과적합 문제가 발생할 수 있습니다. 과적합이란 훈련에 사용되는 이미지에 과도하게 학습되어서 새로운 이미지를 제대로 인식하지 못하는 현상입니다.

Image augmentation은 이미지 인식에 있어서 과적합 문제를 해결하기 위한 매우 간단하면서 강력한 이미지 전처리 기법입니다.

이미지 어그멘테이션 기법을 사용하면 훈련 과정에서 이미지를 회전(rotation) 시키는 등의 변화를 적용합니다. 적은 수의 이미지 데이터셋을 이용해서 이미지 인식의 정확도를 높이고 과적합 문제를 개선할 수 있습니다.

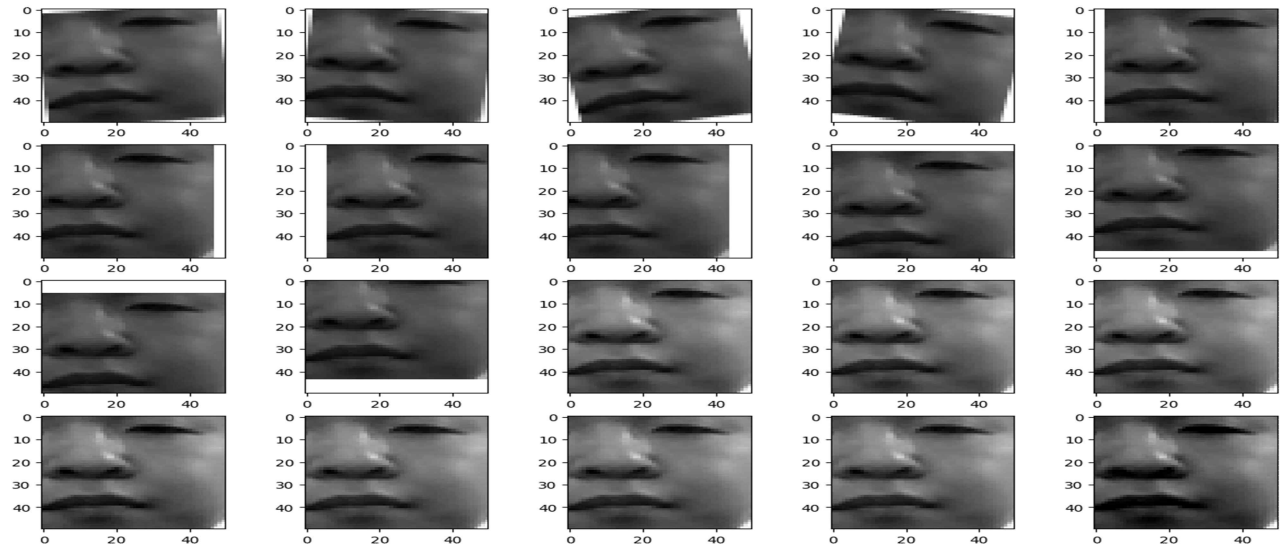


그림 33. Data Augmentation 적용한 이미지 데이터

3)

```
le = LabelEncoder()

le.fit(names)

labels = le.classes_

name_vec = le.transform(names)

categorical_name_vec = to_categorical(name_vec)
```

✓ 0.2s

그림 34. One-hot encoding

학습에 사용되는 데이터셋에 있어, scikit-learn에서 제공하는 머신러닝 알고리즘은 문자열 label을 입력값으로 받을 수 없습니다. 이에, 해당 값을 숫자로 인코딩하는 전처리 작업(Preprocessing)을 본격적인 학습 전에 One-hot Encoding으로 진행했습니다.

```
print(categorical_name_vec)
```

✓ 0.4s

```
[[1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 ...
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]]
```

그림 35. One-hot Encoding 결과

4)cnn 모델 설계

tf.keras.models 모듈의 Sequential 클래스를 사용해서 인공신경망의 각 층을 순서대로 쌓을 수 있습니다. add() 메서드를 이용해서 합성곱 층 Conv2D와 Max pooling 층 MaxPooling2D를 반복해서 구성합니다.

첫번째 Conv2D 층의 첫번째 인자 64는 filters 값입니다. 합성곱 연산에서 사용되는 필터(filter)는 이미지에서 특징 (feature)을 분리해내는 기능을 합니다.

두번째 인자 (3, 3)은 kernel_size 값입니다. kernel_size는 합성곱에 사용되는 필터 (=커널)의 크기입니다.

활성화함수 (Activation function)는 'relu'로 지정하고, 풀링 (Pooling)은 합성곱에 의해 얻어진 Feature map으로부터 값을 샘플링해서 정보를 압축하는 과정을 의미합니다.

맥스풀링 (Max-pooling)은 특정 영역에서 가장 큰 값을 샘플링하는 풀링 방식이며, 풀링 필터의 크기를 2×2 영역으로 설정했습니다.

```
model.add(Conv2D(64,
                 (3,3),
                 padding="valid",
                 activation="relu",
                 input_shape=input_shape))
model.add(Conv2D(64,
                 (3,3),
                 padding="valid",
                 activation="relu",
                 input_shape=input_shape))

model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Conv2D(128,
                 (3,3),
                 padding="valid",
                 activation="relu"))
model.add(Conv2D(128,
                 (3,3),
                 padding="valid",
                 activation="relu"))
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(128, activation="relu"))
model.add(Dense(64, activation="relu"))
model.add(Dense(len(labels)))  # equal to number of classes
model.add(Activation("softmax"))
```

그림 36. CNN을 활용한 모델 구현

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 64)	640
conv2d_1 (Conv2D)	(None, 46, 46, 64)	36928
max_pooling2d (MaxPooling2D)	(None, 23, 23, 64)	0
conv2d_2 (Conv2D)	(None, 21, 21, 128)	73856
conv2d_3 (Conv2D)	(None, 19, 19, 128)	147584
max_pooling2d_1 (MaxPooling2D)	(None, 9, 9, 128)	0
flatten (Flatten)	(None, 10368)	0
dense (Dense)	(None, 128)	1327232
dense_1 (Dense)	(None, 64)	8256
dense_2 (Dense)	(None, 3)	195

...

그림 37. CNN 모델의 parameter와 shape

5)모델 학습 결과

모델 학습 결과 이와 같이 epoch가 늘어날수록 정확도가 늘어나는 것을 확인할 수 있었습니다.

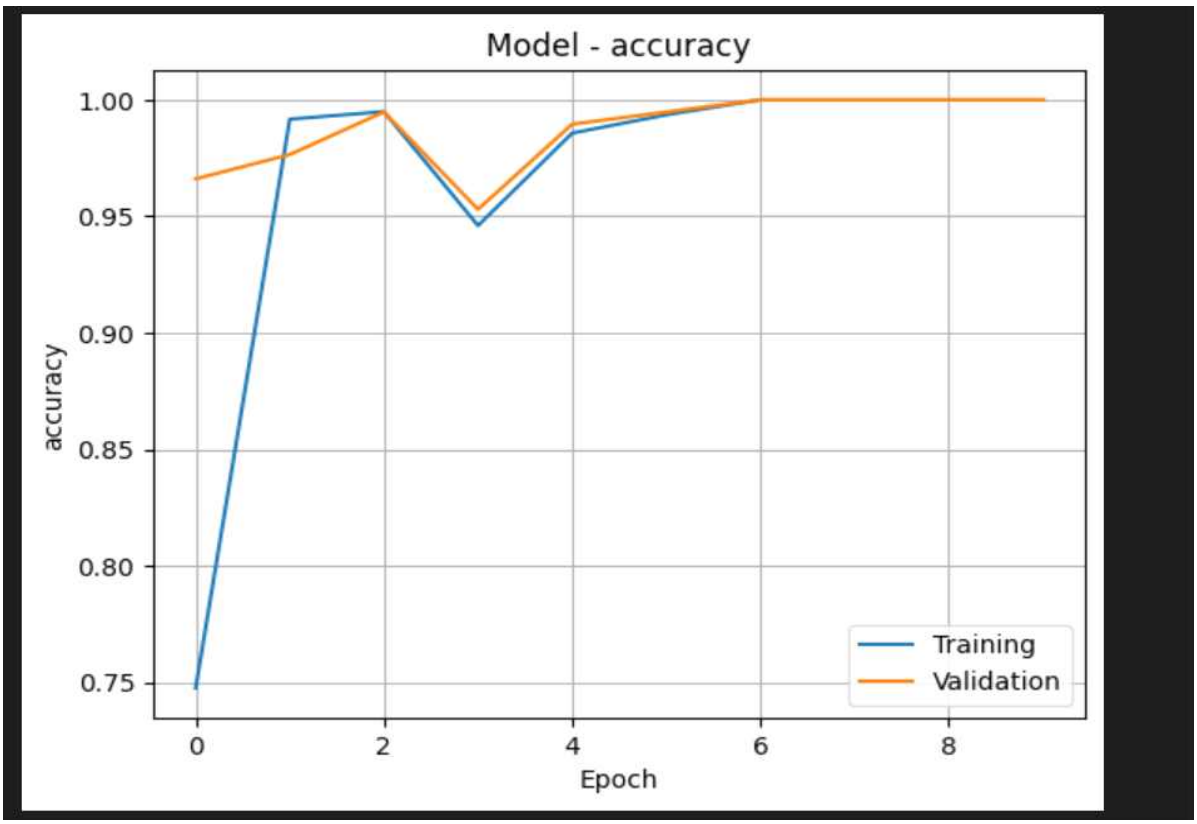


그림 38. CNN 모델의 Accuracy

6)실제 얼굴인식

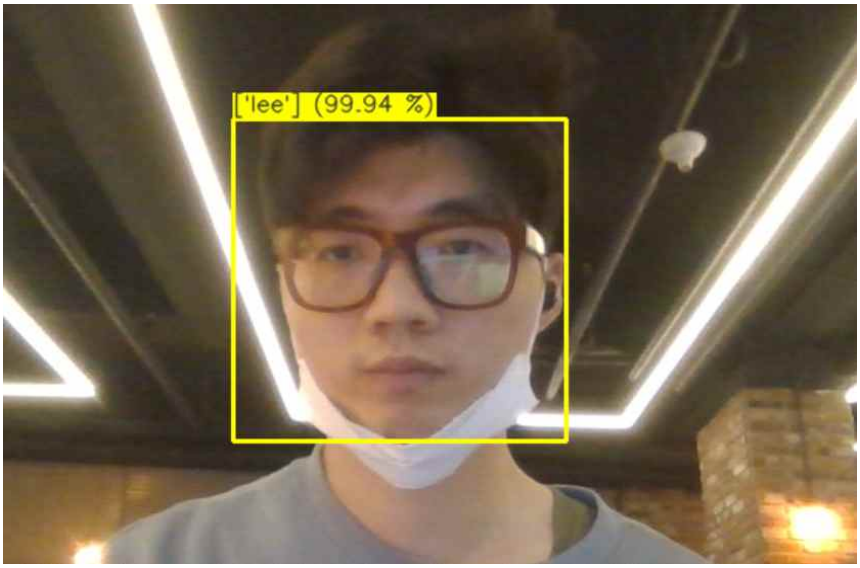


그림 39. 안경을 쓰고 인식시켰을 때 사용자와 일치하는 profile 확인

실시간으로 촬영되고 있는 얼굴과 학습된 모델 결과를 사용해서 예측해보면 정확도가 매우 높게 나옵니다.

결론)

전반적인 confidence 자체는 저희가 최종 작품에서 사용한 LBPH 모델보다 높게 책정되는 것으로 보이는 듯하지만, 해당 방법론을 사용하는 데 있어 치명적인 문제점이 존재했습니다.

우선 사진이 많을수록 학습시키는 양이 많아져 실용적으로 사용하기 힘든 양의 시간이 필요합니다. image augmentation을 활용해 늘어나는 데이터를 학습시키는 과정이 포함되어 있기 때문입니다. 더욱 성능이 좋은 하드웨어를 사용하면 완화할 수 있는 문제로 생각되지만, 현재 구현하고자 하는 작품의 특성상 이는 제한될 것이라고 판단했습니다.

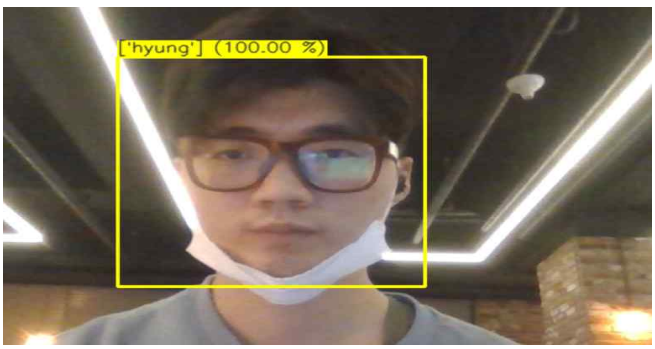


그림 40. 타인으로 인식되는 경우인데, 100%의 confidence를 출력하고 있다.

또한, 얼굴의 미세한 각도 변화로 인해 학습된 사용자의 얼굴을 제3자로 인식하는 경우가 잦다는 것을 확인했습니다. 2D 카메라를 이용했을 때 입력되는 데이터가 입체를 반영하지 못하기에 발생하는 문제로 판단했는데, 해당 이슈가 CNN에서 극명하게 드러났습니다. 위와 같은 이슈들로, CNN 대신 최종 작품에서 채택한 모델을 사용하게 되었습니다.

3. 눈 깜빡임 구현

3.1 눈 깜빡임 인식

3.1.1 Haar 특징 이용

위에 설명한 Haar cascade를 통해 얼굴 특징값 중 하나인 눈 검출 객체 알고리즘을 사용하여서 눈을 검출해주었습니다.

코드 진행 및 그 결과는 다음과 같습니다.

Object 검출 알고리즘(Haar cascades) 눈 검출 알고리즘을 불러옵니다.

```
eyes_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_eye_tree_eyeglasses.xml")
```

그림 41. blink검출을 위한 알고리즘 불러오기

detectMultiscale 함수로 해당 알고리즘을 실행합니다.

```
eyes = eyes_cascade.detectMultiScale(eye_face, 1.3, 5, minSize=(50, 50))
```

그림 42. detectMultiscale 함수를 이용하여 눈 검출

실시간 촬영되고 있는 화면에서 인식된 눈의 길이가, 기존에 설정한 눈의 길이와 일치할 경우 검출이 성공되게끔 하였습니다.

눈을 인식한 후 s를 입력할 경우 first_read 변수가 false로 바뀝니다. 기준에 부합하는 눈의 길이와 일치할 경우 눈이 열려있는 상태로 판단합니다.

눈이 떠 있는 상태에서 눈을 감을 때, 눈이 있는 상태에서 눈이 없어지는데 이것을 깜빡임으로 인식하게 하였습니다. 조건문을 통해 구현했습니다.

```
if len(eyes) >= 2:
    if first_read:
        cv2.putText(image, "Eye's detected, press s to check blink", (70, 70), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (0, 255, 0), 2)
    else:
        cv2.putText(image, "Eye's Open", (70, 70), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (255, 255, 255), 2)
else:
    if first_read:
        cv2.putText(image, "No Eye's detected", (70, 70), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (255, 255, 255), 2)
    else:
        cv2.putText(image, "Blink Detected.....!!!!", (70, 70), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (0, 255, 0), 2)
        cv2.imshow('image',image)
        cv2.waitKey(1)
        print("Blink Detected.....!!!!")
```

그림 43. 눈 깜빡임 감지 코드

해당 과정을 사진과 같이 보면 다음과 같습니다. 우선, 학습된 trainer를 바탕으로 해당 얼굴이 threshold를 넘기는 confidence와 함께 동일인으로 판별될 경우, 아래 깜빡임 검출 알고리즘으로 넘어갑니다.

- 눈을 검출할 경우 깜빡임 단계로 넘어가기 위한 키(s)를 사전 설정 및 입력

그림 44. 사진의 눈은 인식하지만 눈 깜빡임이 없어 log-in 과정으로 넘어가지 않음

- 깜빡임 단계에서 열려있는 눈을 검출

- 열려있는 눈이 인식되고 있는 상황에서 눈을 감으면 깜빡임으로 인식

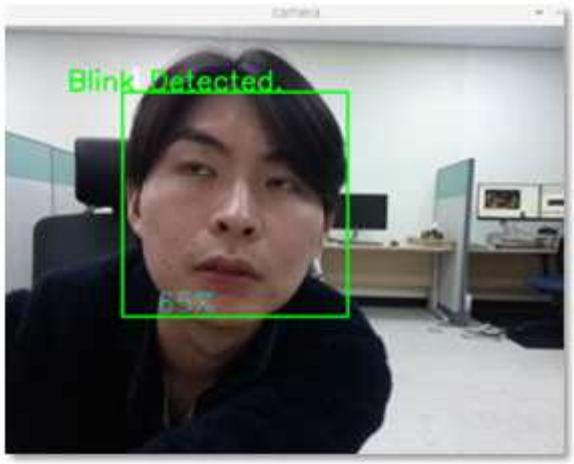


그림 45. 눈 깜빡임 후에 사용자의 얼굴 일치 여부 판단

- 지정 횟수만큼 눈 깜빡임을 인식할 경우, 잠금 해제

4. 데이터 암호화

4.1 데이터 암호화 구현

4.1.1 AES 알고리즘

(1) AES 개요 [11], [12]

고급 암호화 표준(Advanced Encryption Standard)인 AES는 과거의 표준인 DES를 대체한 암호 알고리즘으로, 암호화 및 복호화 과정에서 동일한 키를 사용하는 대칭 키 알고리즘입니다.

DES에 비해 키 사이즈가 자유로워 가변 길이의 블록과 키 사용이 가능하며(128, 192, 256 bit), 속도와 효율성 면에서 뛰어납니다.

	Key kength (Nk words)	Block length (Nb words)	Number of Rounds
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

< Key-Block-Round Combinations >

그림 46. 가변 길이의 키를 사용 가능한 AES

(2) AES의 구조상 장점

AES는 SPN 구조를 띄고 있습니다. SPN 구조는 암복호화 과정에서 역함수가 필요하도록 설계되어야 한다는 단점이 있지만, 중간 과정에서 비트의 이동 없이 한 번에 암복호화가 가능해 Feistel 구조에 비해 효율적인 설계가 가능합니다. Feistel 구조를 사용한 DES의 경우, Swap 단계를 포함해 연산량이 많이 소요됩니다. 안전성 문제로 인해 DES는 AES와 같이 단독으로 사용할 수 없으며, Triple DES와 같이 중첩해 사용되고 있습니다.

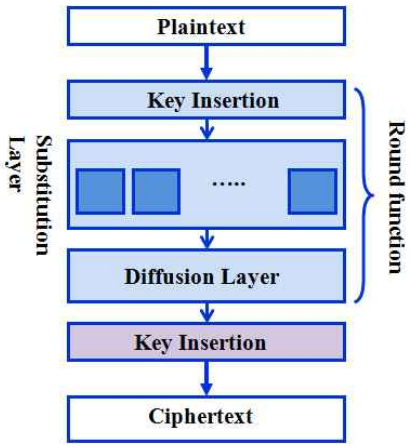


그림 47. SPN 구조

(3) AES 동작 과정

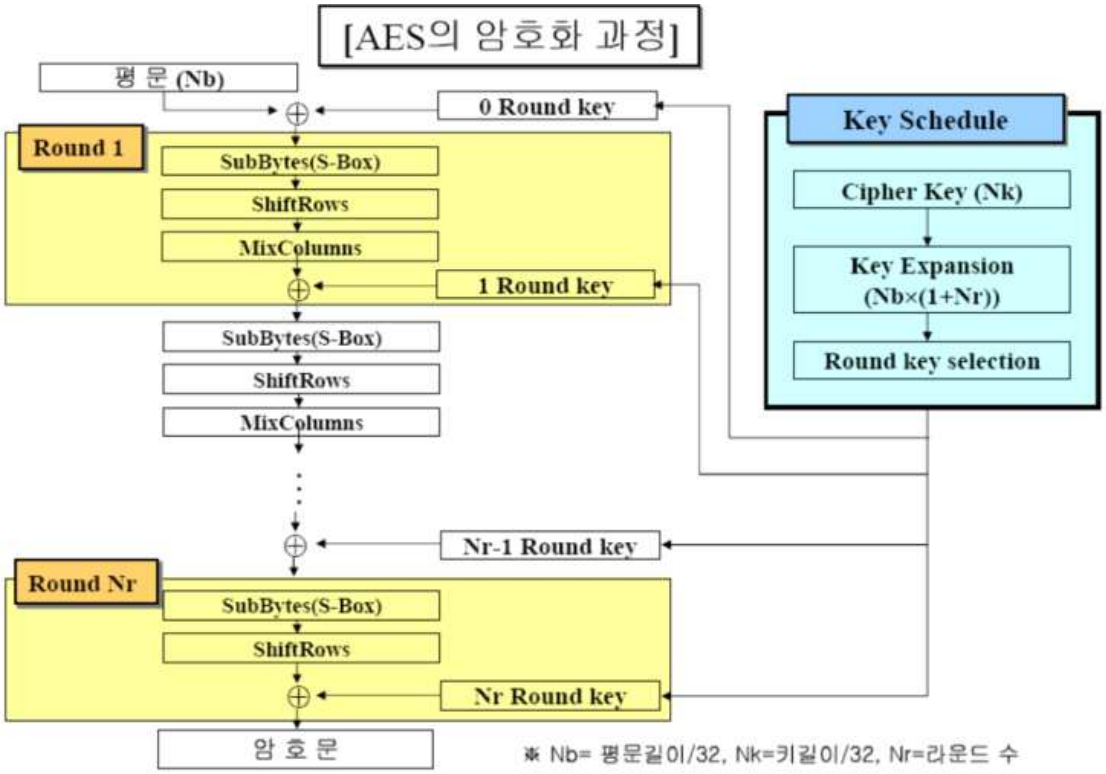


그림 48. AES 암호화 과정

round의 횟수(Nr)에 따라 SubBytes, ShiftRows, MixColumns, AddRoundkey가 반복되는 것을 확인할 수 있습니다. 각 과정은 아래와 같습니다.

- SubBytes

만일 128 비트 입력 중 한 바이트의 값이 9a라면, S-box를 기반으로 b8로 치환됩니다. 값의 앞 자리는 x값, 뒷 자리는 y값을 의미합니다.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

< Sbox >

그림 49. SubBytes에서 사용되는 S-Box

- ShiftRows

상태 행렬의 행에 따른 왼쪽 방향의 cyclic 회전을 수행합니다. 첫 번째 행은 회전되지 않으며, 두 번째 행은 1byte, 세 번째 행은 2bytes, 네 번째 행은 3bytes 만큼 왼쪽으로 cyclic 회전합니다.

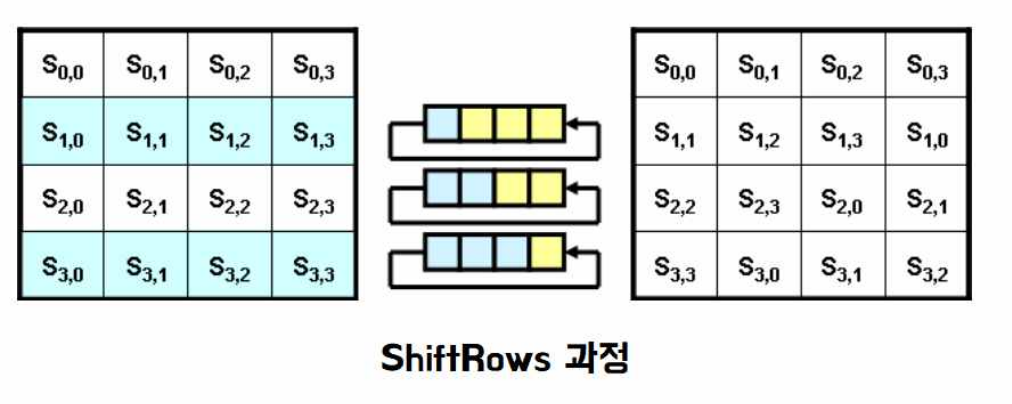
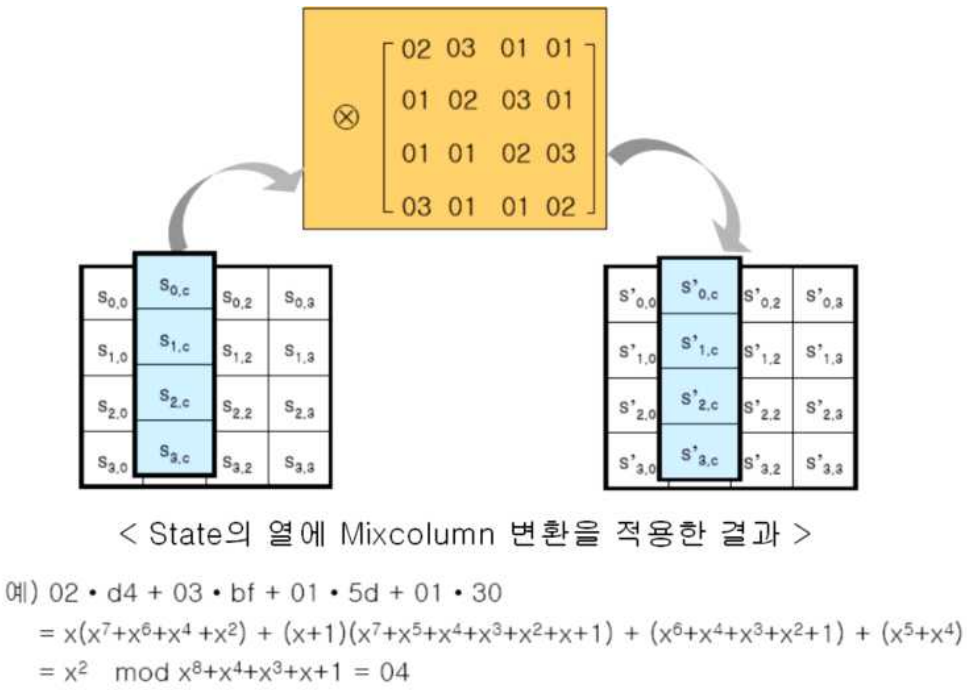


그림 50. ShiftRows의 cyclic 회전

- MixColumns

State의 각 column에 대해 다음과 같이 행렬 곱셈을 수행합니다. 식에서 각 symbol은 유한체 GF(28) 상의 요소이며 0 혹은 1의 계수를 갖는 7차 이항의 다항식으로 표현 가능합니다. 요소간 곱셈은 기약 다항식(x^8+x^4+x^3+x+1)을 법으로 하는 다항식 곱셈이며, 마지막 round에서는 MixColumn을 하지 않습니다.



예) 02 • d4 + 03 • bf + 01 • 5d + 01 • 30

$= x(x^7+x^6+x^4+x^2) + (x+1)(x^7+x^5+x^4+x^3+x^2+x+1) + (x^6+x^4+x^3+x^2+1) + (x^5+x^4)$

$= x^2 \mod x^8+x^4+x^3+x+1 = 04$

그림 51. State의 열에 대한 MixColumn 변환 적용

- AddRoundKey

실제 encryption이 수행되는 과정으로, state 내 각각의 byte들이 각 RoundKey와 XOR 연산됩니다. RoundKey는 Key Expansion Schedule에 따라 key로부터 유도됩니다.

4.1.2 암호화 결과

(1) 데이터 암호화

그림19와 같이 roi가 지정되고, 해당 이미지는 지정된 개수 만큼 학습에 사용됩니다. 학습이 진행됨에 따라, 학습에 사용되는 데이터셋 폴더와 trainer 폴더가 아래와 같이 채워집니다.

```
[INFO] waiting for input...
[INFO] enter password : capstone
enter user id end press <return> ==>  1
[INFO] Initializing face capture. Look the camera and wait ...
[INFO] Exiting Program and cleanup stuff
[INFO] Training faces. It will take a few seconds. Wait ...
[INFO] 1 faces trained. Exiting Program
```

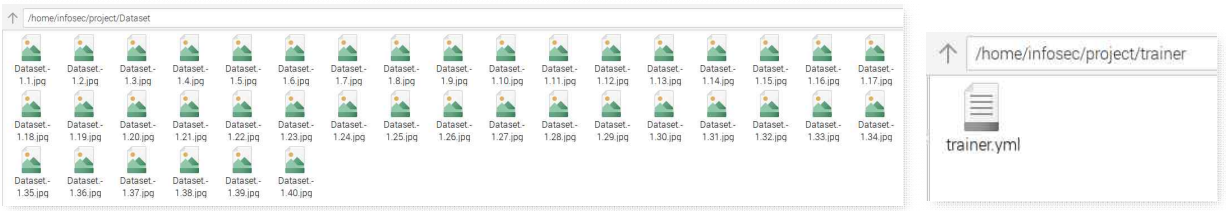


그림 49, 50, 54. 얼굴 학습 과정

학습이 완료되면, 학습에 사용한 출입 허가자의 얼굴 데이터를 AES로 암호화하도록 설계했습니다. 해당 파일을 열람하면, 아래와 같이 유의미한 정보를 얻을 수 없는 암호화된 텍스트만 확인할 수 있습니다.

```
[INFO] Data used for training are encrypted.  Exiting Program.
```

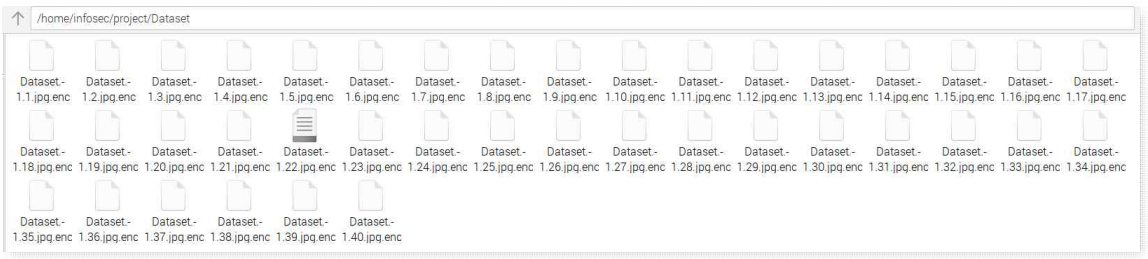


그림 52, 56. 얼굴 학습 종료 후 학습에 사용된 데이터 암호화

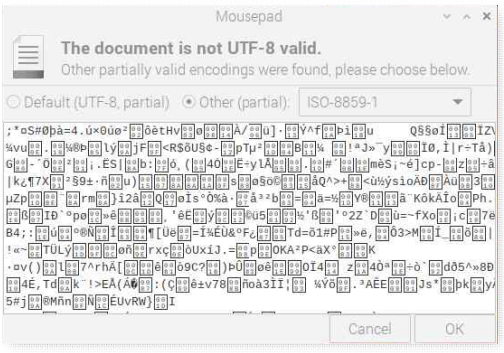


그림 57. 암호화된 데이터 파일 열람 시 얻을 수 있는 무의미한 정보

(2) 데이터 복호화 기능 추가

앞서 작성한 것과 같이 얼굴 감지 및 인식, 눈과 깜빡임 인식 등 사용자 인증 과정이 모두 끝나 출입이 허가된 경우, 해당 암호화된 데이터를 복호화할 수 있는 기능을 추가했습니다. 그 결과는 아래와 같습니다.

```
[INFO] waiting for input...
[INFO] Blink your eyes for three times to open.
[INFO] Blink Detected (blink count : 1)
[INFO] Blink Detected (blink count : 2)
[INFO] Blink Detected (blink count : 3)

[INFO] Exiting Program and cleanup stuff
sh: 1: cls: not found
[INFO] Data used for training are decrypted
sh: 1: cls: not found
```

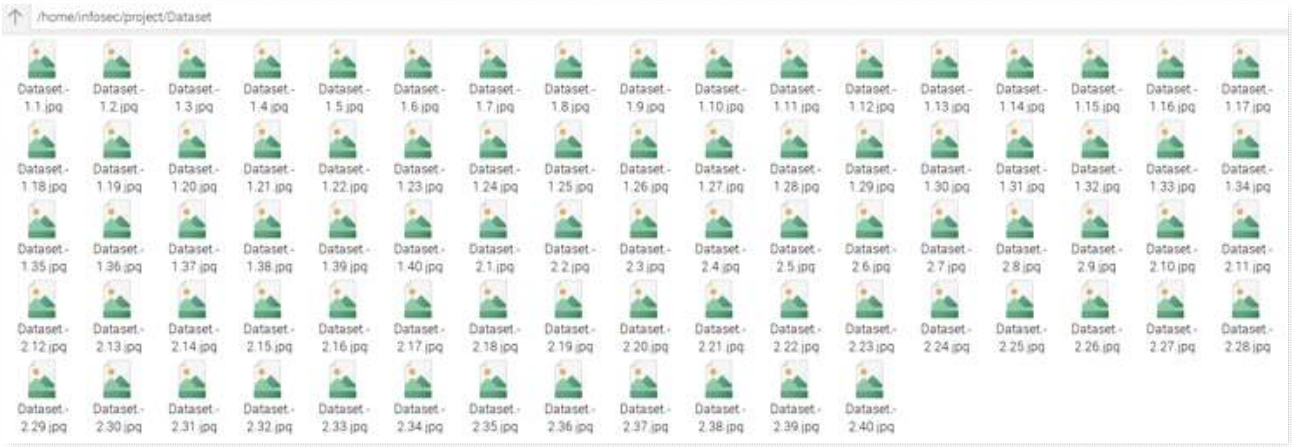


그림 55, 59. 사용자 인증 완료 시, 데이터 복호화 기능 추가

5. 하드웨어

5.1 하드웨어 구성

5.1.1 하드웨어 개요

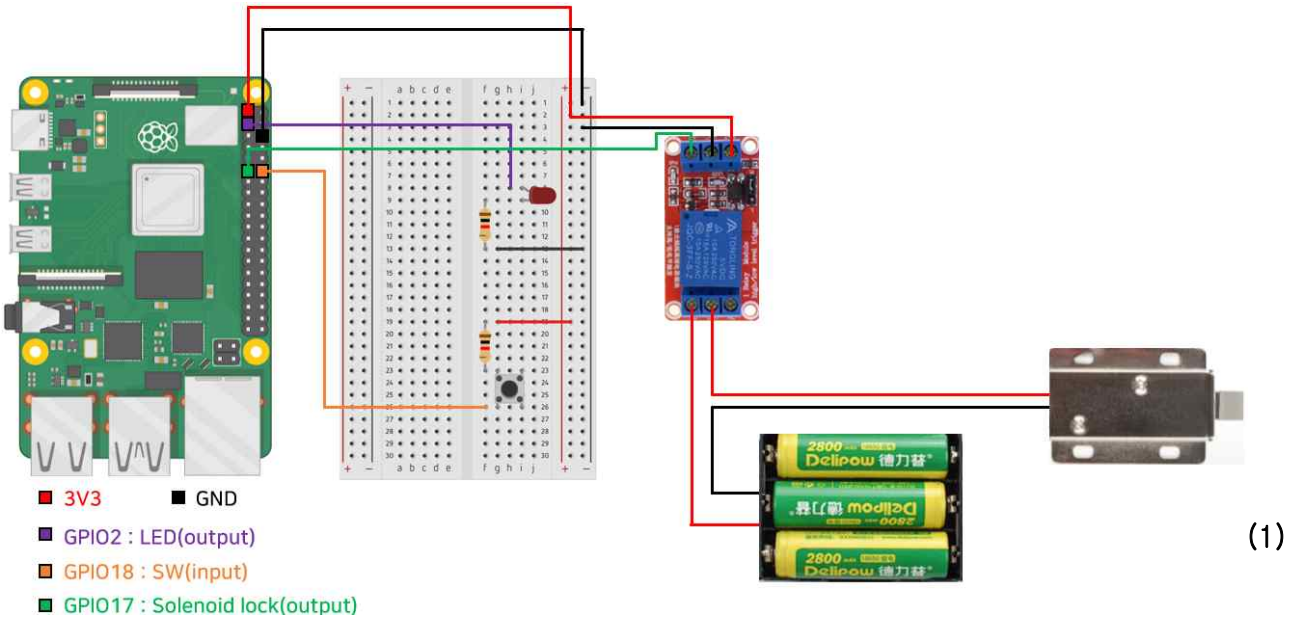


그림 60. 전체 하드웨어 개요 및 연결

하드웨어는 위 그림과 같이 라즈베리파이, 브레드보드, LED, 택트 스위치, 저항, 1채널 릴레이 모듈, 리튬이온배터리, 솔레노이드 도어락으로 구성되어 있습니다.

라즈베리파이의 GPIO pin의 입/출력 신호를 제어하여 전체 하드웨어를 제어합니다. 스위치 1을 누르면 LED 1이 켜지며 얼굴 학습이 시작되고, 스위치2를 누르면 LED2가 켜지면서 얼굴 인식이 시작됩니다. 얼굴이 인식되면 솔레노이드 도어락 잠금이 해제됩니다.

5.1.2 구성 요소

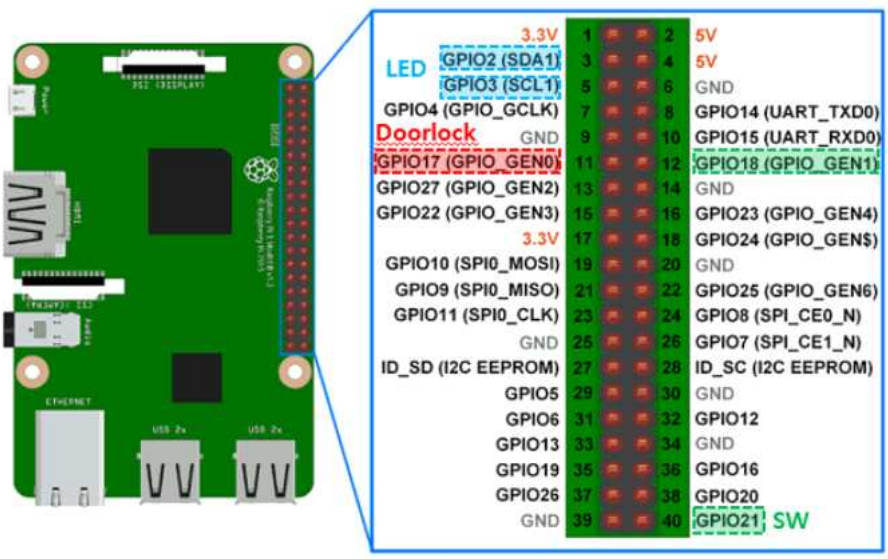


그림 61 Raspberry Pi 4 의 Pin map [13]

(1) 라즈베리파이

GPIO2, 3 pin은 output pin으로 설정되어 High를 출력하면 LED on, Low를 출력하면 LED off 상태가 됩니다. GPIO18, 21은 input pin으로 설정되어 SW를 입력하면 High, 입력하지 않으면 Low 상태가 됩니다. GPIO17은 output pin으로 설정되어 High를 출력하면 Solenoid Doorlock이 unlock되고, Low를 출력하면 lock됩니다.

(2) 1ch optocoupler relay module

그림 62. Relay Module의 Pin map [14]

1채널 릴레이 모듈은 Relay를 제어할 수 있는 모듈로써 기본적으로 5V에서 동작합니다. Relay는 전자석의 원리로 전류가 흐르면 자기장을 형성해 자기력으로 자석을 끌어당겼다가 전류가 흐르지 않으면 자석을 놓는 원리입니다. 즉, 이 모듈은 스위치로써 작동할 수 있습니다. Relay에 센서, 모듈 또는 가정에서 사용하는 멀티탭이나 형광등 스위치 등을 연결하여 On/Off 제어가 가능합니다. 본 작품에서는 Solenoid의 Lock/Unlock을 제어하는 스위치의 역할을 수행할 수 있습니다.

릴레이 모듈의 NO와 NC는 각각 Normal Open, Normal Close를 의미합니다. NO는 자동제어에서 A 접점으로 릴레이 신호가 off일 때에는 NO와 C가 떨어져 있다가 릴레이 신호가 on일 때 NO와 C가 연결됩니다. 반대로, NC는 자동제어에서 B접점으로 릴레이 신호가 off일 때에는 NC와 C가 연결되어 있다가 릴레이 신호가 on일 때 NC와 C가 떨어집니다.
동작시간이 많지 않은 기기의 경우 불필요한 전력소비와 릴레이의 수명 단축을 막기 위해 NO를 사용하는 것이 좋습니다. 본 작품은 도어락의 특성상 동작시간이 짧기 때문에 NO를 사용하였습니다.

릴레이 모듈의 연결을 보면, DC-, DC+, Input signal을 각각 라즈베리파이의 GND, 3.3VDC, GPIO17 pin에 연결합니다. 그 후, NO(Normally Open), COM(Common)을 각각 Li-ion 배터리, Solenoid Doorlock에 연결합니다.

(3) Solenoid Doorlock

그림 63. 솔레노이드 도어락 제품 [15]

솔레노이드는 스프링 형태로 도선을 원통형으로 감아서 만든 기기입니다. 솔레노이드의 경우 전류가 흐르면 주위에 자기장이 형성되는 현상이 발생합니다. 이러한 솔레노이드의 성질을 활용하여, 전류가 흐르면 도어락이 열리고 흐르지 않을 경우에는 도어락이 잠기는 원리입니다. 라즈베리파이의 GPIO17 pin과 연결하여 제어하며, High일때 Unlock, Low일 때 Lock입니다. 얼굴 인식에 성공하면 라즈베리파이의 GPIO17에서 High 신호를 출력하여 Solenoid Doorlock의 잠금을 해제할 수 있습니다.

5.1.3 라즈베리파이 코딩

(1) 초기 설정

```
import cv2
import numpy as np
from PIL import Image
import RPi.GPIO as GPIO
import os
from Crypto import Random
from Crypto.Cipher import AES
from time import sleep

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

GPIO.setup(17, GPIO.OUT)
```

그림 64.GPIO 핀들의 번호를 지정하는 라이브러리(RPi.GPIO) 호출 및 설정

(2) LED / SW

```
LED_pin_1 = 2  # LED 핀은 라즈베리파이 GPIO 2번핀으로
LED_pin_2 = 3

sw_pin_1 = 18  # 스위치 핀은 라즈베리파이 GPIO 17번핀으로
sw_pin_2 = 21

GPIO.setup(LED_pin_1, GPIO.OUT)  # LED 핀을 출력으로 설정
GPIO.setup(sw_pin_1, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

GPIO.setup(LED_pin_2, GPIO.OUT)  # LED 핀을 출력으로 설정
GPIO.setup(sw_pin_2, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
```

그림 65. 라즈베리파이 GPIO에 LED, Switch 연결

GPIO2, GPIO3은 LED를 제어하는 Output pin으로 설정하고, GPIO18, GPIO21은 SW를 제어하는 Input pin으로 설정합니다.

GPIO2, GPIO3가 High이면 LED가 켜지고, Low이면 LED가 꺼집니다. GPIO18, GPIO21은 SW가 눌리면 High, 눌리지 않으면 Low 상태가 됩니다.

```
try:
    GPIO.output(LED_pin_1, GPIO.LOW)
    GPIO.output(LED_pin_2, GPIO.LOW)
    print(" [INFO] waiting for input...")
```

그림 66. 초기 설정 LED 제어

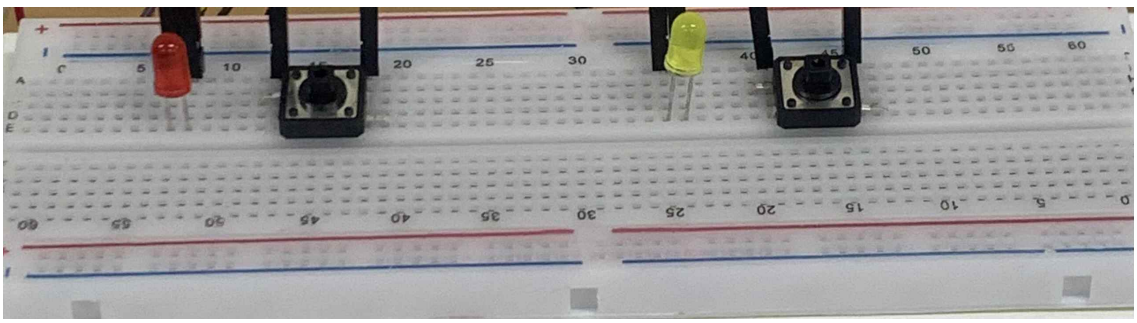


그림 67. LED, switch 연결 사진

프로그램이 실행되면, LED1, LED2 모두 Off 상태로 초기화됩니다.

```
while True:  # 무한루프 시작: 아두이노의 loop()와 같음
    if GPIO.input(sw_pin_1) == GPIO.HIGH:
        GPIO.output(LED_pin_1, GPIO.HIGH)
        dataset()
        train()
        aes_enc()
        GPIO.output(LED_pin_1, GPIO.LOW)
```

그림 68. SW1 입력 시 코드 실행

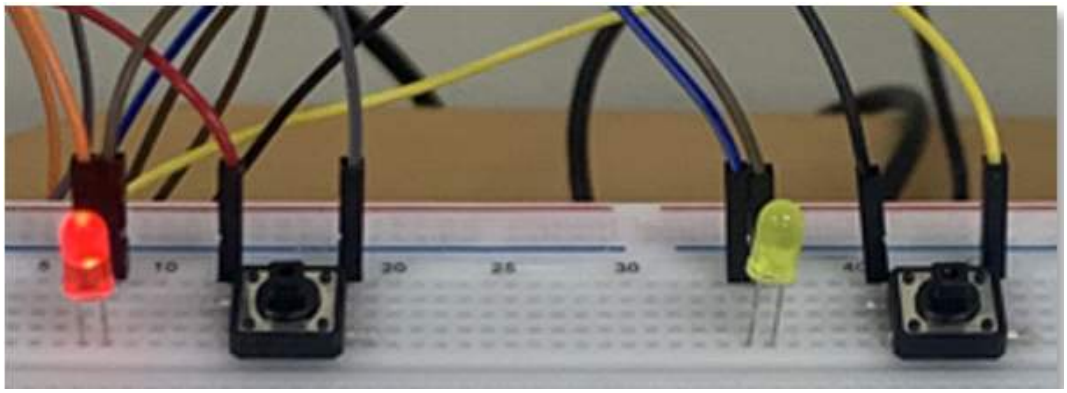


그림 69. LED, sw 사진

SW1을 누르면 LED1이 켜지며 데이터(사진) 수집/학습/암호화를 진행합니다. 모든 과정이 끝나면 LED1이 꺼집니다.

```
if GPIO.input(sw_pin_2) == GPIO.HIGH:
    GPIO.output(LED_pin_2, GPIO.HIGH)
    recognition()
    aes_dec()
    GPIO.output(LED_pin_2, GPIO.LOW)
```

그림 70. sw2 입력 시 코드 실행

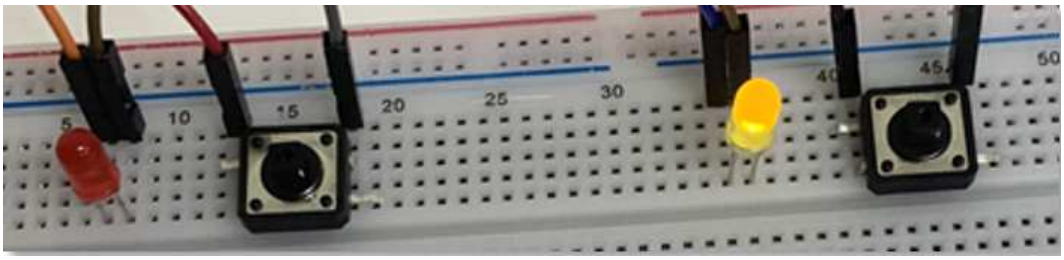


그림 71. LED, sw 사진

SW2를 누르면 LED2가 켜지면서 얼굴 인식 및 복호화를 진행합니다. 모든 과정이 끝나면 LED2가 꺼집니다.

(3)Solenoid Doorlock

```
GPIO.setup(17, GPIO.OUT)
    elif k == ord('s'):
        first_read = False

    if blinkcount == 3:
        GPIO.output(17, 1)
        sleep(5)
        GPIO.output(17, 0)
        break
# Do a bit of cleanup
print("\n [INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```

그림 73. Solenoid 잠금 장치 제어 코드

GPIO17을 Output으로 설정하여 얼굴을 인식한 후, 눈 깜빡임을 3번 감지하면 GPIO170이 High가 되면서 Solenoid Doorlock의 잠금을 해제할 수 있습니다. 그 후, 5초간 잠금 해제가 지속되고 GPIO170이 Low가 되면서 Solenoid Doorlock이 잠깁니다.

6. 결론 및 추후과제

6.1 결론

기존에 존재하는 2D 카메라를 사용하는 얼굴 인식 잠금 해제 기능 사용 재화의 문제점을 파악한 결과 1) 사용자의 사진을 통해 간단하게 잠금 해제 2) 수면 중인 사용자의 얼굴을 스캔해 잠금 해제 3) 사용자의 민감한 정보가 해킹 등으로 유출 등의 다양한 이슈가 있음을 알 수 있었습니다. 특히 사용자의 사진을 통해 간단하게 잠금 해제가 주요 이슈였으며 3D 카메라를 사용한 경우, 수면 중인 사용자의 얼굴 스캔 후 잠금 해제 등의 이슈가 있었습니다.

또한, 민감한 생체 정보 데이터가 해킹 등으로 유출되는 것을 방지해야 하는 고려점이 존재했습니다.
저희는 이러한 1) 의도성을 가진 사람에 의한 얼굴 인식 잠금 해제 2) 민감한 생체 정보 데이터 유출 방지를 위한 보안성 확보에 주안점을 두어 프로젝트를 진행하였습니다.

다음과 같은 이슈들을 해결하기 위해 얼굴 인식과 더불어 눈 깜빡임과 같이 특정 행동을 인식할 때 도어락 잠금을 해제할 수 있는 능동적 기능을 탑재하였으며 실제 테스트를 통해 그 기능을 수행할 수 있음을 확인했습니다.

보안성 확보의 경우, 암호 표준으로 사용되며 카카오톡 / 텔레그램 과 같은 대기업에서 활용되는 AES 암호화 Protocol을 사용하여 실용적으로 사용될 수 있는 보안 수준을 달성하는데 성공했습니다.

6.2 추후과제

1) 개선방안

본 프로젝트는 만일 공격자가 출입 허가자의 눈 깜빡임 영상을 선명한 화질로 보유하고 있다면 공격에 성공할 수도 있다는 허점을 가지고 있습니다. 이에 대한 방어로, 눈 깜빡임뿐만 아니라 얼굴을 사용한 행동을 랜덤하게 요구할 수 있다면 방어에 성공할 수 있을 것입니다. 예를 들어, 한 눈만 깜빡이거나, 혀를 내밀게 하는 등의 여러 행동을 랜덤하게 요구하는 것입니다. 출입 허가자의 해당 영상들을 공격자가 모두 가질 수 없음을 이용한 방어 방안입니다. 해당 사항까지 보완되면 개성과 경쟁력을 확보한 Face ID 기술이 될 것이라고 생각합니다.

또한 동형 암호를 활용해 서버에 저장된 출입 허가자의 데이터와 출입 시도자의 데이터, 위 두 데이터를 모두 암호화한 상태에서 그 유사도를 비교하는 방식을 적용하면 보안성 측면에서 더 좋은 작품이 될 것입니다. 기술적 한계로 위 과정은 구현하지 못하였지만 해당 부분을 구현한다면 더 안전한 Face ID 기술이 될 것입니다.

마지막으로 CNN 등 얼굴 인식을 위한 다양한 딥러닝 모델을 직접 구현하여 학습에 사용했을 때보다 현재의 모델이 정확도, 속도 면에서 더 우월한 성능을 보여 채택해 사용했습니다. 타 모델의 해당 한계를 극복하고 현재 모델보다 더 적합한 딥러닝 모델을 구현할 수 있다면 보다 정확하고 빠른 서비스를 제공할 수 있을 것입니다.

2) 적용방안

최신 아이폰과 같이 현존 최고 기술들이 집대성된 고성능, 소형 장비는 그 가격이 매우 높습니다. 간단해 보이는 조그마한 사다리꼴 핸드폰 카메라 박스에도 수많은 초소형 장비가 내장되어있습니다. 해당 프로젝트를 통해 제작한 Face ID 도어락의 경우, 2D 카메라와 Raspberry Pi 등 수급이 어렵지 않은 장비를 활용하여 생산 및 가격 면에서 경쟁력을 가지고 있어 부담스럽지 않게 실생활에서 사용될 수 있습니다. 또한, 현존 가장 안전하며 실용적인 암호 프로토콜 중 하나인 AES protocol을 사용하기에 보안 면에서도 현존 장비에 밀리지 않는 실용적이며 안전한 서비스로써 사용될 수 있습니다.

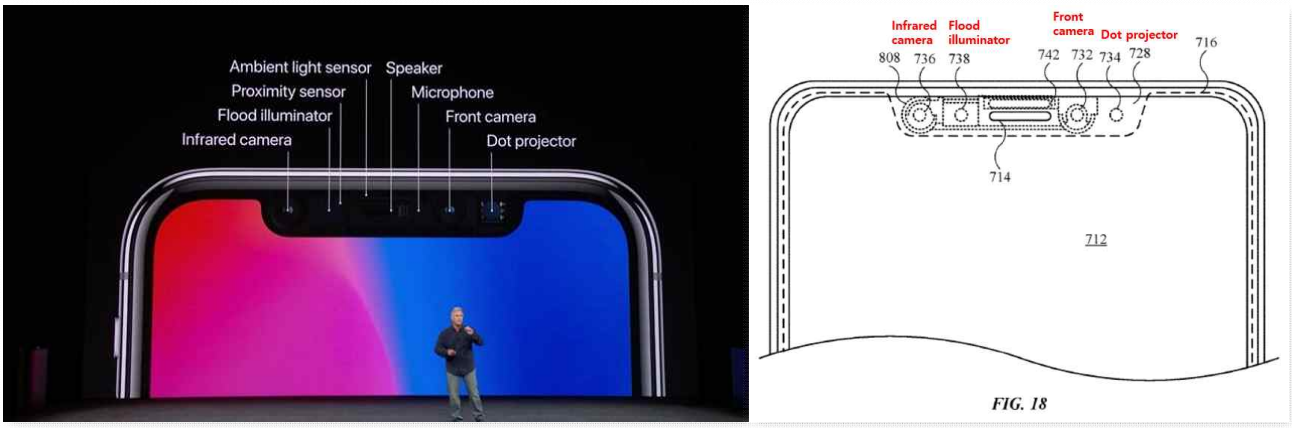


그림 74. 초소형, 고성능 장비가 집약된 아이폰의 카메라 박스 [3]

7. 참고 자료

[1] 혹시 갤럭시s20 시리즈는 얼굴인식 잠금해제 사진으로 풀리나요?
https://r1.community.samsung.com/t5/%EA%B0%A4%EB%9F%AD%EC%8B%9C-s/%ED%98%B9%EC%8B%9C-%EA%B0%A4%EB%9F%AD%EC%8B%9Cs20-%EC%8B%9C%EB%A6%AC%EC%A6%88%EB%8A%94-%EC%96%BC%EA%B5%B4%EC%9D%B8%EC%8B%9D-%EC%9E%A0%EA%B8%88%ED%95%B4%EC%A0%9C-%EC%82%AC%EC%A7%84%EC%9C%BC%EB%A1%9C-%ED%92%80%EB%A6%AC%EB%82%98%EC%9A%94/td-p/7894122
[2] 보안, 얼굴 사진으로 자금 화면 해제되는 스마트폰 다수
https://infrawaretech-qa.tistory.com/22
[3] 애플 Face ID(페이스 아이디)와 갤럭시의 얼굴 인식 기술 원리
https://m.blog.naver.com/taekyoon99/221949792896
[4] Face ID에 적용된 첨단 기술에 관하여 https://support.apple.com/ko-kr/HT208108
[5] 삼성전자 또 해킹 당해...미 고객정보 유출
https://zdnet.co.kr/view/?no=20220905163420
[6] 악성앱 이용한 스마트폰 해킹 공포... 실제 감염시 어떤 증상 나타날까
https://www.boannews.com/media/view.asp?idx=96073
[7] 논문 P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR, 2001
[8] Haar Cascades에 대해 https://webnautes.tistory.com/1352
[9] Local binary pattern (LBP)의 원리 및 활용
https://bskyvision.com/entry/Local-binary-pattern-LBP%EC%9D%98-%EC%9B%90%EB%A6%AC-%EB%B0%8F-%ED%99%9C%EC%9A%A9
[10] 이미지 어그멘테이션의 효과 https://codetorial.net/tensorflow/image_augmentation.html
[11] AES 암호 알고리즘(Advanced Encryption Standard) https://www.crocus.co.kr/1230
[12] AES Proposal: Rijndael (by Joan Daemen, Vincent Rijmen)
[13] 라즈베리파이 Pin map
https://www.electronicwings.com/raspberry-pi/raspberry-pi-gpio-access
[14] 1ch optocoupler relay module https://lora.tistory.com/27
[15] 솔레노이드 도어락
https://digiwarestore.com/en/other-appliances/solenoid-door-lock-12v-dc-267059.html

부록

```
importcv2
importnumpy asnp
fromPIL importImage
importRPi.GPIO asGPIO
importos
fromCrypto importRandom
fromCrypto.Cipher importAES
fromtime importsleep

GPIO.setmode(GPIO.BCM)      # GPIO 핀들의 번호를 지정하는 규칙 설정
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)


tmp=0
password_key = "capstone"
password = input("\n[INFO] enter password : ")

while True:
    ifpassword == password_key:
        break

    else:
        print("\n[INFO] Wrong Password")
password = input("\n[INFO] enter password : ")
### 이부분은 아두이노 코딩의 setup()에 해당합니다
LED_pin_1 = 2                     # LED 핀은 라즈베리파이 GPIO 2번핀으로
LED_pin_2 = 3
#LED_pin_3 = 4
sw_pin_1 = 18                    # 스위치 핀은 라즈베리파이 GPIO 17번핀으로
sw_pin_2 = 21
#sw_pin_3 = 4
GPIO.setup(LED_pin_1, GPIO.OUT)   # LED 핀을 출력으로 설정
GPIO.setup(sw_pin_1, GPIO.IN, pull_up_down= GPIO.PUD_DOWN)
GPIO.setup(LED_pin_2, GPIO.OUT)   # LED 핀을 출력으로 설정
GPIO.setup(sw_pin_2, GPIO.IN, pull_up_down= GPIO.PUD_DOWN)



path = '/home/infosec/project/Dataset'
recognizer = cv2.face_LBPHFaceRecognizer.create()
detector = cv2.CascadeClassifier('/home/infosec/project/cascades/data/haarcascade_f
rontalface_default.xml');
def dataset():
cam = cv2.VideoCapture(0)
cam.set(3, 640) # set video width
cam.set(4, 480) # set video height
face_detector = cv2.CascadeClassifier('/home/infosec/project/cascades/data/haarcasc
ade_frontalface_default.xml')
    # For each person, enter one numeric face id
face_id = input('\nenter user id end press <return> ==>  ')
    print("\n[INFO] Initializing face capture. Look the camera and wait ...")
    # Initialize individual sampling face count
count = 0
    while(True):
```

```
ret, img = cam.read()
        #img = cv2.flip(img, -1) # flip video image vertically
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_detector.detectMultiScale(gray, 1.3, 5)
        for(x,y,w,h) infaces:
cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
count += 1
            # Save the captured image into the datasets folder
cv2.imwrite("/home/infosec/project/Dataset/Dataset."+ str(face_id) + '.'+ str(count
) + ".jpg", gray[y:y+h,x:x+w])
cv2.imshow('image', img)
k = cv2.waitKey(100) & 0xff # Press 'ESC' for exiting video
        ifk == 27:
            break
        elifcount >= 40: # Take 30 face sample and stop video
            break
# Do a bit of cleanup
    print("\n[INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
def getImagesAndLabels(path):
imagePaths = [os.path.join(path,f) forf inos.listdir(path)]
faceSamples=[]
ids = []
    forimagePath inimagePaths:
PIL_img = Image.open(imagePath).convert('L') # convert it to grayscale
img_numpy = np.array(PIL_img,'uint8')
        id= int(os.path.split(imagePath)[-1].split(".")[1])
faces = detector.detectMultiScale(img_numpy)
        for(x,y,w,h) infaces:
faceSamples.append(img_numpy[y:y+h,x:x+w])
ids.append(id)
    returnfaceSamples,ids
def aes_enc():
    ifos.path.isfile('data.txt.enc'):
        while True:
password = password_key
enc.decrypt_file("data.txt.enc")
p = ''
            with open("data.txt", "r") asf:
p = f.readlines()
            ifp[0] == password:
enc.encrypt_file("data.txt")
                break
        while True:
clear()
enc.encrypt_all_files('/home/infosec/project/Dataset/')
            print(' [INFO] Data used for training are encrypted.  Exiting Program.'
)
            exit()

def aes_dec():
    ifos.path.isfile('data.txt.enc'):
        while True:
password = password_key
enc.decrypt_file("data.txt.enc")
p = ''
            with open("data.txt", "r") asf:
```

```
p = f.readlines()
            ifp[0] == password:
enc.encrypt_file("data.txt")
                break
        while True:
            try:
clear()
enc.decrypt_all_files('/home/infosec/project/Dataset/')
                print(' [INFO] Data used for training are decrypted')
                exit()
            except:
clear()
                print(' [INFO] No Data to decrypt')
                exit()
def train():
    print("\n[INFO] Training faces. It will take a few seconds. Wait ...")
faces,ids = getImagesAndLabels(path)
recognizer.train(faces, np.array(ids))
    # Save the model into trainer/trainer.yml
recognizer.write('/home/infosec/project/trainer/trainer.yml') # recognizer.save() w
orked on Mac, but not on Pi
    # Print the numer of faces trained and end program
    print("\n[INFO] {0}faces trained. Exiting Program".format(len(np.unique(ids))))
def recognition():
blinkcount=0

recognizer.read('/home/infosec/project/trainer/trainer.yml')
cascadePath =  "/home/infosec/project/cascades/data/haarcascade_frontalface_default
.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);
eyes_cascade = cv2.CascadeClassifier("/home/infosec/project/cascades/data/haarcasca
de_eye_tree_eyeglasses.xml")
font = cv2.FONT_HERSHEY_SIMPLEX
    id= 0
names = ['0', '1', '2', '3', '']
first_read = True

cam = cv2.VideoCapture(0)
cam.set(3, 640) # set video widht
cam.set(4, 480) # set video height
minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)
    print(' [INFO] Blink your eyes for three times to open.')
    while True:
tmp=0

ret, img =cam.read()
        #img = cv2.flip(img, -1) # Flip vertically (상하,좌우 대칭) 이 줄을 없애서 원
본그대로 카메라가 찍게 만듬.
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

faces = faceCascade.detectMultiScale(
gray,
            scaleFactor= 1.2,
            minNeighbors= 5,
            minSize= (int(minW), int(minH)),
)
        for(x,y,w,h) infaces:
```

```
img=cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
            id, confidence = recognizer.predict(gray[y:y+h,x:x+w])
eye_face = gray[y:y + h, x:x + w]
eye_face_clr = img[y:y + h, x:x + w]
            # Check if confidence is less them 100 ==> "0" is perfect match
eyes = eyes_cascade.detectMultiScale(eye_face, 1.3, 5, minSize=(50, 50))
            if(confidence < 90):
                id= names[id]
confidence = "  {0}%".format(round(100- confidence))
                if len(eyes) >= 2:
                    iffirst_read:
cv2.putText(img, "Eye's detected, press s to check blink", (70, 70), cv2.FONT_HERSH
EY_SIMPLEX,1, (0, 255, 0), 2)
                    else:
cv2.putText(img, "Eye's Open", (70, 70), cv2.FONT_HERSHEY_SIMPLEX,1, (255, 255, 255
), 2)
                else:
                    iffirst_read:
cv2.putText(img, "No Eye's detected", (70, 70), cv2.FONT_HERSHEY_SIMPLEX,
                                    1, (255, 255, 255), 2)
                    else:
cv2.putText(img, "Blink Detected.", (70, 70), cv2.FONT_HERSHEY_SIMPLEX,1, (0, 255,
0), 2)
cv2.waitKey(1)
                        print(" [INFO] Blink Detected (blink count : "+ str(blinkco
unt + 1) + ')')
blinkcount+=1
        #elif (confidence > 70):
            #GPIO.output(17,0)
            #sleep(1)

            else:
            #GPIO.output(17,0)
                id= "stranger"
confidence = "  {0}%".format(round(100- confidence))

cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)
cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)

cv2.imshow('camera',img)
k = cv2.waitKey(10) & 0xff # Press 'ESC' for exiting video
        ifk == 27:
         break
        elifk == ord('s'):
first_read = False
        ifblinkcount==3:
GPIO.output(17,1)
sleep(5)
GPIO.output(17,0)
            break
    # Do a bit of cleanup
    print("\n[INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
#if __name__ == "__main__":
class Encryptor:
    def __init__(self, key):
        self.key = key
```

```
    def pad(self, s):
        returns + b"\0"* (AES.block_size - len(s) % AES.block_size)
    def encrypt(self, message, key, key_size=256):
message = self.pad(message)
iv = Random.new().read(AES.block_size)
cipher = AES.new(key, AES.MODE_CBC, iv)
        returniv + cipher.encrypt(message)
    def encrypt_file(self, file_name):
        with open(file_name, 'rb') asfo:
plaintext = fo.read()
enc = self.encrypt(plaintext, self.key)
        with open(file_name + ".enc", 'wb') asfo:
fo.write(enc)
os.remove(file_name)
    def decrypt(self, ciphertext, key):
iv = ciphertext[:AES.block_size]
cipher = AES.new(key, AES.MODE_CBC, iv)
plaintext = cipher.decrypt(ciphertext[AES.block_size:])
        returnplaintext.rstrip(b"\0")
    def decrypt_file(self, file_name):
        with open(file_name, 'rb') asfo:
ciphertext = fo.read()
dec = self.decrypt(ciphertext, self.key)
        with open(file_name[:-4], 'wb') asfo:
fo.write(dec)
os.remove(file_name)
    def getAllFiles(self, path):
        #dir_path = os.path.dirname(os.path.realpath(__file__))
dirs = []
        fordirName, subdirList, fileList inos.walk(path):
            forfname infileList:
                if(fname != 'script.py' andfname != 'data.txt.enc'):
dirs.append(dirName + "//"+ fname)
        returndirs
    def encrypt_all_files(self, path):
dirs = self.getAllFiles(path)
        forfile_name indirs:
            self.encrypt_file(file_name)
    def decrypt_all_files(self, path):
dirs = self.getAllFiles(path)
        forfile_name indirs:
            self.decrypt_file(file_name)


key = b'[EX\xc8\xd5\xbfI{\xa2$\x05(\xd5\x18\xbf\xc0\x85)\x10nc\x94\x02)j\xdf\xcb\xc
4\x94\x9d(\x9e'
enc = Encryptor(key)
clear = lambda: os.system('cls')
try:
GPIO.output(LED_pin_1, GPIO.LOW)
GPIO.output(LED_pin_2, GPIO.LOW)
    print(" [INFO] waiting for input...")
    while True:                          # 무한루프 시작: 아두이노의 loop()와 같음
        ifGPIO.input(sw_pin_1) == GPIO.HIGH:
GPIO.output(LED_pin_1, GPIO.HIGH)
dataset()
train()
aes_enc()
```

```
GPIO.output(LED_pin_1, GPIO.LOW)

        ifGPIO.input(sw_pin_2) == GPIO.HIGH:
GPIO.output(LED_pin_2, GPIO.HIGH)
recognition()
aes_dec()
GPIO.output(LED_pin_2, GPIO.LOW)
finally:                                  # try 구문이 종료되면
GPIO.cleanup()
```

#CNN 코드

```
#!/usr/bin/env python
# coding: utf-8
# In[5]:


importos
importcv2
importitertools
importnumpy asnp
importmatplotlib.pyplot asplt
fromsklearn.preprocessing importLabelEncoder
fromsklearn.model_selection importtrain_test_split
fromsklearn.metrics importclassification_report
fromsklearn.metrics importconfusion_matrix


# In[6]:


importkeras
fromkeras.models importSequential, Model
fromkeras.layers importDense, Activation, Input
fromkeras.utils importto_categorical


# In[7]:


fromkeras.layers importConv2D, MaxPool2D, Flatten


# In[8]:


def detect_face(img):
img = img[70:195,78:172]
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img = cv2.resize(img, (50, 50))
    returnimg


# In[9]:
```

```
def print_progress(val, val_len, folder, bar_size=20):
progr = "#"*round((val)*bar_size/val_len) + " "*round((val_len - (val))*bar_size/va
l_len)
    ifval == 0:
        print("", end= "\n")
    else:
        print("[%s] (%dsamples)\tlabel : %s \t\t"% (progr, val+1, folder), end="\r"
)


# In[10]:


dataset_folder ='C:/Users/home/Dev/opencvtube/src/faces/'
names = []
images = []
forfolder inos.listdir(dataset_folder):
files = os.listdir(os.path.join(dataset_folder, folder))[:150]
    if len(files) < 50:
        continue
    fori, name in enumerate(files):
        ifname.find(".jpg") > -1:
img = cv2.imread(os.path.join(dataset_folder + folder, name))
img = detect_face(img) # detect face using mtcnn and crop to 100x100
            ifimg is not None:
images.append(img)
names.append(folder)
print_progress(i, len(files), folder)


# In[11]:


print("number of samples :", len(names))


# In[65]:


def img_augmentation(img):
h, w = img.shape
center = (w // 2, h // 2)
M_rot_5 = cv2.getRotationMatrix2D(center, 5, 1.0)
M_rot_neg_5 = cv2.getRotationMatrix2D(center, -5, 1.0)
M_rot_10 = cv2.getRotationMatrix2D(center, 10, 1.0)
M_rot_neg_10 = cv2.getRotationMatrix2D(center, -10, 1.0)
M_trans_3 = np.float32([[1, 0, 3], [0, 1, 0]])
M_trans_neg_3 = np.float32([[1, 0, -3], [0, 1, 0]])
M_trans_6 = np.float32([[1, 0, 6], [0, 1, 0]])
M_trans_neg_6 = np.float32([[1, 0, -6], [0, 1, 0]])
M_trans_y3 = np.float32([[1, 0, 0], [0, 1, 3]])
M_trans_neg_y3 = np.float32([[1, 0, 0], [0, 1, -3]])
M_trans_y6 = np.float32([[1, 0, 0], [0, 1, 6]])
M_trans_neg_y6 = np.float32([[1, 0, 0], [0, 1, -6]])

imgs = []
imgs.append(cv2.warpAffine(img, M_rot_5, (w, h), borderValue=(255,255,255)))
imgs.append(cv2.warpAffine(img, M_rot_neg_5, (w, h), borderValue=(255,255,255)))
```

```
imgs.append(cv2.warpAffine(img, M_rot_10, (w, h), borderValue=(255,255,255)))
imgs.append(cv2.warpAffine(img, M_rot_neg_10, (w, h), borderValue=(255,255,255)))
imgs.append(cv2.warpAffine(img, M_trans_3, (w, h), borderValue=(255,255,255)))
imgs.append(cv2.warpAffine(img, M_trans_neg_3, (w, h), borderValue=(255,255,255)))
imgs.append(cv2.warpAffine(img, M_trans_6, (w, h), borderValue=(255,255,255)))
imgs.append(cv2.warpAffine(img, M_trans_neg_6, (w, h), borderValue=(255,255,255)))
imgs.append(cv2.warpAffine(img, M_trans_y3, (w, h), borderValue=(255,255,255)))
imgs.append(cv2.warpAffine(img, M_trans_neg_y3, (w, h), borderValue=(255,255,255)))
imgs.append(cv2.warpAffine(img, M_trans_y6, (w, h), borderValue=(255,255,255)))
imgs.append(cv2.warpAffine(img, M_trans_neg_y6, (w, h), borderValue=(255,255,255)))
imgs.append(cv2.add(img, 10))
imgs.append(cv2.add(img, 30))
imgs.append(cv2.add(img, -10))
imgs.append(cv2.add(img, -30))
imgs.append(cv2.add(img, 15))
imgs.append(cv2.add(img, 45))
imgs.append(cv2.add(img, -15))
imgs.append(cv2.add(img, -45))

    returnimgs


# In[12]:


plt.imshow(images[160], cmap="gray")


# In[67]:


img_test = images[0]
augmented_image_test = img_augmentation(img_test)
plt.figure(figsize=(15,10))
fori, img in enumerate(augmented_image_test):
plt.subplot(4,5,i+1)
plt.imshow(img, cmap="gray")
plt.show()


# In[68]:


augmented_images = []
augmented_names = []
fori, img in enumerate(images):
    try:
augmented_images.extend(img_augmentation(img))
augmented_names.extend([names[i]] * 20)
    except:
        print(i)


# In[69]:


len(augmented_images), len(augmented_names)
```

```
# In[70]:


images.extend(augmented_images)
names.extend(augmented_names)


# In[71]:


len(images), len(names)


# In[72]:


unique, counts = np.unique(names, return_counts= True)
foritem in zip(unique, counts):
    print(item)


# In[73]:


def print_data(label_distr, label_name):
plt.figure(figsize=(12,6))
my_circle = plt.Circle( (0,0), 0.7, color='white')
plt.pie(label_distr, labels=label_name, autopct='%1.1f%%')
plt.gcf().gca().add_artist(my_circle)
plt.show()

unique = np.unique(names)
label_distr = {i:names.count(i) fori innames}.values()
print_data(label_distr, unique)


# In[74]:


n = 1000
def randc(labels, l):
    returnnp.random.choice(np.where(np.array(labels) == l)[0], n, replace=False)
mask = np.hstack([randc(names, l) forl innp.unique(names)])


# In[75]:


names = [names[m] form inmask]
images = [images[m] form inmask]


# In[76]:


label_distr = {i:names.count(i) fori innames}.values()
print_data(label_distr, unique)
```

```
# In[77]:


len(names)


# In[78]:


le = LabelEncoder()
le.fit(names)
labels = le.classes_
name_vec = le.transform(names)
categorical_name_vec = to_categorical(name_vec)


# In[79]:


print("number of class :", len(labels))
print(labels)


# In[80]:


print(name_vec)


# In[81]:


print(categorical_name_vec)


# In[82]:


x_train, x_test, y_train, y_test = train_test_split(np.array(images, dtype=np.float
32),   # input data
np.array(categorical_name_vec),       # target/output data
                                                test_size=0.15,
                                                random_state=42)


# In[83]:


print(x_train.shape, y_train.shape, x_test.shape,  y_test.shape)


# In[84]:


x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], x_train.shape[2], 1)
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], x_test.shape[2], 1)
```

```
# In[85]:


x_train.shape, x_test.shape


# In[86]:


def cnn_model(input_shape):
model = Sequential()

model.add(Conv2D(64,
(3,3),
                 padding="valid",
                 activation="relu",
                 input_shape=input_shape))
model.add(Conv2D(64,
(3,3),
                 padding="valid",
                 activation="relu",
                 input_shape=input_shape))

model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Conv2D(128,
(3,3),
                 padding="valid",
                 activation="relu"))
model.add(Conv2D(128,
(3,3),
                 padding="valid",
                 activation="relu"))
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(128, activation="relu"))
model.add(Dense(64, activation="relu"))
model.add(Dense(len(labels)))  # equal to number of classes
model.add(Activation("softmax"))

model.summary()

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


# In[87]:


input_shape = x_train[0].shape
EPOCHS = 10
BATCH_SIZE = 32
```

```
model = cnn_model(input_shape)
history = model.fit(x_train,
y_train,
                    epochs=EPOCHS,
                    batch_size=BATCH_SIZE,
                    shuffle=True,
                    validation_split=0.15   # 15% of train dataset will be used as
validation set
)


# In[88]:


def evaluate_model_(history):
names = [['accuracy', 'val_accuracy'],
['loss', 'val_loss']]
    forname innames :
fig1, ax_acc = plt.subplots()
plt.plot(history.history[name[0]])
plt.plot(history.history[name[1]])
plt.xlabel('Epoch')
plt.ylabel(name[0])
plt.title('Model - '+ name[0])
plt.legend(['Training', 'Validation'], loc='lower right')
plt.grid()
plt.show()

evaluate_model_(history)


# In[89]:


model.save("model-cnn-facerecognition.h5")


# In[90]:


y_pred=model.predict(x_test)


# In[91]:


def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    ifnormalize:
cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
plt.figure(figsize=(8, 8))

plt.imshow(cm, interpolation='nearest', cmap=cmap)
plt.title(title)
plt.colorbar()
```

```
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)
fmt = '.2f' ifnormalize else 'd'
thresh = cm.max() / 2.
    fori, j initertools.product(range(cm.shape[0]), range(cm.shape[1])):
plt.text(j, i, format(cm[i, j], fmt),
                horizontalalignment="center",
                color="white" ifcm[i, j] > thresh else "black")
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()


# In[92]:


# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test.argmax(axis=1), y_pred.argmax(axis=1))
np.set_printoptions(precision=2)


# Plot non-normalized confusion matrix
plot_confusion_matrix(cnf_matrix, classes=labels,normalize=False,
                      title='Confusion matrix')


# In[93]:


print(classification_report(y_test.argmax(axis=1),
y_pred.argmax(axis=1),
                            target_names=labels))


# In[94]:


fromkeras.models importload_model


# In[95]:


def draw_ped(img, label, x0, y0, xt, yt, color=(255,127,0), text_color=(255,255,255
)):
(w, h), baseline = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.5, 1)
cv2.rectangle(img,
(x0, y0 + baseline),
(max(xt, x0 + w), yt),
color,
                  2)
cv2.rectangle(img,
(x0, y0 - h),
(x0 + w, y0 + baseline),
color,
-1)
```

```
cv2.putText(img,
label,
(x0, y0),
cv2.FONT_HERSHEY_SIMPLEX,
                0.5,
text_color,
                1,
cv2.LINE_AA)
    returnimg


# In[97]:


# --------- load Haar Cascade model -------------
face_cascade = cv2.CascadeClassifier('C:/Users/home/Dev/opencvtube/src/cascades/dat
a/haarcascade_frontalface_default.xml')
# --------- load Keras CNN model -------------
model = load_model('C:/Users/home/Dev/opencvtube/src/model-cnn-facerecognition.h5')
print("[INFO] finish load model...")
cap = cv2.VideoCapture(0)
whilecap.isOpened() :
ret, frame = cap.read()
frame = cv2.flip(frame,1)
    ifret:
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.1, 5)
        for(x, y, w, h) infaces:

face_img = gray[y:y+h, x:x+w]
face_img = cv2.resize(face_img, (50, 50))
face_img = face_img.reshape(1, 50, 50, 1)

result = model.predict(face_img)
idx = result.argmax(axis=1)
confidence = result.max(axis=1)*100
            ifconfidence > 80:
label_text = "%s(%.2f %%)"% (labels[idx], confidence)
            else:
label_text = "N/A"
frame = draw_ped(frame, label_text, x, y, x + w, y + h, color=(0,255,255), text_col
or=(50,50,50))

cv2.imshow('Detect Face', frame)
    else:
        break
    ifcv2.waitKey(10) == ord('q'):
        break

cv2.destroyAllWindows()
cap.release()


# In[ ]:


cv2.destroyAllWindows()
cap.release()
```

```
# In[ ]:


cap = cv2.VideoCapture(0)
my_name = "lee"
os.mkdir(dataset_folder + my_name)
num_sample = 500
i = 0
whilecap.isOpened():
ret, frame = cap.read()

    ifret :
cv2.imshow("Capture Photo", frame)
cv2.imwrite("C:/Users/home/Dev/opencvtube/src/faces/%s/%s_%04d.jpg"%  (my_name, my_
name, i), cv2.resize(frame, (250,250)))

        ifcv2.waitKey(100) == ord('q') ori == num_sample:
            break
i += 1
cap.release()
cv2.destroyAllWindows()


# In[ ]:




# In[ ]:




# In[ ]:
```