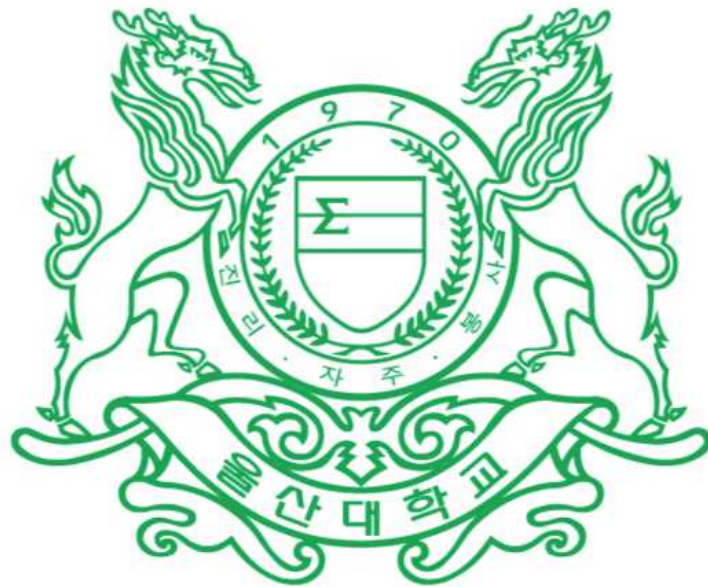


# 졸업작품개발보고서

딥러닝 기반 비대면 시험 부정행위 방지 시스템



학 과

IT융합학부

담당교수

김대환 교수님

팀원

김룡하 20182083

김태형 20182095

부원국 20182115

# 졸업작품개발보고서

## <목차>

1. 작품명 .....	3
2. 작품 개요 .....	3
3. 개발 배경 및 목적 .....	4
3.1. 졸업작품 주제 선정 배경 .....	4
3.2. 졸업작품의 필요성 .....	5
3.3. 졸업작품의 개발 목적 .....	7
4. 졸업작품 내용 .....	8
4.1. 시스템 구성 .....	8
4.2. 주요 기능 흐름도 .....	12
4.3. 개발환경 .....	13
4.4. 데이터 전처리 및 모델 학습 .....	13
4.5. 기능 구성 .....	26
4.6. 사용법 .....	32
4.7. 기능구현 .....	38
4.8. 함수별 기능 .....	42
4.9. 기존 연구와 결과 비교 .....	43
5. 제작 일정 .....	44
6. 향후 계획과 기대효과 .....	45

## 1. 작품명

국문 : 딥러닝 기반 비대면 시험 부정행위 방지 시스템

영문 : DEEP LEARNING-BASED NON-FACE TEST ILLEGAL ACT

## 2. 작품 개요

딥러닝을 활용하여 비대면 시험 중 부정행위를 방지하고 공정한 환경에서 시험이 시행될 수 있도록 하기 위해 본 작품의 주제를 ‘딥러닝 기반 부정행위 방지 시스템’으로 선정하였습니다. 코로나 19 이후 비대면 시험에 대한 수요가 증가하였지만, 비대면의 약점을 이용한 부정행위가 잇따라 발생하였습니다. 이로 인해 비대면 시험의 공정성 문제가 제기되면서 비대면 시험의 신뢰도가 하락하게 되었습니다. 따라서 우리는 비대면 시험이 공정한 환경에서 시행되어 비대면 시험에 대한 신뢰도를 향상을 위해 본 작품을 설계하였습니다.

본 작품의 주요 기능은 크게 세 가지로 분류됩니다.

첫 번째, 얼굴인증 기능으로 DeepFace를 이용하여 Database 내의 이미지를 비교하고, 얼굴 이미지 간의 유사도를 측정하여 얼굴인증 기능을 구현하였습니다.

두 번째, 동공 움직임 탐지를 통한 이상행동 분류 기능으로 MLP 모델을 활용하여 사용자의 고개 방향을 예측합니다. 사용자의 고개 방향이 정면을 향하고 동공의 움직임이 기준을 벗어나게 된다면 이상행동으로 분류합니다.

세 번째, 시계열 데이터를 활용한 고개방향 이상탐지 기능으로 LSTM 모델을 활용하여 사용자의 고개 방향이 기준치에서 5초 이상 벗어난 경우를 이상탐지하고, MLP 모델을 활용하여 이상탐지된 사용자의 고개 방향을 예측합니다.

본 작품의 주요 기능에 편의성을 더하기 위해 사용자 인터페이스, 이상행동 알림 시스템과 결합하여 구현했습니다.

### 3. 개발 배경 및 목적

#### 3.1. 졸업작품 주제 선정 배경

UCLASS 공지사항 : “교내 원격수업 및 교류대학 원격수업의 중간고사/기말고사 부정행위 관련 안내”

원격수업 시험 부정행위 관련 안내

1. 교내 원격수업 및 교류대학 원격수업의 중간고사/기말고사 응시와 관련하여, 에브리타임과 카카오톡 오픈채팅방 등의 수단을 활용하여 사람을 모집하고 역할을 분배하여 시험 문제를 같이 풀이하는 상황이 계속 확인되고 있습니다.

원격교육지원센터에서는 본 사안에 대하여 심각하게 받아들이고 있으며, 부정행위 가담 및 기도(모의)에 대한 모니터링을 진행하고 있습니다.

...

3. 아울러 다음과 같이 학생 징계가 진행되었거나 부정행위에 대한 사실확인이 진행되고 있음을 말씀드립니다.

- 2022학년도 2학기 <○○○○> 과목 중간고사 부정행위로 7일 근신 및 해당 과목 F학점 처리(해당자 : n명)

...

\* 위에 명시한 사안들 이외에 추가로 확인되는 사안 및 인원들에 대해 조사 진행 예정(중간고사에 대해서도 확인 시 조사 및 징계 예정)

부정행위에 있어 발생 후 조치를 취하게 되기보다 예방이 되어 아무런 조치가 되지 않기를 바라며 학생 여러분의 양심을 지켜주시기를 간곡히 부탁드립니다.

출처 : (<https://ulms.ulsan.ac.kr/mod/ubboard/article.php?id=1&bwid=317151>)

**인터뷰 : 울산대학교 원격지원팀**

Q : 현재 “부울경 대학 이러닝 지원센터”에서 울산대학교만 대면시험을 진행하고 있는데 해당 이유에 대해서 알 수 있을까요?

A : 네. 에브리타임 게시판을 통해서 모집한 인원들이 오픈 채팅방, PC실 등 모여서 시험을 응시한다는 게시글이 올라왔다고 신고가 들어왔었습니다. 그래서 오픈 채팅방에 입장해보고, IP주소의 시간대를 확인하여 실제로 단체로 시험을 응시하는 것을 확인했습니다.

Q : 부정행위를 방지하기 위해 어떤 대안을 강구하셨나요?

A : 일단 ULMS에서 카피킬러를 사용해서 표절률을 확인하여 부정행위를 예방하였고, 최종적으로는 GELC에게 울산대학교의 시험 방식을 비대면에서 대면으로 변경하도록 요청하여 대면시험을 응시할 수 있도록 하였습니다.

**[졸업작품 주제 선정 이유]**

울산대학교에서 실시하는 교내 원격수업 및 교류대학 원격수업 과정에서 부정행위가 잇따라 속출하고 있습니다. 현재 “부울경 대학 이러닝 지원센터 (GELC)”에서도 경남권 대학 중 울산대학교만 유일하게 대면으로 시험을 치르고 있습니다. 울산대학교 재학생으로서 교내 비대면 시험 부정행위 문제가 심각하다는 것을 인지하였고, 본 졸업작품을 통해 앞서 인지한 문제점을 해결하고 싶어 본 졸업작품의 주제를 ‘딥러닝 기반 부정행위 방지 시스템’으로 선정하게 되었습니다.

**3.2. 졸업작품의 필요성**

**기사 : “국제무역사 1급시험 온라인,비대면 방식으로 바뀐다”**

한국무역협회(KITA, 회장 구자열)가 주관하는 ‘국제무역사 1급 자격시험’이 비대면 온라인 시험 방식(IBT: Internet-based Test)으로 바뀐다.

온라인 시험은 응시자가 시험장이 아닌, 독립된 개인 공간에서 카메라가 탑재된 PC와 스마트폰을 통해 원격 감독 하에 시험을 치르는 방식이다.

...

무역협회는 온라인 시험을 통해 연간 국제무역사 시험 시행 횟수를 2회에서 5회로 확대하고, 합격자 발표 기간을 기존의 17일에서 6일로 단축하는 등 응시자 편의도 개선했다고 밝혔다.

....

출처 : 한국무역 협회

([https://www.kita.net/cmmrcInfo/cmmrcNews/cmmrcNews/cmmrcNewsDetail.do?JSESSIONID\\_KITA=342795A6D931DA1D920DCBB3F657E50C.Hyper?pageIndex=1&nIndex=74716&sSiteid=1](https://www.kita.net/cmmrcInfo/cmmrcNews/cmmrcNews/cmmrcNewsDetail.do?JSESSIONID_KITA=342795A6D931DA1D920DCBB3F657E50C.Hyper?pageIndex=1&nIndex=74716&sSiteid=1))

기사 : “[단독] 비대면은 비리의 장?…○○대 또 부정행위”

재학생들이 잇따라 신종 코로나바이러스 감염증(코로나19)에 확진되면서 학교 수업을 전면 비대면으로 전환한 ○○대에서 또 한 번 부정행위가 적발돼 공분을 사고 있다

....

해당 시험은 부정행위를 방지하기 위해 화상 회의 플랫폼 줌(zoom)을 켜 교수와 조교가 시험을 치는 학생들을 실시간으로 감독할 수 있게 진행됐다. 수강생마다 총 두대의 기기를 이용했는데 컴퓨터로는 얼굴을, 스마트폰으로는 책상과 몸을 비추게 해 부정행위를 예방하려 한 것이다.

하지만 당시 시험에 임했던 복수의 수강생들에 따르면 이 같은 조치에도 일부 수강생들은 비대면 시험의 취약성을 악용해 화면이 비치지 않는 곳에서 마우스를 사용하고 온라인 검색을 하는 등 부정행위를 저질렀다.

....

출처 : 서울경제 (<https://www.sedaily.com/NewsView/1ZAL14GBLW>)

기사 : “[단독] 비대면은 비리의 장?…○○대 또 부정행위”

재학생들이 잇따라 신종 코로나바이러스 감염증(코로나19)에 확진되면서 학교 수업을 전면 비대면으로 전환한 ○○대에서 또 한 번 부정행위가 적발돼 공분을 사고 있다

....

해당 시험은 부정행위를 방지하기 위해 화상 회의 플랫폼 줌(zoom)을 켜 교수와 조교가 시험을 치는 학생들을 실시간으로 감독할 수 있게 진행됐다. 수강생마다 총 두대의 기기를 이용했는데 컴퓨터로는 얼굴을, 스마트폰으로는 책상과 몸을 비추게 해 부정행위를 예방하려 한 것이다.

하지만 당시 시험에 임했던 복수의 수강생들에 따르면 이 같은 조치에도 일부 수강생들은 비대면 시험의 취약성을 악용해 화면이 비치지 않는 곳에서 마우스를 사용하고 온라인 검색을 하는 등 부정행위를 저질렀다.

....

출처 : 서울경제 (<https://www.sedaily.com/NewsView/1ZAL14GBLW>)

## [비대면 시험의 공정성 문제 해결의 필요성]

코로나19 이후 비대면 시스템의 보급으로 인해 비대면 시험의 수요가 증가함에 있습니다. 하지만, 원격 감독의 어려움, 기술적인 취약점 등 다양한 원인으로 인하여 부정행위가 속출하여 시험의 공정성이 훼손되어 학생들 간의 불평등이 발생하였습니다.

이런 문제점들로 인해 비대면 시험의 신뢰도가 하락하여 기존에 진행하던 비대면 시험을 유지하기가 어려워 대면으로 전환하는 경우도 발생하였습니다. 따라서, 공정하고 신뢰성 있는 평가를 위해서는 부정행위를 탐지하고 예방할 수 있는 시스템의 도입이 필수적입니다.

핵심 기술인 딥러닝을 접목하여 부정행위 패턴 감지 및 대응에 활용한다면 응시자의 부정행위 검출을 더욱 쉽고 빠르게 할 수 있습니다. 이를 통해 비대면 시험의 공정성을 향상시킬 수 있고, 학생들 간의 불평등도 해소할 수 있습니다.

### 3.3. 졸업작품의 개발 목적

본 졸업작품의 목적은 비대면 시험의 공정성 문제를 딥러닝을 활용하여 해결하고 학생들 간의 불평등을 개선하는 것입니다. 딥러닝을 활용한 부정행위 방지 시스템을 개발하여 학생들에게 공정한 시험 환경을 제공하고, 시험 결과의 신뢰도를 확보하는 것을 목표로 합니다.

딥러닝은 이미지와 영상 데이터를 처리하고 패턴을 학습하는 데에 강점을 지니고 있습니다. 따라서, 시험 중 부정행위를 탐지하는 데에 유용하게 활용될 수 있습니다. 본 졸업작품은 본인인증과 시선과 고개 방향의 이상 탐지를 딥러닝을 활용하여 구현하였습니다. 이를 통해 시험 중 부정행위를 신속하게 탐지하고 예방할 수 있습니다.

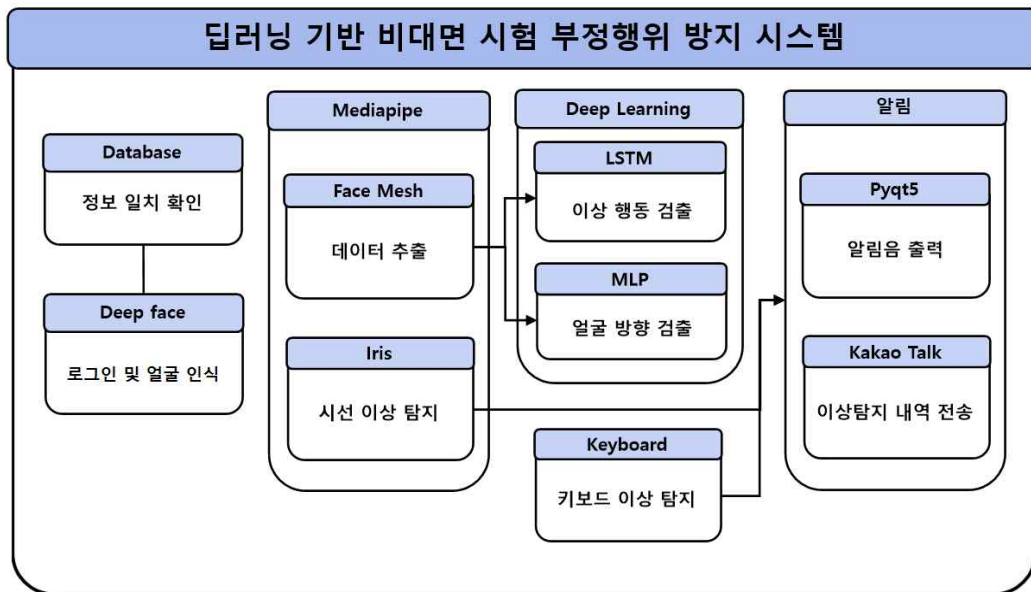
작품의 특징점은 LSTM을 활용하여 응시자의 시선이 화면 밖에 특정 시간 이상 머무르게 되었을 때를 이상행동으로 분류하였다는 점입니다. 이상행동이라고 분류되기 위해서는 사용자가 시험지 기준에서 벗어나더라도 특정 시간 이상 지정 기준치에서 벗어나야 합니다. 이를 통해 억울한 상황을 최소화하고 공정한 환경에서 시험을 응시할 수 있도록 합니다.

이렇게 구현된 ‘딥러닝 기반 부정행위 방지 시스템’으로 비대면 시험의 부정행위로 인한 불평등을 최소화하여, 공정한 환경에서 시험을 응시할 수 있고, 시험 결과의 신뢰도를 확보할 수 있도록 하는 것을 목표로 하고 있습니다.

## 4. 졸업작품 내용

### 4.1. 시스템 구성

○ 소프트웨어 구성



‘딥러닝 기반 비대면 시험 부정행위 방지 시스템’은 다음과 같은 소프트웨어 구성으로 이루어져 있습니다.

- 사용자의 본인인증 : 사용자의 신원을 확인하고 대리 시험을 방지하기 위해 본인인증 기능이 구현되어 있습니다. 얼굴 인식을 통해 사용자를 식별하고 인증하는 과정을 거칩니다.
- 시험 간 이상행동 탐지 : 딥러닝을 활용하여 시험 중 부정행위를 탐지하는 기능이 구현되어 있습니다. 사용자의 시선과 고개 방향의 이상행동을 학습한 딥러닝 모델을 활용하여 검출합니다.
- 알림 및 로그 데이터 전송 : 부정행위가 탐지되면 시스템에서 알림을 줘 사용자에게 경고합니다. 또, 해당 내용을 시험관리자에게 실시간으로 전송하여 부정행위 탐지 내용을 즉각적으로 파악할 수 있습니다.

이를 통해 사용자가 시험을 치르는 동안 대리 시험, 시험 화면 외의 다른 화면 참고 등, 부정행위라고 판단될 수 있는 요소를 실시간으로 파악하고 경고 알림을 줘, 사용자의 부정행위를 방지하는 프로그램입니다.



#### 4.1.1. 소프트웨어 구현 도구 및 기술

1. 사용자 인터페이스 : 사용자 인터페이스 구성에 PyQt5를 활용하였습니다. PyQt5는 Python 프로그램 언어를 위한 Qt 프레임워크의 파이썬 바인딩으로, GUI 애플리케이션 개발에 사용됩니다. PyQt5 활용을 통해 본 프로그램의 개발언어인 파이썬과의 통합성을 강화하고 개발 과정을 간소화하여 생산성을 향상시켰습니다. 또한, 크로스 플랫폼 지원을 통해 사용자들이 다양한 운영체제에서 사용하더라도 일관된 경험을 제공할 수 있도록 하였습니다. Qt Designer와의 통합으로 사용자 인터페이스를 시각적으로 설계하여 사용자의 편의성을 높였습니다.

본 졸업작품에서는 로그인, 시험 화면, 로그 데이터 출력 총 3개의 페이지로 구성하였습니다. 페이지에 필요한 UI 요소를 생성하고 이벤트 처리 함수를 통해 원하는 동작과 기능을 수행하도록 하였습니다.



2. 데이터베이스 : Sqlite3 모듈을 활용하여 데이터베이스를 구성하였습니다. 해당 모듈은 파이썬에서 데이터베이스를 다루기 위해 제공하는 내장 모듈로, 데이터베이스의 구축, 연결, 쿼리 실행, 결과 처리 등 다양한 기능을 수행할 수 있습니다.

파이썬과의 통합성으로 데이터베이스와의 상호작용을 편리하게 처리할 수 있으며, 강력한 SQL 쿼리 기능을 활용하여 데이터 삽입, 조회, 갱신, 삭제할 수 있습니다. 또한, 파이썬 내장 모듈로 제공되어 별도의 서버 구축 과정이 필요하지 않아 개발 과정을 간소화하였습니다. Sqlite3의 단일파일 데이터베이스 특성을 활용하여, 데이터베이스 생성 및 공유를 간편하고 효율적으로 처리하였습니다.

본 졸업작품에서는 사용자가 인터페이스에 입력한 로그인 정보와 얼굴인증 모델을 통한 예측값을 데이터베이스의 정보와 비교하여 일치 여부를 판단할 수 있도록 하였습니다. 이를 통해 안전하고 정확한 인증 절차를 수행할 수 있었고, 데이터베이스와의 상호작용을 간편하게 처리할 수 있었습니다.



3. DeepFace : 얼굴 인식 및 분석을 위한 딥러닝 기반 오픈소스 라이브러리인 Deep Face를 사용합니다. Facebook AI Research에서 개발되었으며, 이미지나 비디오에서 얼굴을 인식하고, 얼굴의 특징을 추출하여 얼굴 감지, 얼굴 인식, 유사도 측정 등 다양한 얼굴 관련 작업을 수행합니다. 대규모 데이터셋을 통해 학습되어, 다양한 환경에서 얼굴 인식에 강점을 보여 얼굴 유사도 측정을 정확히 수행할 수 있습니다. 또, GPU 가속을 지원하여 실시간 얼굴 인식 작업을 빠르게 처리할 수 있습니다.

본 졸업작품에서는 GPU 가속을 활용하여 얼굴 인식 작업을 빠른 속도로 얼굴 인식 작업을 수행하며, 데이터베이스 내의 이미지 데이터와 입력된 이미지 데이터 간의 유사도를 측정하여 대리 시험을 방지하는 본인인증 과정에서 활용이 됩니다.



4. Mediapipe : Google에서 개발한 오픈소스 프레임워크로, 컴퓨터 비전과 머신러닝을 기반으로 한 실시간 응용프로그램 개발에 유용하게 사용됩니다. 이 프레임워크는 다양한 비주얼 컴퓨팅 작업을 수행하는데 활용되며, 주로 비디오 데이터에서 객체 감지, 추적, 포즈 추정, 얼굴 감지 및 인식, 손 감지, 자세 추정 등 다양한 작업을 실시간으로 처리할 수 있습니다. 또, 최적화된 알고리즘과 기술을 통해 빠른 처리 속도와 정확성을 제공합니다. 이를 통해 실시간 환경에서도 높은 성능을 발휘할 수 있습니다.

Mediapipe는 다양한 모듈과 기능을 제공합니다. 예를 들어, face landmark 모듈은 얼굴 이미지에서 다양한 얼굴 특징 지점을 식별하는데 사용됩니다. 이를 활용하여 얼굴의 형태, 표정 등을 추정할 수 있습니다. 또한, iris 모듈은 눈동자를 실시간으로 추적하고 분석하는 기능을 제공합니다. 이를 활용하여 사용자의 눈동자 움직임을 감지하고 시선 추적 등의 작업을 수행할 수 있습니다.

본 졸업작품에서는 Mediapipe에서 제공하는 face landmark를 활용하여 데이터 전처리 과정에 활용하였고, iris를 활용하여 실시간 눈동자를 추적하는 기능에 활용하였습니다. Mediapipe의 다양한 기능과 알고리즘을 활용하여 실시간 비주얼 컴퓨팅 작업을 수행하였고, 이를 통해 작품에 정확성과 실용성을 더했습니다.



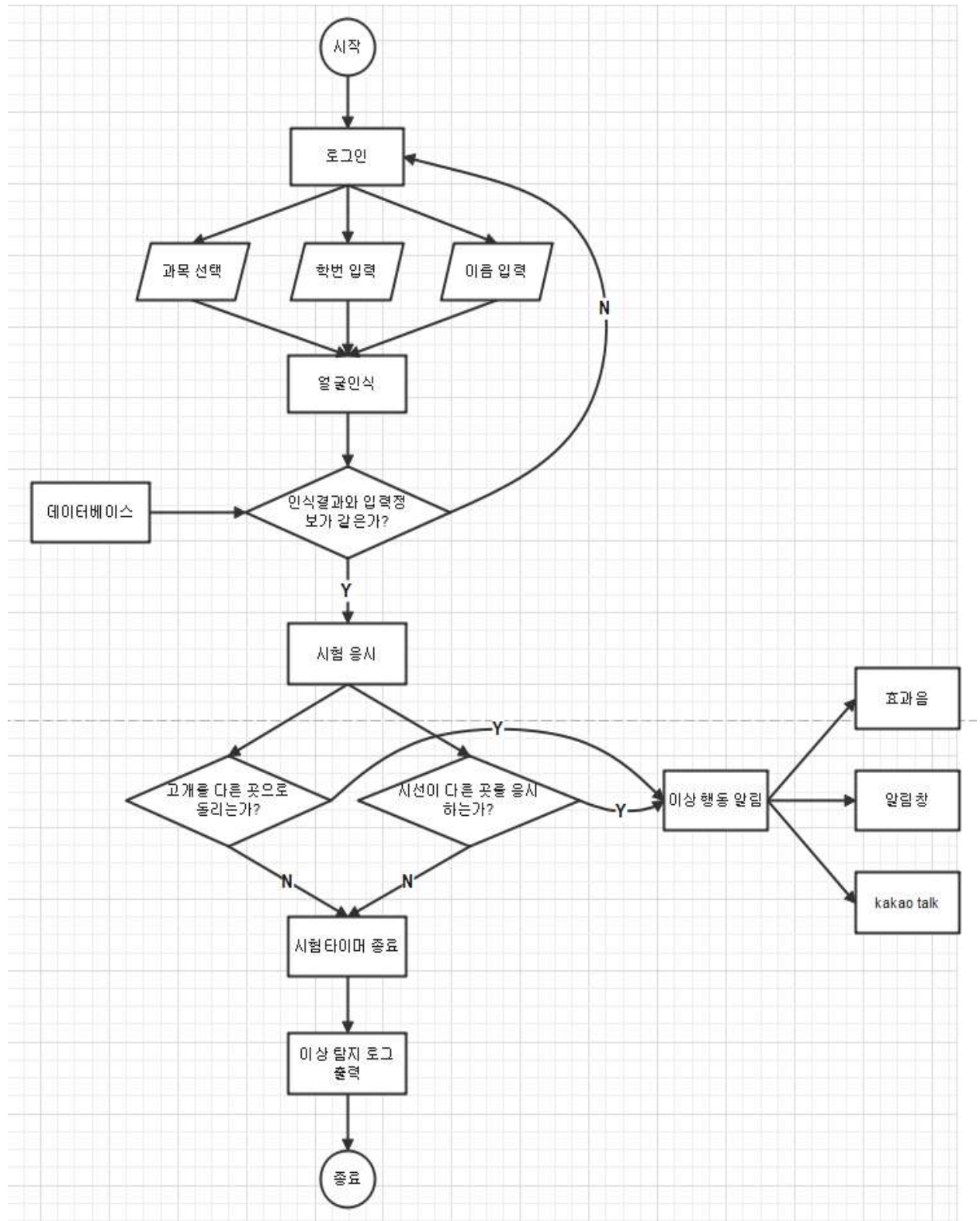
5. LSTM : Long Short-Term Memory의 약어로, 시퀀스 데이터나 시계열 데이터를 처리하기 위해 설계된 순환 신경망(RNN)의 한 종류입니다. LSTM은 단기 기억 문제와 기울기 소실 문제로 인한 장기 의존성을 잘 학습하지 못하는 RNN의 한계를 극복하기 위해 고안되었습니다. 메모리 셀과 게이트 메커니즘을 도입하여 장기 의존성을 모델링하고 정보의 유지 및 추가, 출력 등을 관리할 수 있습니다.

본 졸업작품에서는 LSTM의 장점인 시간적인 의존성을 갖는 데이터를 다루는데 효과적이며, 패턴 학습과 예측에 큰 도움을 줄 수 있다는 점을 활용하여, 입력 데이터인 시계열 데이터를 통해 복잡한 패턴을 학습하고, 정상 동작과 비정상 동작을 이진 분류하는 모델을 구축하여, 이상행동 예측에 활용하였습니다.

6. 다중 페셉트론(MLP) : Multilayer Perceptron은 인공 신경망의 일종으로, 입력층, 은닉층, 출력층 등 여러 개의 층으로 구성되어 있습니다. MLP는 비선형 문제를 해결하기 위해 사용되며, 각 층의 뉴런들은 활성화 함수를 통과해 입력을 처리하고 출력을 생성합니다. 은닉층은 중간 계산을 담당하며, 뉴런의 가중치와 활성화 함수를 사용하여 입력 신호를 변환합니다. 은닉층을 여러 개 사용하는 것으로 신경망의 복잡도를 증가시킬 수 있으며, 이를 통해 더 복잡한 문제를 해결할 수 있습니다.

본 졸업작품에서는 비선형 문제 해결 능력을 활용하여 다양한 패턴을 학습시키고 사람의 머리 회전 방향을 예측하도록 구축하였습니다. MLP는 복잡한 패턴을 학습할 수 있어 입력 데이터의 다양한 특징과 관계를 학습하여 정확한 예측을 수행할 수 있습니다.

## 4.2. 주요 기능 흐름도



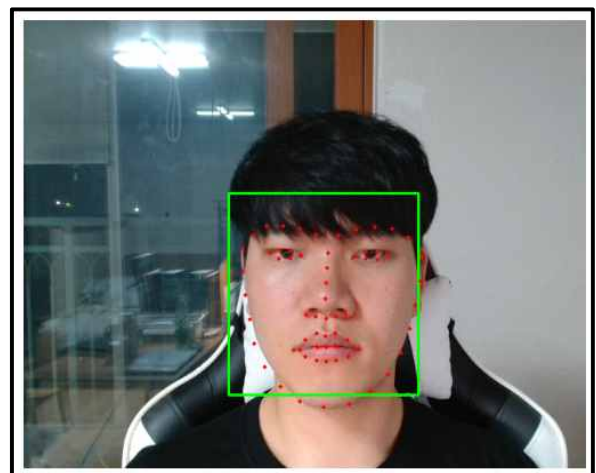
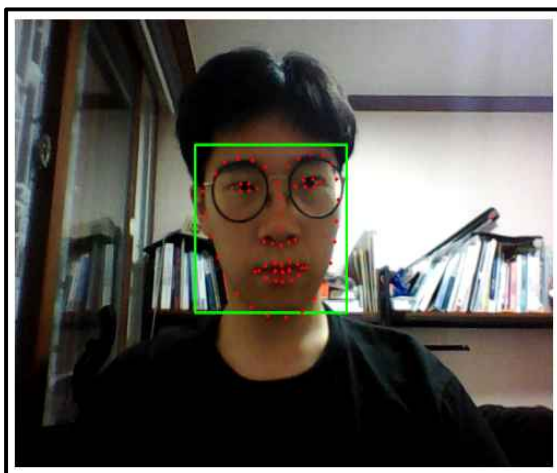
### 4.3. 개발환경

- ▷ 언어 : Python, SQL
- ▷ 개발 도구 : VScode, Jupyter-notebook
- ▷ 라이브러리 : Tensorflow, PyQt5, Mediapipe, scikit-learn, opencv
- ▷ 운영체제 : window
- ▷ 데이터베이스 : Sqlite3
- ▷ 설계도구 : Git, GitMind, Google Drive
- ▷ 딥러닝 개발 환경 : NVIDIA RTX 3080ti, cuda == 11.2, cudnn == 8.1.0

### 4.4. 데이터 전처리 및 모델 학습

#### 4.4.1. dlib과 SVM 모델을 사용한 얼굴 분류

- 1) 이미지 데이터를 dlib 라이브러리의 detector 함수를 사용하여 얼굴을 검출하고, 검출된 얼굴 영역에서 특징점을 예측하기 위해 shape\_predictor 함수를 사용하여 얼굴 영역 내의 눈, 코, 입 등의 특징점을 좌표로 반환합니다.



[ 얼굴 검출과 특징점 좌표 추출 ]

2) 3개의 클래스 (boo, ha, tae)로 분류해 SVM 모델을 학습하고 테스트하였습니다.



[ 3개의 클래스로 분류된 학습 데이터 ]



[ 테스트 데이터 ]

```
Predicted person: tae  
Predicted person: tae  
Predicted person: tae  
Predicted person: tae  
Predicted person: tae
```

[ 테스트 결과 ]

- 3) 10개의 클래스로 증강하여 SVM 모델을 학습하고 같은 테스트 데이터를 이용하여 예측하였습니다.



[ 증강된 클래스의 학습데이터 ]

```
Predicted person: tae  
Predicted person: tae  
Predicted person: tae  
Predicted person: ji  
Predicted person: tae
```

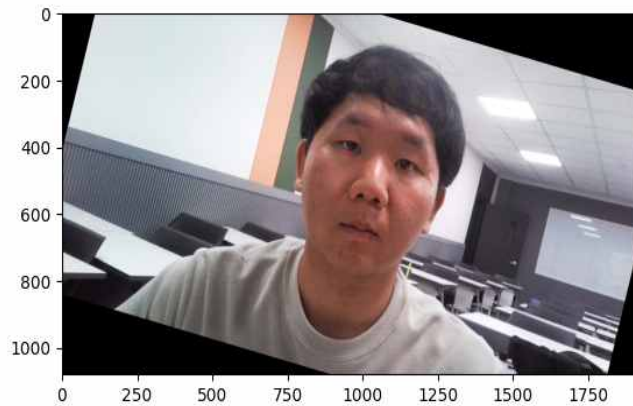
[ 테스트 결과 ]

예측 결과 클래스가 증가하였을 때 기대성능에 미치지 못하는 것을 확인하였습니다.

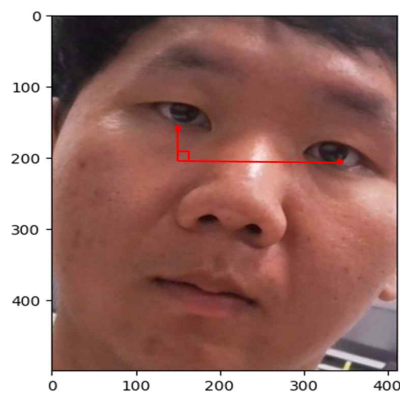


#### 4.4.2. Deepface를 활용한 얼굴 유사도 비교

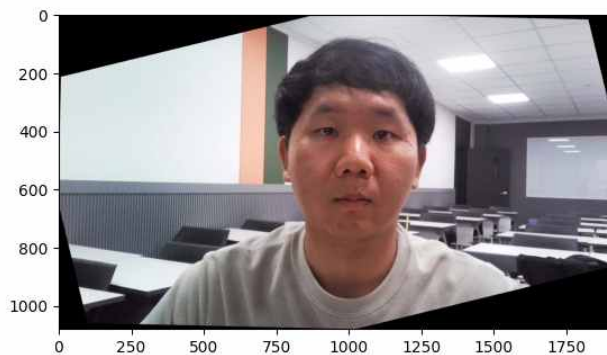
- 1) 얼굴 검출 및 정렬에는 MTCNN 알고리즘을 사용했습니다. 눈의 위치 좌표를 MTCNN 알고리즘을 사용해서 검출합니다.



검출된 눈의 좌표를 유클리드 계산하여 코사인 법칙을 통해 얼굴이 정면을 바라볼 수 있도록 회전하였습니다.



이미지를 회전하여 얼굴 정렬을 진행하였습니다.





2) 입력된 이미지와 데이터베이스 내에 있는 사진들을 'VGG-Face' 모델을 사용하여 사진에서 추출한 얼굴 특징들 사이의 유사도를 계산하여 입력된 이미지와 동일한 이미지를 찾는 방법으로 얼굴 인식을 진행하였습니다.

	identity	VGG-Face_cosine
0	test//boo_6.jpg	0.005768
1	test//boo_7.jpg	0.008322
2	test//boo_4.jpg	0.016961
3	test//boo_9.jpg	0.038976
4	test//boo_3.jpg	0.050683
5	test//boo_2.jpg	0.057389
6	test//boo_8.jpg	0.058217
7	test//boo_10.jpg	0.073329
8	test//boo_5.jpg	0.082877
9	test//boo_1.jpg	0.089866
10	test//tae_18.jpg	0.148959
11	test//tae_3.jpg	0.156467
12	test//tae_21.jpg	0.157475
13	test//tae_6.jpg	0.159176
14	test//tae_12.jpg	0.166475
15	test//tae_13.jpg	0.167354
16	test//tae_4.jpg	0.169674
17	test//tae_7.jpg	0.173286
18	test//tae_22.jpg	0.177080
19	test//tae_5.jpg	0.179089
20	test//tae_1.jpg	0.201633
21	test//ha_4.jpg	0.234563
22	test//ha_5.jpg	0.251493
23	test//tae_2.jpg	0.257885
24	test//ha_7.jpg	0.258254

[ 유사도 계산 결과 ]

VGG-Face\_cosine 검사 0.1 이하의 임계값에서 boo로 분류가 된 것을 볼 수 있습니다.

### 3) 4.4.1.에서 학습한 모델과 비교

기존 4.4.1.에서 학습한 모델을 사용하게 된다면 데이터베이스에 새로운 사람이 추가되었을 때, 추가적인 학습을 해야 한다는 단점이 발생했습니다.

또, 클래스가 늘어나게 되면 예측 정확도가 떨어지는 결과가 발생하였습니다.

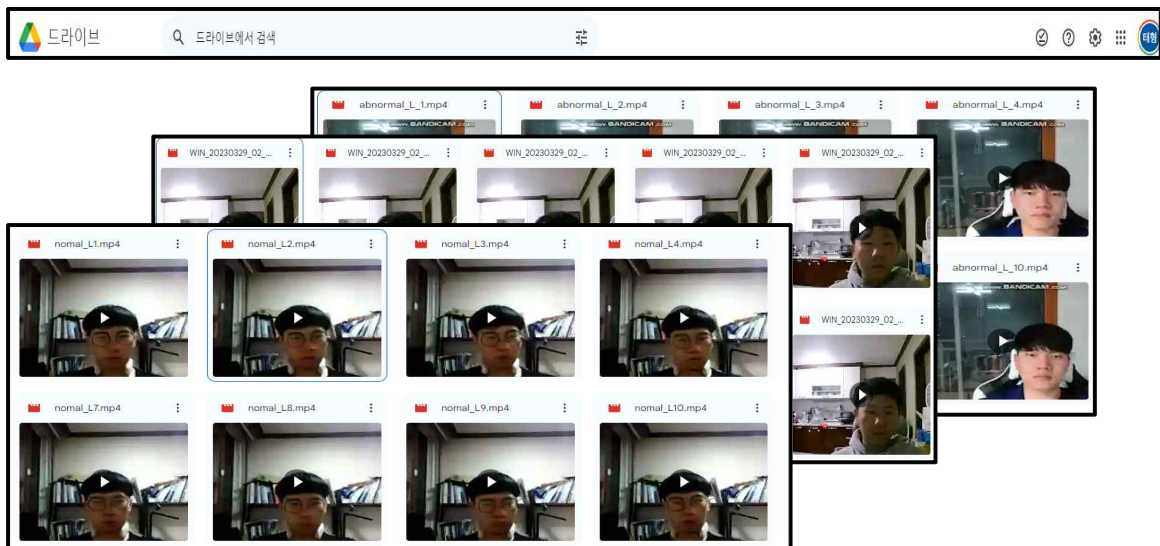
따라서 본 졸업작품에서는 DeepFace를 활용한 얼굴의 유사도 비교 모델을 사용하여 추가적인 학습 과정 없이 데이터베이스 내의 이미지와 입력 이미지간의 비교를 통해 얼굴인증 과정을 수행하였습니다.

### 4.4.3. Mediapipe를 활용한 데이터 전처리

1) 영상 및 이미지 데이터 수집 : 모델 학습을 위해 데이터 수집을 두 분류로 진행하였습니다.

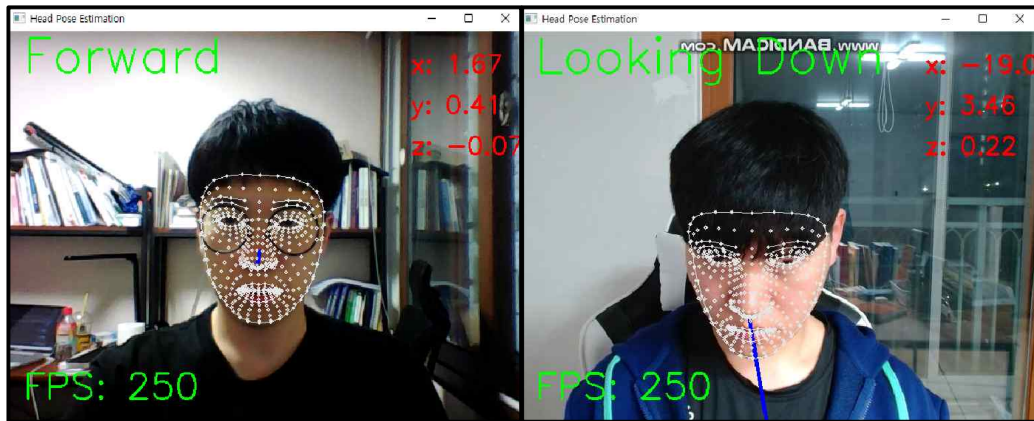
- LSTM 모델의 데이터 : 모니터의 크기를 기준으로 5초 이상 모니터 밖으로 고개 방향이 고정되어있는 경우를 abnormal, 5초 이하의 시간이나 모니터 기준 안쪽으로 고개 방향이 고정되어있는 경우를 normal로 지정하고 영상 데이터를 다양하게 수집하였습니다.

- MLP 모델의 데이터 : 고개 방향을 정면, 오른쪽, 왼쪽, 위, 아래 총 5가지로 분류하여 이미지 데이터를 수집하였습니다. 모니터 크기 비율에 맞춰 오른쪽, 왼쪽의 임계값이 위쪽, 아래쪽 임계값에 비해 크게 설정하였습니다.



[ 모델 학습을 위해 수집된 데이터 ]

- 2) Mediapipe를 활용한 얼굴 랜드마크 추출 : Mediapipe의 FaceMesh 모듈을 사용하여 얼굴 랜드마크 데이터를 추출합니다. 추출된 데이터를 활용하여 얼굴을 감지하고 3D 모델링으로 얼굴의 구조와 특징을 파악합니다. 이를 통해 얼굴의 3D 좌표를 추출합니다.



- 3) 추출된 3D 좌표를 활용한 PnP 문제 해결 : PnP 문제는 3D 공간의 점들과 해당 점들의 2D 투영 좌표 사이의 관계를 이용하여 카메라의 위치 및 자세를 추정하는 컴퓨터 비전 문제입니다. cv2.solvePnP 함수를 사용하여 얼굴의 회전 벡터와 변위 벡터를 계산합니다. 계산된 회전 벡터와 변위 벡터를 이용하여 회전 행렬을 계산합니다.
- 4) 회전 행렬로부터 각도 추출 : 회전 행렬로부터 각도를 추출하기 위해 cv2.Rodrigues 함수를 사용합니다. 해당 함수는 회전 벡터를 회전 행렬로 변환하는 기능을 수행합니다. 이후, cv2.RQDecomp3x3 함수를 사용하여 회전 행렬로부터 각도를 추출합니다. 해당 함수는 회전 행렬을 분해하여 각도와 기타 행렬들을 추출하는 기능을 제공합니다.
- 5) x, y, z 좌표 추출 : 각도 추출 과정을 통해 얼굴의 x, y, z 값이 추출됩니다. x축의 경우 위, 아래 y축의 경우 왼쪽, 오른쪽 z축의 경우 앞, 뒤 방향으로 각 영상과 이미지의 데이터가 전처리 되었습니다. 전처리 된 데이터를 관리가 용이 하도록 csv 파일로 변환하여 저장하였습니다.

이름	x	y	z
abnormal_D_1_TK	-16.58742	2.428898	0.2657155
abnormal_D_2_TK	-16.42876	2.3205696	0.2584536
abnormal_D_3_TK	-15.58971	2.6969491	0.2359369
abnormal_D_4_TK	-16.12653	2.503585	0.2337774
abnormal_D_5_TK	-16.86713	2.8062956	0.2419284
abnormal_D_6_TK	-15.61904	2.8292248	0.2168433
abnormal_D_7_TK	-16.17921	2.7641328	0.202608
abnormal_D_8_TK	-16.26983	2.4751934	0.2047432
abnormal_D_9_TK	-15.5214	2.5902482	0.200926
abnormal_D_10_TK	-15.61831	2.5326676	0.1957536
abnormal_D_11_TK	-15.93225	2.6952782	0.208851
abnormal_D_12_TK	-14.88077	2.3761466	0.1654411

[ 전처리된 csv 파일 ]

#### 4.4.4. MLP 모델 학습

- 1) 전처리 데이터 라벨링 : MLP 모델의 학습을 위해 4.4.3.에서 전처리 된 csv 데이터를 방향별로 Foward, Down, Up, Right, Left로 라벨링 하여 나누었습니다. 나누어진 라벨값들을 각각 0~4의 숫자로 매핑하고, 그 숫자를 one-hot 인코딩하였습니다.

x	y	z	Label
6.6515943	-0.994021	-0.216149	F
6.6414329	-1.16039	-0.212882	F
6.8863553	-1.133071	-0.227781	F
6.8313156	-1.091919	-0.220542	F
6.7597209	-1.116724	-0.220452	F
6.516895	-1.146306	-0.220542	F
6.7596824	-1.171016	-0.230398	F
6.6792446	-1.120042	-0.227646	F
6.6204738	-1.100169	-0.231125	F

x	y	z	Label
166	R	762	L
622	R	473	L
399	R	578	L
594	R	374	L
123	R	055	L
981	R	209	L
731	R	551	L
235	R	584	L
874	R		

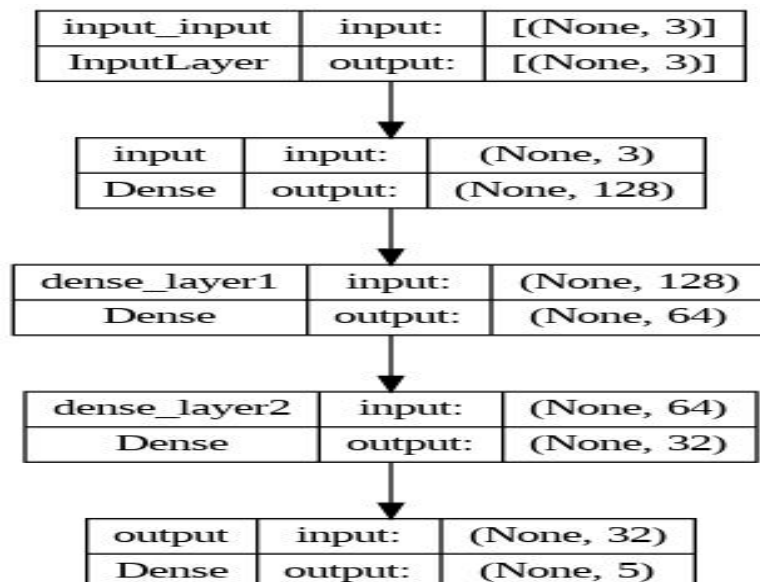
x	y	z	Label
298	L	075	U
773	U	806	U
951	U	477	U
373	U	959	U
548	U	794	U

x	y	z	Label
315	D	901	D
529	D	123	D
829	D	701	D
101	D	155	D
417	D	909	D

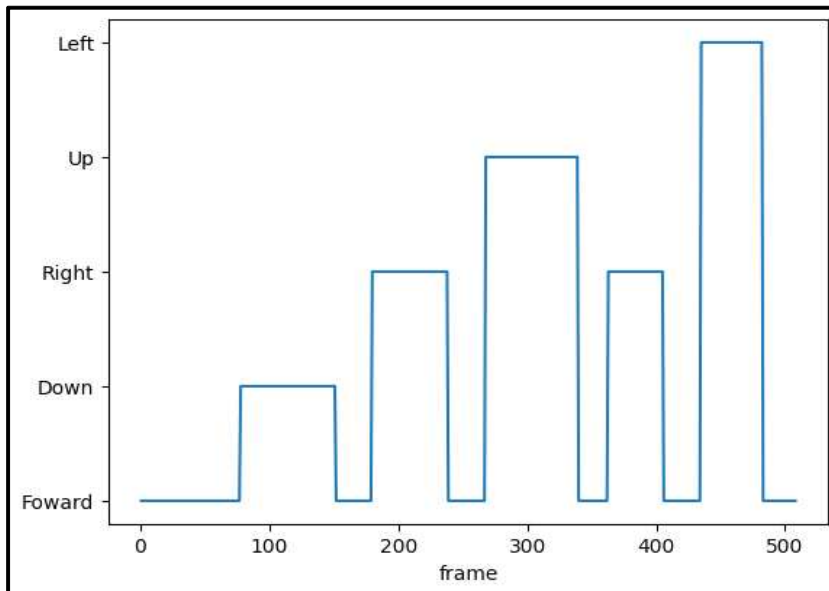
[ MLP 학습을 위해 라벨링된 데이터 ]

- 2) 모델 구축 : 각 레이어를 선형으로 연결하는 Sequential 모델을 생성하고, 입력 데이터의 형태는 x, y, z로 (3, ) 형태로 지정하였습니다. 그리고 'Dense'레이어를 구축하고 마지막 레이어에는 클래스에 개수에 해당하는 5개의 출력형태를 지정하였습니다.



[ MLP 모델 구조도 ]

3) 모델 테스트 : 각 방향별을 응시한 테스트 영상에서 학습한 모델을 적용하였습니다.



방향별로 나눈 임계값에 따라 영상 데이터의 각 프레임의 결과를 시각화한 것입니다.

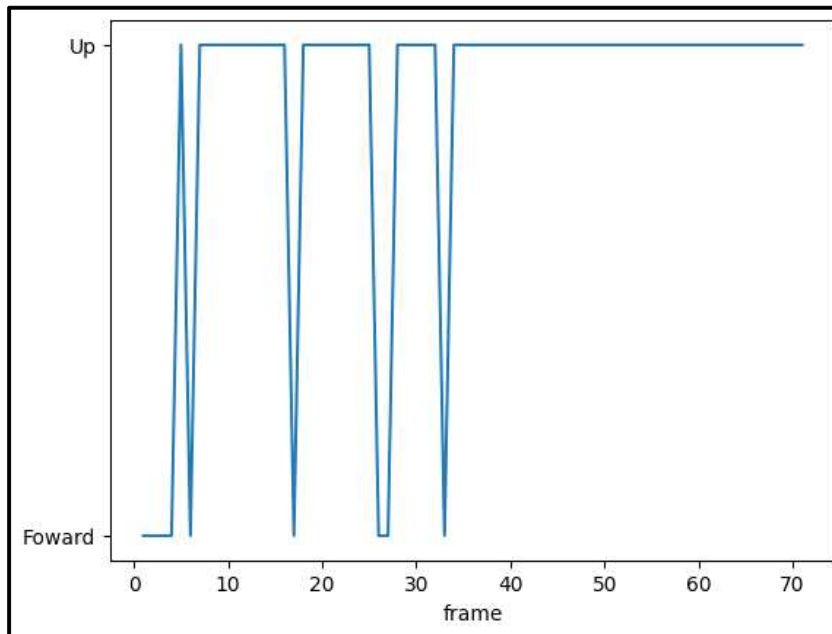
```
n = 50 # 한 줄에 출력할 레이블 개수

for i in range(0, len(predicted_labels), n):
    labels_batch = predicted_labels[i:i+n]
    labels_str = ' '.join(labels_batch)
    print(labels_str)
```

```

FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
FFFFFFFF
```

영상 데이터를 학습한 MLP 모델을 통해 예측한 결과입니다. 예측 정확도가 높게 측정된 것을 볼 수 있습니다.



[ 방향이 제대로 분류되지 못한 데이터 ]

영상 데이터에서 정면을 봤다고 생각했으나 임계값 근처에서 정면과 위쪽 방향 사이에 있어 정확한 방향을 측정할 수 없는 데이터를 MLP 모델을 통해 예측하였습니다.

```

n = 10 # 한 줄에 출력할 레이블 개수
for i in range(0, len(predicted_labels), n):
    labels_batch = predicted_labels[i:i+n]
    labels_str = ' '.join(labels_batch)
    print(labels_str)

```

```

F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F
F F F F F F F F F F

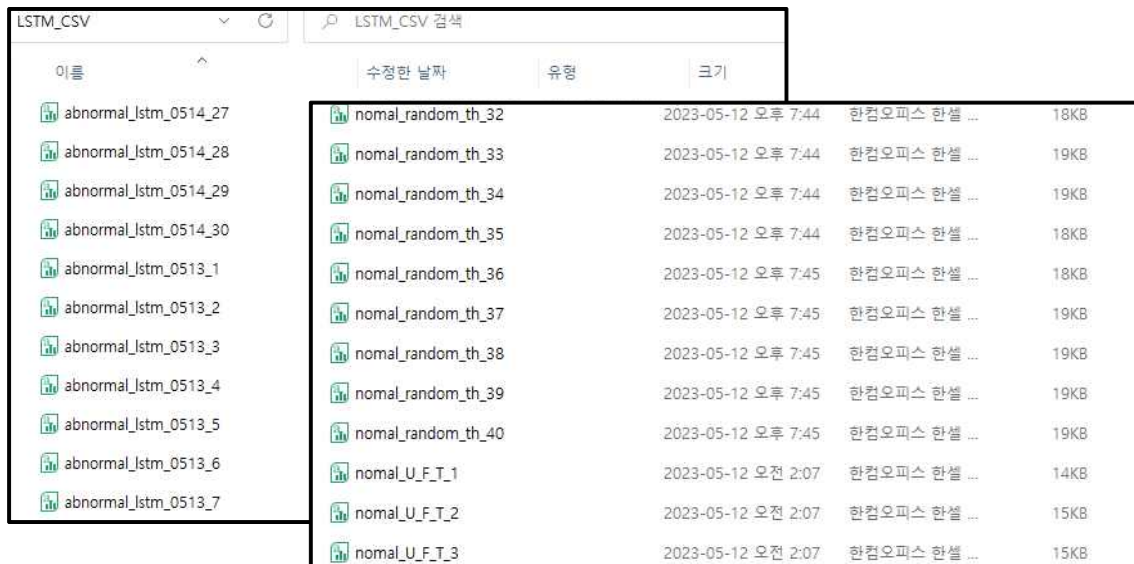
```

학습한 모델을 통해 예측한 결과입니다. 정확한 방향 파악이 힘들었던 데이터에 대해서 Foward로 유의미한 예측 결과값을 도출해냈습니다.



#### 4.4.5. LSTM 모델 학습

- 1) 전처리 데이터 라벨링 : LSTM 모델 학습을 위해 4.4.3.에서 전처리된 csv데이터를 abnormal과 normal로 나누어 각 폴더에 저장하였고 csv를 읽어올 때 이름을 기준으로 normal은 0 abnormal은 1로 라벨링 하였습니다.



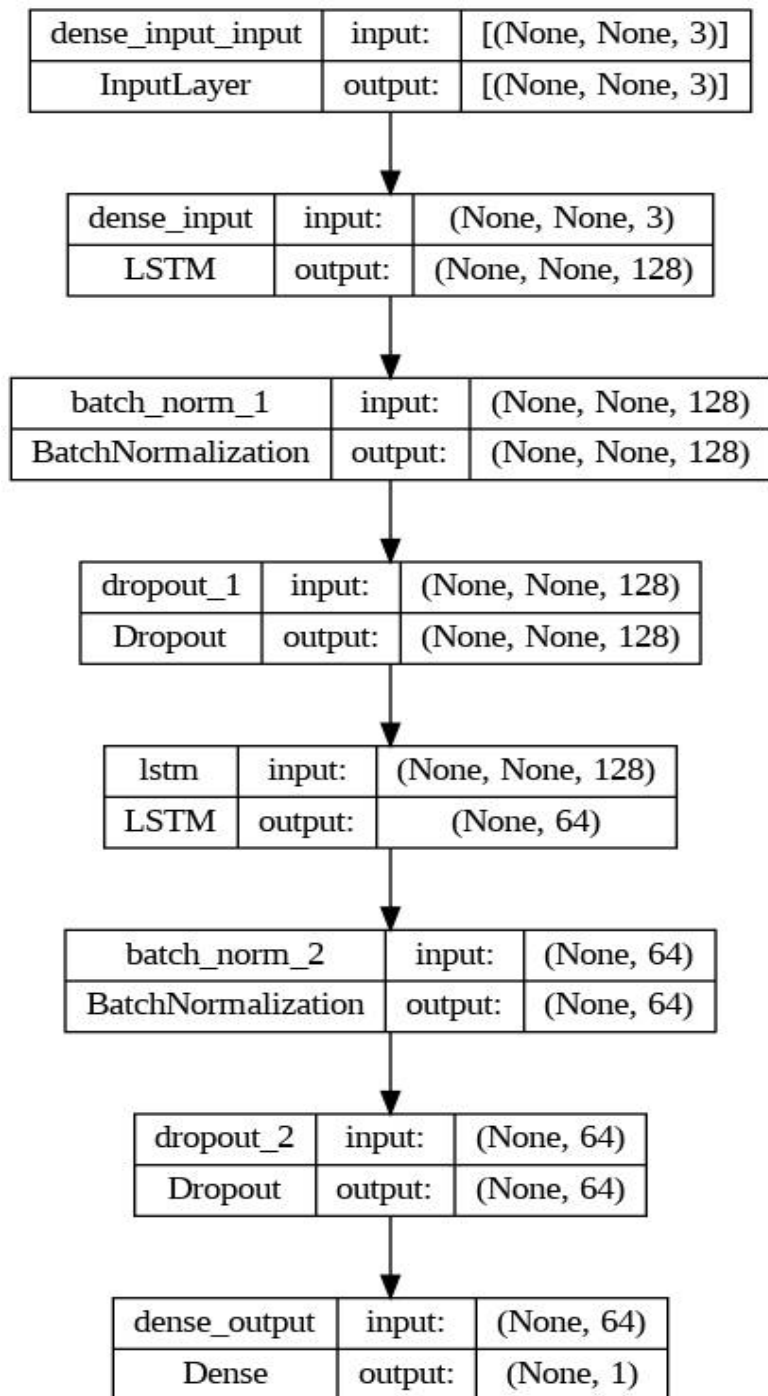
이름	수정된 날짜	유형	크기
abnormal_lstm_0514_27			
abnormal_lstm_0514_28			
abnormal_lstm_0514_29			
abnormal_lstm_0514_30			
abnormal_lstm_0513_1			
abnormal_lstm_0513_2			
abnormal_lstm_0513_3			
abnormal_lstm_0513_4			
abnormal_lstm_0513_5			
abnormal_lstm_0513_6			
abnormal_lstm_0513_7			
normal_random_th_32	2023-05-12 오후 7:44	한컴오피스 한셀 ...	18KB
normal_random_th_33	2023-05-12 오후 7:44	한컴오피스 한셀 ...	19KB
normal_random_th_34	2023-05-12 오후 7:44	한컴오피스 한셀 ...	19KB
normal_random_th_35	2023-05-12 오후 7:44	한컴오피스 한셀 ...	18KB
normal_random_th_36	2023-05-12 오후 7:45	한컴오피스 한셀 ...	18KB
normal_random_th_37	2023-05-12 오후 7:45	한컴오피스 한셀 ...	19KB
normal_random_th_38	2023-05-12 오후 7:45	한컴오피스 한셀 ...	19KB
normal_random_th_39	2023-05-12 오후 7:45	한컴오피스 한셀 ...	19KB
normal_random_th_40	2023-05-12 오후 7:45	한컴오피스 한셀 ...	19KB
normal_U_F_T_1	2023-05-12 오전 2:07	한컴오피스 한셀 ...	14KB
normal_U_F_T_2	2023-05-12 오전 2:07	한컴오피스 한셀 ...	15KB
normal_U_F_T_3	2023-05-12 오전 2:07	한컴오피스 한셀 ...	15KB

[ LSTM 모델 학습을 위해 라벨링된 데이터 ]

- 2) 모델 구축 : 각 레이어를 선형으로 연결하는 Sequential 모델을 생성하고, 입력 데이터의 형태는 x, y, z로 (None, 3) 형태로 지정하였습니다.

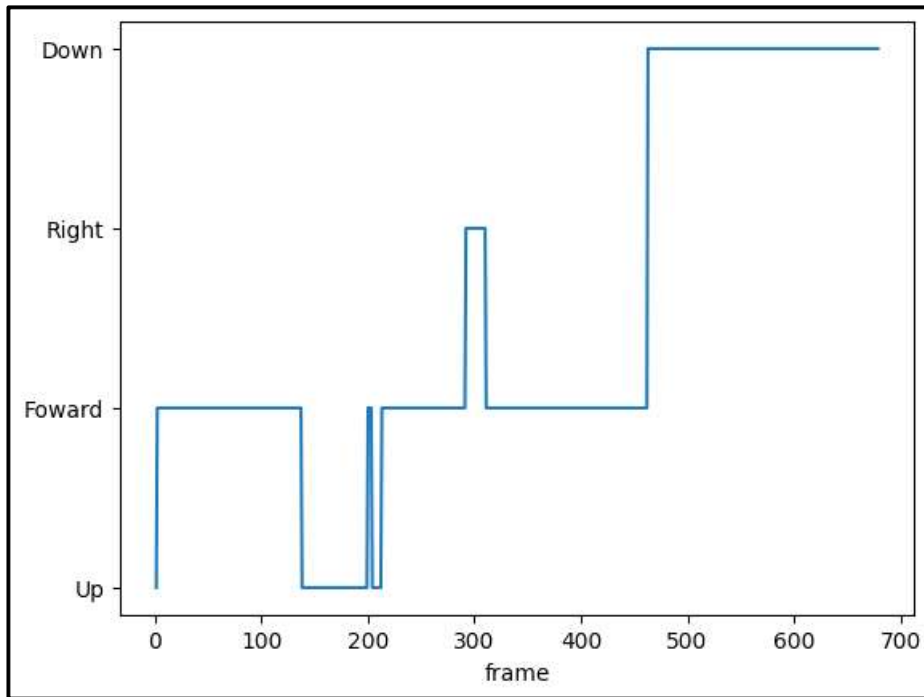
입력 데이터를 정규화하는 BatchNormalization 레이어를 추가하여 데이터의 평균을 0으로 만들고 분산을 1로 조정하여 학습을 안정화하였습니다. 마지막으로 Dropout 레이어를 사용해 코드의 과적합을 방지하였습니다.

출력 레이어는 이진 분류를 위해 1개의 유닛을 가지며, sigmoid 활성화 함수를 사용하여 0과 1 사이의 확률값을 출력합니다. 이를 통해 모델은 입력 데이터를 정규화하고, 드롭아웃을 적용하여 과적합을 방지한 후 이진 분류 작업을 수행합니다.





- 3) 모델 테스트 : 입력된 테스트 데이터를 윈도우 크기 150프레임 (5초) , 스텝 크기 30프레임 (1초)로 분할 한 후 학습된 LSTM 모델을 사용하여 각 윈도우 크기를 스텝 크기만큼 이동하면서 예측 결과를 확인하였습니다.



[ 테스트 영상 데이터 ]

약 500프레임 이전으로부터 150프레임 이상 Down 방향에 위치 하고있습니다. 따라서, 해당 부분은 abnormal로 예측된 것을 볼 수 있습니다.

0초에서5초 normal 1/1 [=====] - 0s 37ms/step	9초에서14초 normal 1/1 [=====] - 0s 37ms/step
1초에서6초 normal 1/1 [=====] - 0s 46ms/step	10초에서15초 normal 1/1 [=====] - 0s 42ms/step
2초에서7초 normal 1/1 [=====] - 0s 46ms/step	11초에서16초 abnormal 1/1 [=====] - 0s 38ms/step
3초에서8초 normal 1/1 [=====] - 0s 38ms/step	12초에서17초 abnormal 1/1 [=====] - 0s 40ms/step
4초에서9초 normal 1/1 [=====] - 0s 38ms/step	13초에서18초 abnormal 1/1 [=====] - 0s 37ms/step
5초에서10초 normal 1/1 [=====] - 0s 51ms/step	14초에서19초 abnormal 1/1 [=====] - 0s 43ms/step
6초에서11초 normal 1/1 [=====] - 0s 36ms/step	15초에서20초 abnormal 1/1 [=====] - 0s 36ms/step
7초에서12초 normal 1/1 [=====] - 0s 40ms/step	16초에서21초 abnormal 1/1 [=====] - 0s 38ms/step
8초에서13초 normal	17초에서22초 abnormal

## 4.5 기능 구성

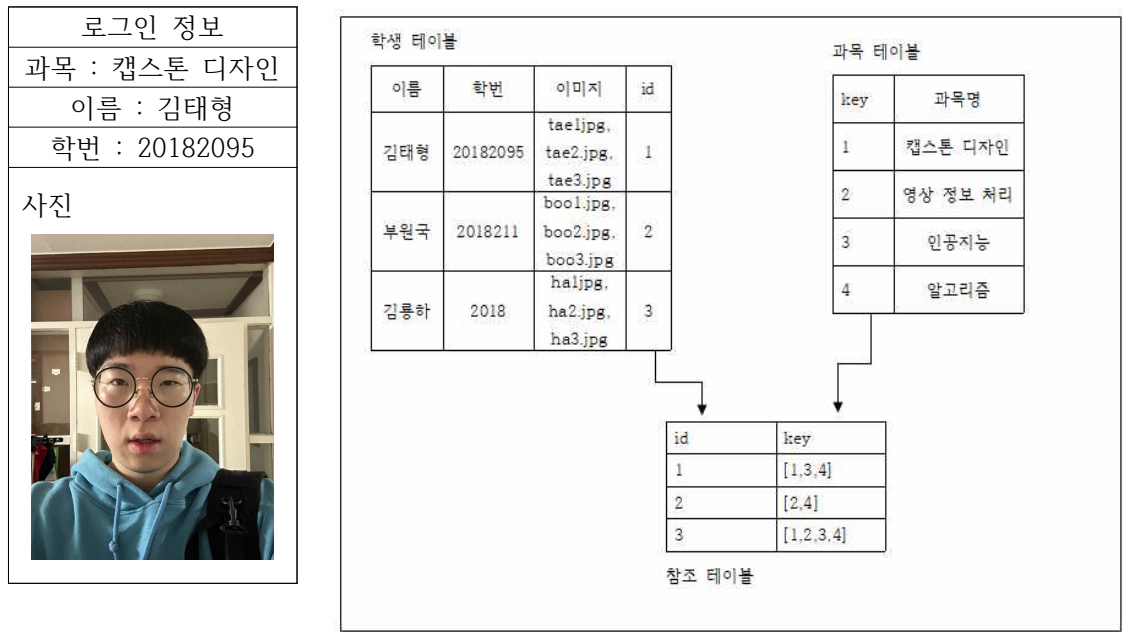
본 졸업작품의 주요 기능은 크게 세가지입니다.

1. 사용자 본인인증 : 로그인 인터페이스에 사용자의 얼굴과 로그인 정보를 입력받습니다. DeepFace를 활용하여 입력받은 정보와 데이터베이스 내의 사용자 정보의 일치 여부를 통해 본인인증 기능을 수행합니다
2. 동공방향 이상탐지 기능 : MLP로 학습된 얼굴 방향 예측모델과 Mediapipe내 iris 모듈의 동공 움직임 추적 알고리즘을 통해 사용자의 얼굴 방향이 정면을 향하고 동공 방향이 기준에서 벗어나는 경우를 탐지하는 기능을 수행합니다.
3. 고개방향 이상탐지 기능 : LSTM과 MLP로 학습된 모델을 활용하여 사용자의 고개 방향이 기준치에서 5초 이상 벗어난 경우를 탐지하고 방향 정보도 함께 예측하는 기능을 수행합니다.

본 졸업작품의 주요 기능과 함께 사용자 인터페이스, 이상 탐지 처리 시스템과 결합하여 사용자의 부정행위 방지와 관리자에게 실시간 부정행위 감지 알림 전송을 통해 비대면 시험의 부정행위를 방지기능을 구현하였습니다.

4.5.1. 사용자 본인인증

1) 입력된 로그인 정보와 웹캠에서 캡처한 이미지 데이터를 데이터베이스 정보와 비교하여 일치유무를 확인합니다.

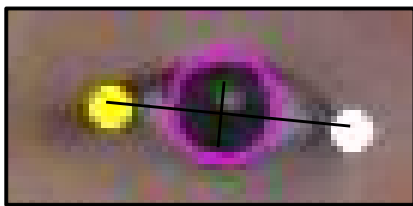


[ 데이터베이스 구조 ]

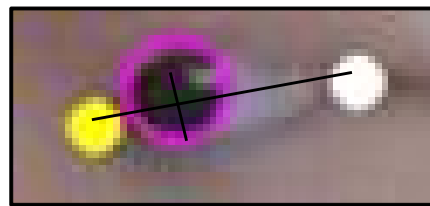
2) 입력된 이미지 데이터와 데이터베이스 내의 이미지 데이터를 4.4.2.에서 언급한 DeepFace의 얼굴 인식 과정을 통해 유사도 비교를 합니다.

#### 4.5.2 동공방향 이상탐지 기능

- 1) Mediapipe를 활용하여 동공의 좌표, 눈머리와 눈꼬리의 좌표를 검출합니다.
- 2) 검출한 좌표를 통해 유클리드 거리를 계산하여 동공의 중심에서 눈머리와 눈꼬리까지의 거리의 비율을 비교해 동공의 위치를 판별합니다. 동공의 중심과 눈의 양 끝 좌표의 비율이 10% 이내의 차이를 가진다면 정면을 바라보는 것으로 판별하였고, 30% 이상의 차이를 가진다면 좌, 우를 바라본다고 판별하였습니다.



[ 정면 ]



[ 왼쪽 ]

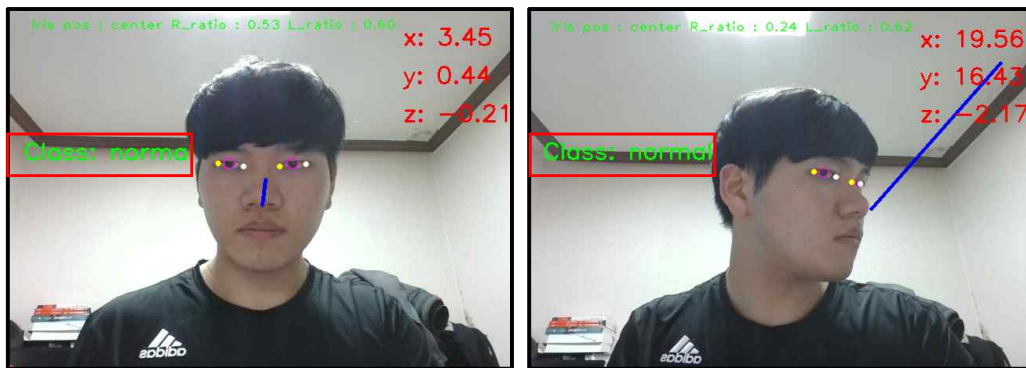
- 3) 4.4.4.에서 학습한 MLP 모델을 사용하여 얼굴 방향이 정면을 바라보고 있고, 검출한 동공 방향이 오른쪽이나 왼쪽일 경우 동공 이상행동으로 탐지하도록 설계했습니다.



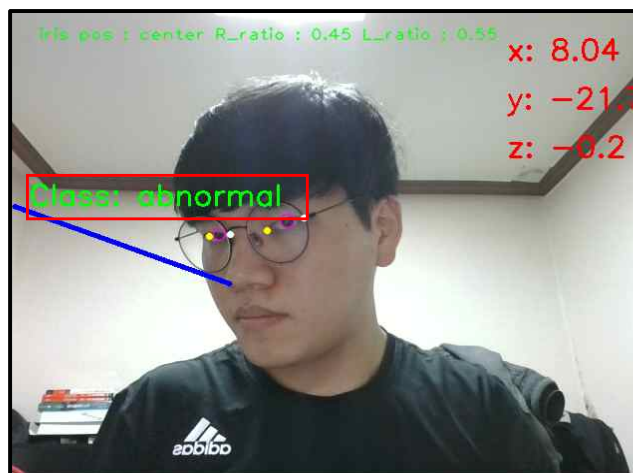
[ 이상행동 탐지 결과 ]

#### 4.5.3. 고개방향 이상탐지 기능

- 1) 4.4.5.에서 학습한 LSTM 모델을 불러와 사용자의 고개 방향이 정면에서 5초 이상 벗어난 경우를 이상행동으로 분류하여 탐지합니다.

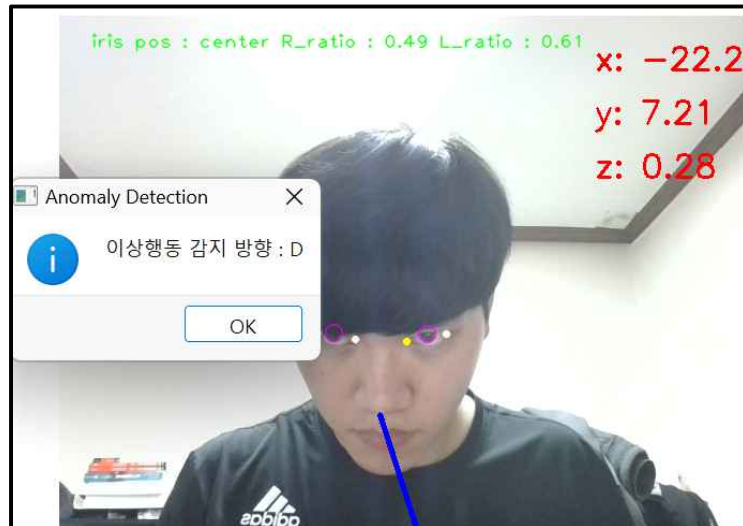


정면을 바라보거나 5초 이하로 다른 곳을 응시할 때는 normal로 분류합니다.



기준에서 벗어나 5초 이상 응시할 경우 abnormal로 분류합니다.

- 2) abnormal 분류와 함께 4.4.4.에서 학습한 MLP 모델을 불러와 고개 방향을 예측합니다.



#### 4.5.4. 카카오톡을 활용한 이상행동 탐지결과 알림 기능

이상 탐지 내역을 실시간으로 시험 감독관에게 보내기 위해 접근성이 좋은 Kakao Talk API를 활용하였습니다.

- 1) 카카오톡 메시지 보내기 기능을 사용하기 위해 kakao developers에 어플리케이션을 등록하고 REST API 키를 받았습니다.

앱 키	
네이티브 앱 키	e0cd94 [redacted]
REST API 키	2f8302 [redacted]
JavaScript 키	4de186 [redacted]
Admin 키	620626 [redacted]

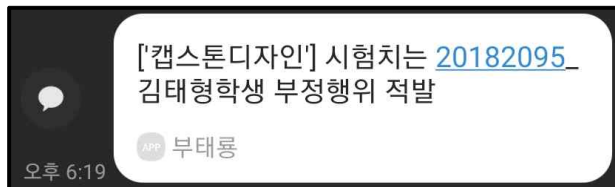
[ 어플리케이션 키 ]

2) REST API를 이용하여 메시지를 보내기 위해 접근 토큰을 발급받습니다.

```
{'access_token': '6w_6MhjWf03DCBVn4X0kuX8HzGYBERn0jF9wyAHdCj11WwAAAYh2SHem', 'token_type': 'bearer', 'refresh_token': 'plWMrU6wkbZroTsEDiy8r4nx470FFvFDBJ_ViRLW3Cj11WwAAAYh2SHel', 'expires_in': 21599, 'scope': 'talk_message', 'refresh_token_expires_in': 5183999}
```

접근 토큰의 유효기간이 6시간 내외이기 때문에 자동 갱신기능을 위해 refresh token을 사용합니다.

3) 이상 탐지가 된 경우 ‘학번’, ‘이름’, ‘시간’, ‘이상 탐지 내용’을 카카오톡 메시지로 보냅니다. HTTP 요청을 위한 헤더에 인증 토큰을 포함하고, 메시지 내용은 JSON 객체를 생성하여 POST 요청을 URL로 보냅니다.



#### 4.5.5. 키보드 및 마우스 이상행동 탐지기능

키보드, 마우스 이벤트 처리를 통해 시험 도중에 부정행위로 판단될 수 있는 키보드 커맨드 와 마우스 우클릭을 감지하여 이상 탐지 처리하였습니다.

##### 1) 키보드

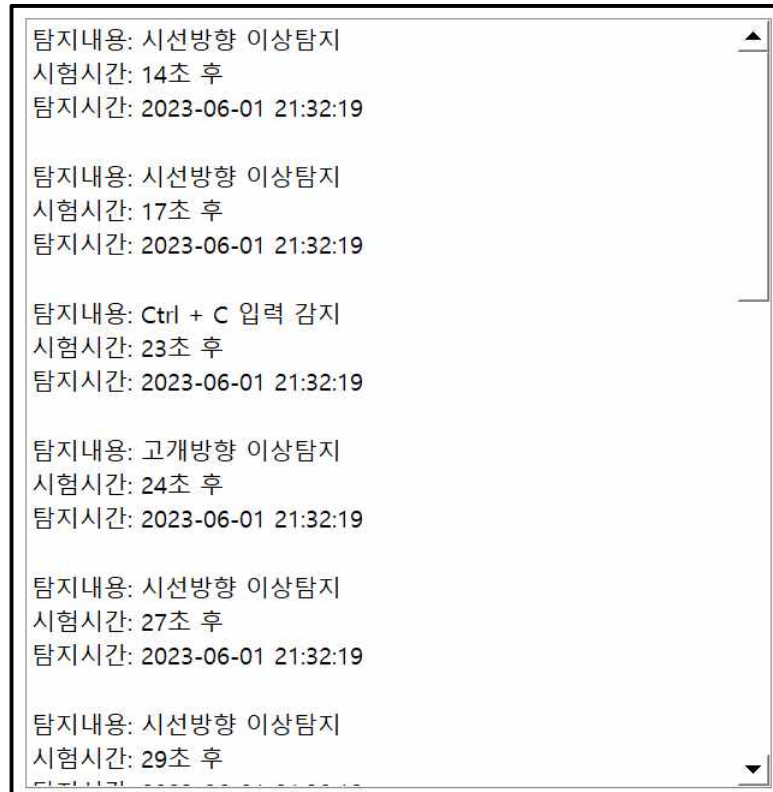
- Ctrl + C , Ctrl + V : 시험 응시 중 복사 붙여넣기를 통해 부정행위가 일어날 수 있는 요소를 감지합니다.
- Alt + Tab : 시험 응시 중 윈도우 창을 바꿔 부정행위가 일어날 수 있는 요소를 감지합니다.

##### 2) 마우스

- 우클릭 : 시험 응시 중 우클릭으로 복사 붙여넣기를 하여 부정행위가 일어날 수 있는 요소를 감지합니다.

#### 4.5.6. 이상행동 탐지결과 로그 기록 기능

이상 탐지 내역을 사용자에게 보여주기 위해 탐지한 내역을 로그 데이터로 저장하였습니다. 마지막 페이지에서 이상 탐지 내용이 기록된 로그 데이터를 보여주어 사용자에게 알려주도록 하였습니다.



[ 이상탐지 결과 로그 데이터 ]

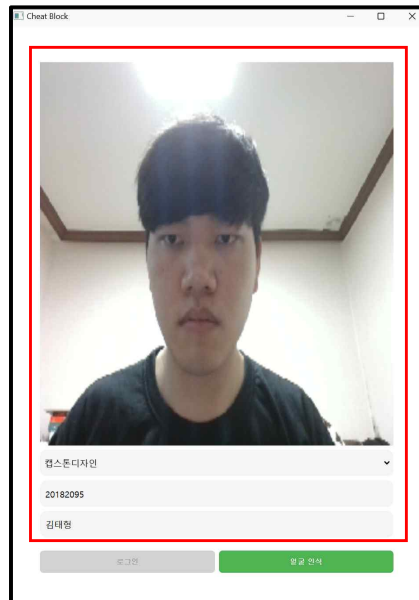
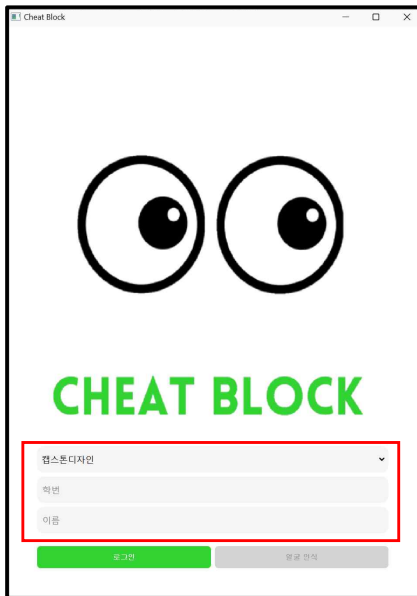
#### 4.6. 사용법

1) exe 파일을 다운받아 시스템을 실행합니다.

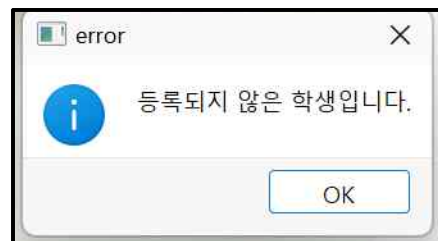
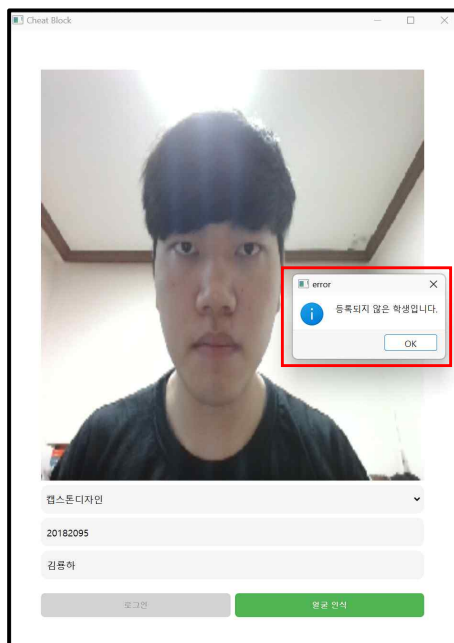
 qt_first	2023-05-30 오전 1:19	파일 폴더	
 cheat_block	2023-05-30 오전 10:07	응용 프로그램	806,683KB



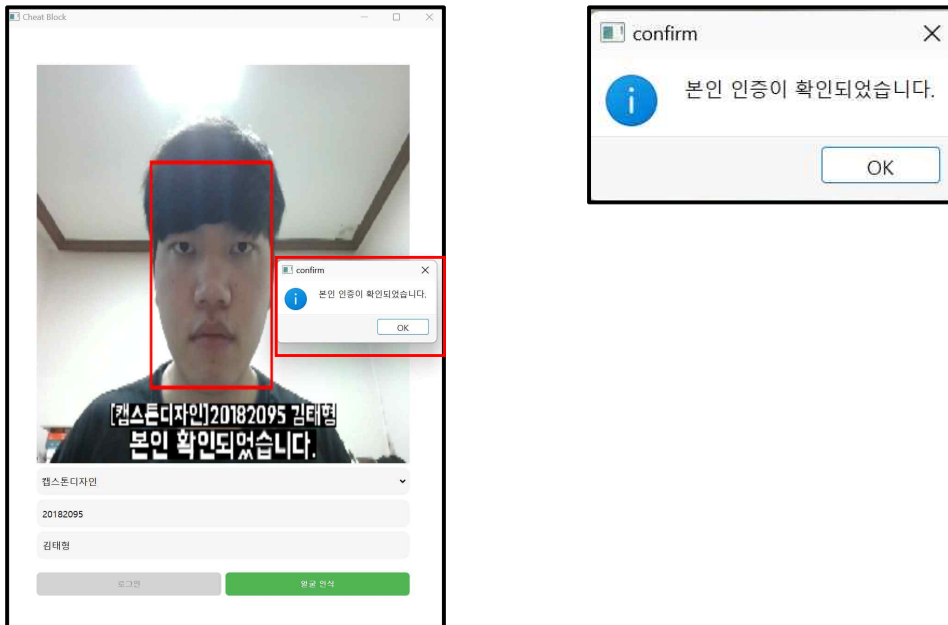
- 2) 사용자 인터페이스에 과목, 이름, 학번 정보를 입력하고 로그인 버튼을 클릭합니다. 로그인 버튼 클릭 시 로고 이미지가 웹캠으로 변환됩니다



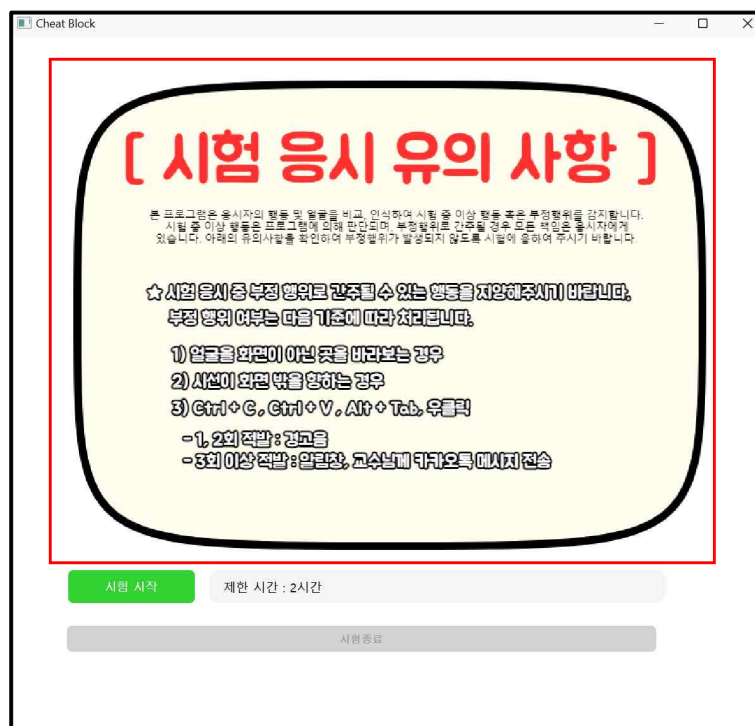
- 3) 웹캠 화면에 사용자의 얼굴이 표시되면 얼굴 인식 버튼을 눌러 본인인증 과정을 거칩니다.
- 4) 입력된 로그인 데이터와 데이터베이스 내의 데이터와의 비교를 통해 일치하지 않으면 “등록되지 않은 학생입니다” 메시지 박스를 출력하고 확인 버튼을 누르면 첫 화면으로 돌아갑니다.



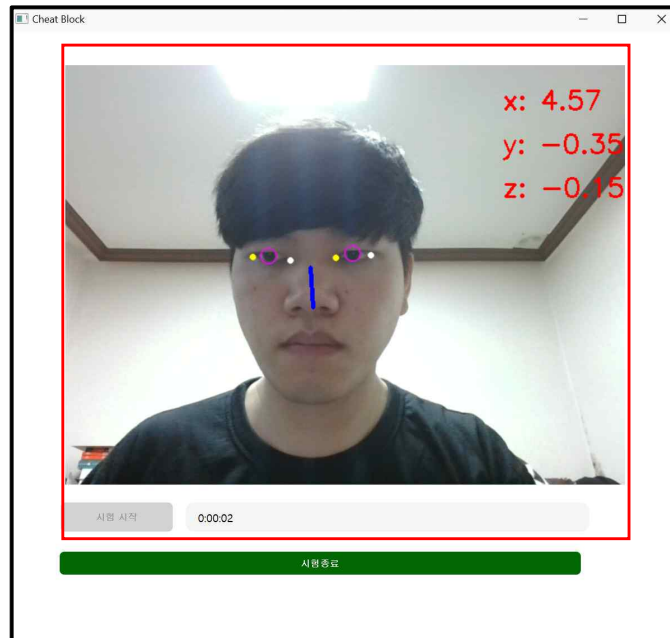
일치한다면 “본인인증이 확인되었습니다.” 메시지 박스를 띄우고 확인 버튼을 누르면 두 번째 화면으로 이동합니다.



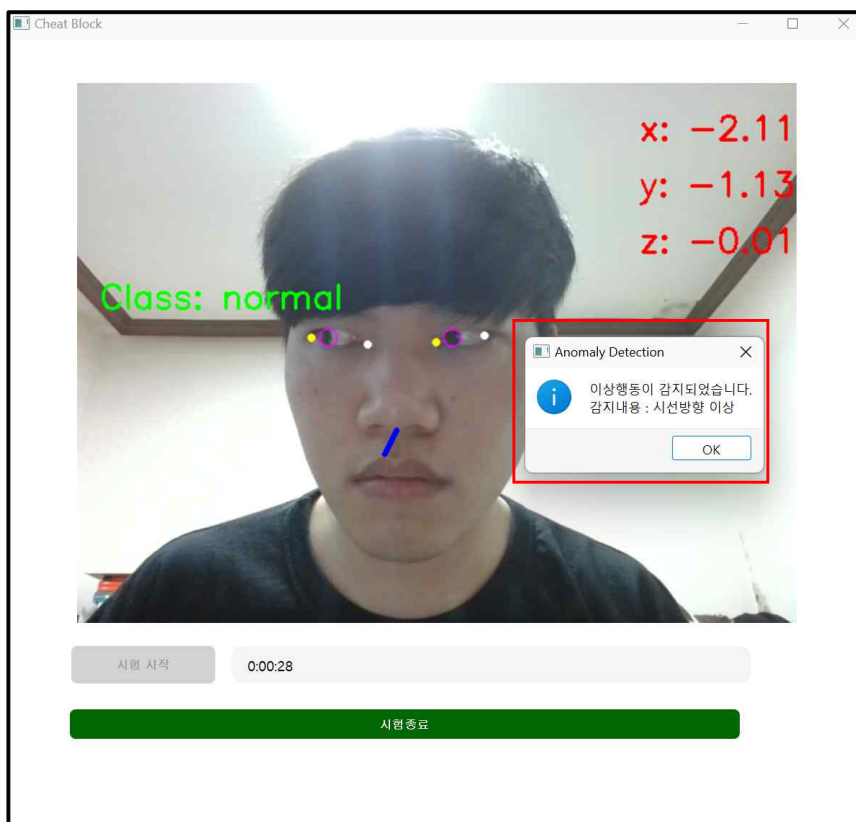
5) 시험 유의 사항을 통해 사용자에게 부정행위 기준에 대해 알려줍니다.



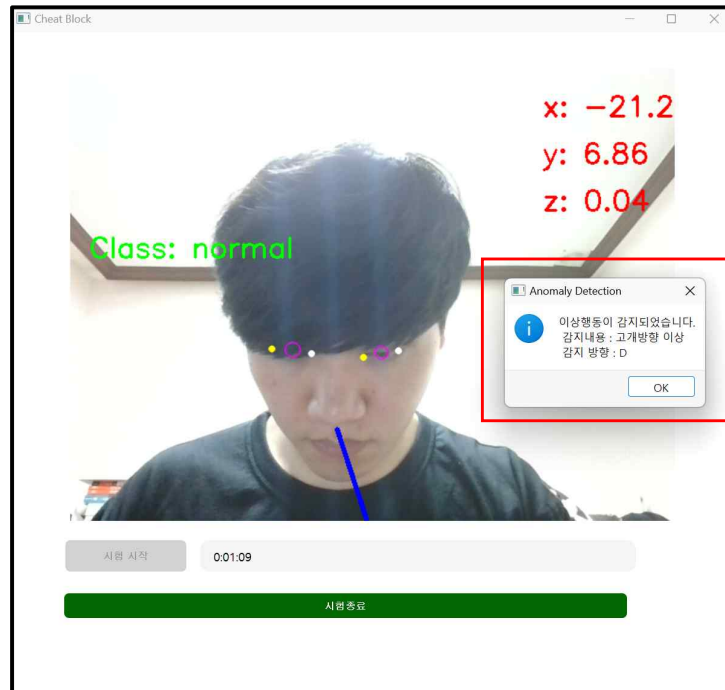
- 6) 시험 시작 버튼을 클릭하면 웹캠이 활성화되고, 2시간으로 설정된 타이머가 흘러 갑니다. 타이머가 2시간이 다 지나면 시험은 자동으로 종료됩니다.



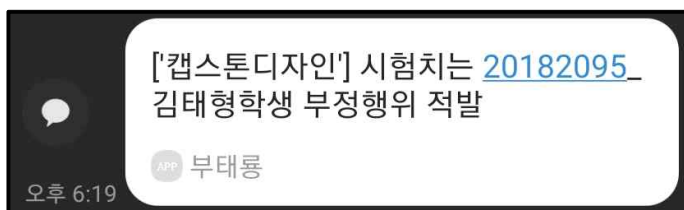
- 7) 사용자의 동공 방향이 다른 곳을 향한다면 이상행동을 탐지하고 경고 메시지를 띄웁니다.



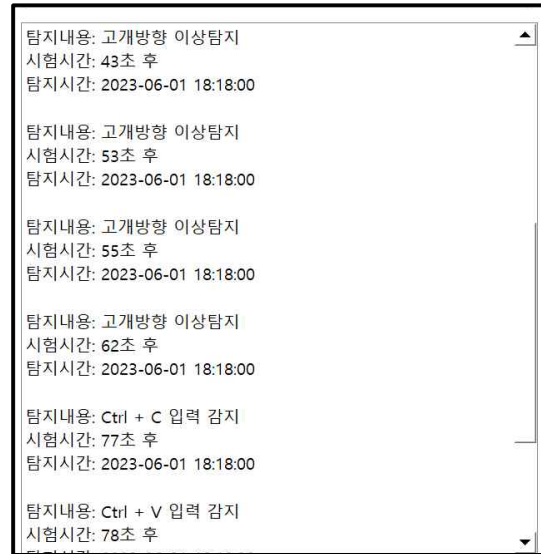
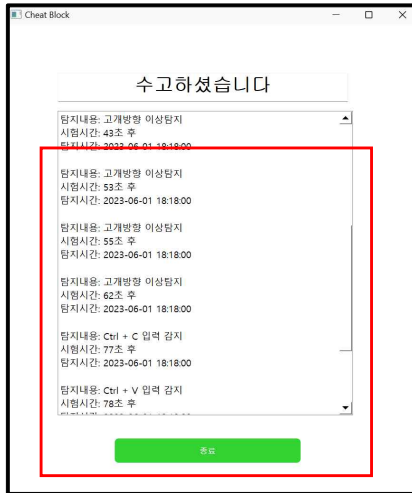
- 8) 사용자의 고개 방향이 다른 곳을 향하면 이상행동을 탐지하고 1회와 2회에 각각 음성 경고를 줍니다. 3회 이상부터 메시지 박스를 통해 경고문구를 띄워줍니다.



- 9) 3회 이상 이상행동을 감지하면 관리자에게 실시간으로 과목, 학번, 이름 정보를 카카오톡 메시지로 전송합니다.



- 10) 시험 종료 버튼을 누르면 시험이 종료되고, 세 번째 페이지로 넘어갑니다. 시험 중 감지하였던 이상행동을 로그 정보로 출력합니다.



## 4.7. 기능구현

### 4.7.1. Head 포즈 추정을 위한 얼굴 특징점 x, y, z 좌표 추출

```
if results.multi_face_landmarks:
    for face_landmarks in results.multi_face_landmarks:
        for idx, lm in enumerate(face_landmarks.landmark):
            if idx == 33 or idx == 263 or idx == 1 or idx == 61 or idx == 291 or idx == 199:
                if idx == 1:
                    nose_2d = (lm.x * img_w, lm.y * img_h)
                    nose_3d = (lm.x * img_w, lm.y * img_h, lm.z * 3000)
                    x, y = int(lm.x * img_w), int(lm.y * img_h)
                    # Get the 2D Coordinates
                    face_2d.append([x, y])
                    # Get the 3D Coordinates
                    face_3d.append([x, y, lm.z])

        # Convert it to the NumPy array
        face_2d = np.array(face_2d, dtype=np.float64)
        # Convert it to the NumPy array
        face_3d = np.array(face_3d, dtype=np.float64)

        # The camera matrix
        focal_length = 1 * img_w

        cam_matrix = np.array([ [focal_length, 0, img_h / 2],
                                [0, focal_length, img_w / 2],
                                [0, 0, 1]])

        # The distortion parameters
        dist_matrix = np.zeros((4, 1), dtype=np.float64)
        # Solve PnP
        success, rot_vec, trans_vec = cv2.solvePnP(face_3d, face_2d, cam_matrix, dist_matrix)
        # Get rotational matrix
        rmat, jac = cv2.Rodrigues(rot_vec)
        # Get angles
        angles, mtxR, mtxQ, Qx, Qy, Qz = cv2.RQDecomp3x3(rmat)

        # Get the y rotation degree
        x = angles[0] * 360
        y = angles[1] * 360
        z = angles[2] * 360

        # Display the nose direction
        nose_3d_projection, jacobian = cv2.projectPoints(nose_3d, rot_vec, trans_vec, cam_matrix, dist_matrix)
        p1 = (int(nose_2d[0]), int(nose_2d[1]))
        p2 = (int(nose_2d[0] + y * 10), int(nose_2d[1] - x * 10))
        cv2.line(image, p1, p2, (255, 0, 0), 3)
```

Mediapipe의 FaceMesh 활용하여 얼굴 특징점을 감지하고, 감지된 특징점들의 3D 좌표를 추출합니다. 추출된 좌표를 활용해 머리의 포즈를 추정하고, 이를 바탕으로 코의 방향을 계산하여 x,y,z 좌표로 나타냅니다.

#### 4.7.2. 시선 방향 감지를 위한 눈동자 위치 판별

```
def euclidean_distance(self, point1, point2):
    x1, y1 = point1.ravel()
    x2, y2 = point2.ravel()
    distance = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
    return distance

def iris_position(self, iris_center_right, iris_center_left, R_right_point, R_left_point, L_right_point,
                  L_left_point):
    R_center_to_right_dist = self.euclidean_distance(iris_center_right, R_right_point)
    R_total_distance = self.euclidean_distance(R_right_point, R_left_point)
    R_ratio = R_center_to_right_dist / R_total_distance
    L_center_to_right_dist = self.euclidean_distance(iris_center_left, L_right_point)
    L_total_distance = self.euclidean_distance(L_right_point, L_left_point)
    L_ratio = L_center_to_right_dist / L_total_distance
    iris_position = ""
    if R_ratio >= 0.65:
        iris_position = "left"
    elif L_ratio <= 0.35:
        iris_position = "right"
    elif 0.45 <= R_ratio <= 0.55:
        iris_position = "center"
    return iris_position, R_ratio, L_ratio
```

눈동자 중심으로부터 각 눈의 오른쪽과 왼쪽 꼭짓점까지의 유클리드 거리를 계산하고, 이를 통해 눈동자의 상대적 위치를 결정합니다. 이상 탐지 기준 비율을 이용하여 눈동자가 왼쪽, 오른쪽 또는 중앙에 있는지를 판별하고 이를 반환합니다.

#### 4.7.3. LSTM모델과 MLP 모델을 이용한 이상탐지

```
lstm_predictions = []

splitted_list = self.x_y_z_list[0:]
self.window_data = np.array(splitted_list)
self.window_data = self.window_data.reshape((1, self.window_data.shape[0], 3))
lstm_predict_result = model_lstm.predict(self.window_data)

if lstm_predict_result > 0.5:
    lstm_predictions.append(1)
    self.event_list.append(1)

    self.log_list.append(['고개방향 이상탐지 ', self.seconds, self.current_time.strftime('%Y-%m-%d %H:%M:%S')])
else:
    lstm_predictions.append(0)
    mlp_new_predictions = model_mlp_2.predict(self.window_data[0])
    new_mlp_labels = [new_mlp_label_mapping[np.argmax(pred)] for pred in mlp_new_predictions]
    most_newmlp_value = new_mlp_labels[-1]

    self.event_list.append(0)
    if most_newmlp_value == 'F' and (iris_pos == "left" or iris_pos == "right"):
        self.Anomaly_detection_message_2()
        self.log_list.append(['시선방향 이상탐지 ', self.seconds, self.current_time.strftime('%Y-%m-%d %H:%M:%S')])

self.count_abnormal(self.event_list)

counter_lstm = Counter(lstm_predictions)
most_lstm_common_value = counter_lstm.most_common(1)[0][0]

lstm_label.append(lstm_labels_map[most_lstm_common_value])
```



입력된 시계열 데이터를 LSTM 모델에 적용하여 이상 탐지를 수행합니다. LSTM 모델의 출력을 기반으로 고개 방향의 이상을 탐지하고, MLP 모델을 활용하여 시선 방향을 탐지합니다.

#### 4.7.4. DeepFace를 이용한 얼굴 유사도 비교

```
frame_np = np.array(self.frame)
df = DeepFace.find(img_path = frame_np, db_path = ".", detector_backend = 'mtcnn')
df = df[0][df[0]["VGG-Face_cosine"] < 0.1]

identity_list = df['identity'].tolist()

if len(identity_list) == 0:
    most_common_identity = "unknown"
else:
    identity_list = [i.split('/')[-1].split('.')[0] for i in identity_list]

    # "_"를 기준으로 분할하여 첫 번째 요소를 가져옵니다.
    identity_prefixes = [i.split('_')[0] for i in identity_list]

    # 가장 많이 등장하는 요소를 찾습니다.
    counter = Counter(identity_prefixes)
    most_common_identity = counter.most_common(1)[0][0]

    if most_common_identity :
        print(most_common_identity)
```

DeepFace를 이용하여 이미지 데이터에서 얼굴을 검색하고 해당 얼굴들의 신원을 식별합니다. 이때 'VGG-Face\_cosine' 값이 0.1보다 작은 이미지 데이터들만을 대상으로 하며, 이들 이미지에서 가장 많이 등장한 신원을 찾아냅니다. 만약 식별된 얼굴이 없다면 "unknown"을 출력합니다.



#### 4.7.5. 로그인 정보와 데이터베이스 정보 비교

```
def check_person_in_database(self, most_common_pred, database_path):
    # 데이터베이스 연결
    conn = sqlite3.connect(database_path)
    c = conn.cursor()

    c.execute("SELECT * FROM student WHERE label=?", (most_common_pred,))
    result = c.fetchone()

    if result is not None:
        # 학생 정보 추출
        id = result[0]
        self.student_id = result[2]
        self.name = result[1]

        # 수강 과목 정보 조회
        c.execute("SELECT subject FROM subject_student INNER JOIN subject ON subject_student.subject_id = subject.key WHERE id=?", (id,))
        self.subjects = c.fetchall()
        self.subjects = [subject[0] for subject in self.subjects]
        self.subjects_str = ", ".join(self.subjects) # Convert list to a string with comma-separated values

    # 연결 종료
    conn.close()
    ui_student_id = self.lineEdit.text()
    ui_name = self.lineEdit_2.text()
    ui_subject_name = self.comboBox.currentText()
    if self.student_id == ui_student_id and self.name == ui_name and ui_subject_name in self.subjects:
        # UI에 입력된 값과 조회된 학생 정보가 일치하는 경우
        # 원하는 작업 수행
        print("Person is registered in the database.")

        # 얼굴 인식 및 추출
        out = DeepFace.extract_faces(self.frame, detector_backend = 'mtcnn')
        facial_area = out[0]["facial_area"]
        x = facial_area['x']
        y = facial_area['y']
        w = facial_area['w']
        h = facial_area['h']
```

SQLite 데이터베이스를 연결하고, 사용자 인터페이스에 입력된 값과 데이터베이스 내의 정보와 비교합니다. 일치하는 경우, DeepFace를 이용하여 얼굴을 인식하고 추출합니다. 이 과정은 학생의 수강 과목 정보를 조회하고, 학생이 데이터베이스에 등록되어 있음을 확인하는 데 사용됩니다.

## 4.8 함수별 기능

- 1) create\_kakao\_api() 함수 :  
카카오톡 메시지 API를 사용하기 위해 토큰 생성을 합니다.  
API에 필요한 정보를 전송하고, 결과를 JSON형식으로 저장합니다.
- 2) check\_person\_in\_database( ) 함수 :  
SQLite 데이터베이스에 연결한 후, 학생 정보와 입력된 정보를 비교합니다.  
일치하는 경우, 얼굴 인식을 수행하고 인증 메시지를 출력합니다.  
일치하지 않는 경우, 오류 메시지를 출력합니다.
- 3) button2Function() 함수 :  
얼굴인증 버튼을 클릭하였을 때 호출되는 함수입니다.  
웹캠 프레임을 이미지로 변환 후, DeepFace를 사용하여 얼굴 인식을 수행합니다.  
인식된 얼굴을 데이터베이스에서 확인하고, 인증 여부를 결정합니다.
- 4) update\_frame( ) 함수 :  
웹캠 프레임을 업데이트하고 사용자의 동공 방향과 고개 방향을 감지하여 이상행동을 탐지합니다.
- 5) iris\_position( ) 함수 :  
오른쪽 눈동자 위치, 왼쪽 눈동자 위치, 오른쪽 눈 끝 점, 오른쪽 눈 시작 점, 왼쪽 눈 끝 점, 왼쪽 눈 시작 점을 인자로 받아 눈동자 위치와 비율을 계산합니다.
- 6) count\_abnormal( ) 함수 :  
이상행동 횟수를 카운트하여 이상행동 횟수에 기반하여 1, 2회 음성 경고 3회 이상 메시지 박스 경고를 출력하고, 3회 이상 이상행동이 감지되면 카카오톡 API를 활용하여 관리자에게 메시지를 전송합니다.
- 7) kakaoapi() 함수 :  
Kakao API를 이용해 메시지를 보내는 역할을 합니다. 만약 메시지 전송이 성공적으로 이루어지면 성공 메시지를 출력하고, 그렇지 않으면 오류 메시지를 출력합니다.
- 8) keyPressEvent, mousePressEvent() 함수 :  
키보드와 마우스 입력을 감지하여 특정 조합이나 우클릭이 감지되면 로그를 기록하고 화면에 출력하는 기능을 합니다.

## 4.9. 기존 연구 결과와 비교

본 졸업작품의 주요 기능인 본인인증, 이상탐지를 관련 연구와 비교하였습니다.

제목	얼굴 인식	이상 탐지	본 졸업작품과 비교
[1]	Haar cascade 알고리즘	특수키 인식 방식, 차단 프로그램 관리	Haar cascade 알고리즘은 얼굴의 명암차이를 통해 얼굴인식을 하는 반면, 본 작품에서는 DeepFace를 활용하여 얼굴을 인식하고 분석하는 더 정교하고 정확한 방법을 채택하였습니다.
[2]	FaceNet, SVM	CNN	CNN은 이미지 처리 기반 딥러닝 모델로 본 졸업작품은 LSTM을 활용하여 시계열 데이터를 기반으로 이상탐지 기능을 수행하였습니다.
[3]	S3FD, Dlib	마우스 움직임 패턴 추출	Dlib의 HOG 기반 얼굴 탐지기는 본 졸업작품과 비교하여, 상대적으로 경량화되어 있으며, 모델의 크기가 작고 실행에 필요한 계산 비용이 낮습니다.

[1] 응시자 행동로그와 영상데이터 분석을 통한 온라인 시험 부정행위 방지 시스템 구현

출원인 : 대구가톨릭대학교 컴퓨터소프트웨어학부

[2] 인공지능 무인 감독 시스템

출원인 : 인천대학교 임베디드시스템공학과

[3] 온라인 시험 환경에서의 응시자 행동로그와 영상데이터 분석을 통한 부정행위자 감별

출원인 : 서강대학교 아트앤테크놀로지, 가톨릭대학교 정보통신전자공학부, 네이버 주식회사

## 5. 제작일정

(제작기간 : 2022. 09. 01 ~ 2023. 06. 01)										담당학생
제작내용	월 별									
	9월	10월	11월	12월	1월	2월	3월	4월	5월	
주제 선정 및 자료 조사										팀원 전체
데이터 수집										팀원 전체
시스템 설계 구상										팀원 전체
개발 환경 구축										부원국, 김룡하
Mediapipe를 이용한 데이터 전처리										김태형, 김룡하
LSTM 모델 구현 및 학습										김태형, 부원국
MLP 모델 구현 및 학습										김룡하
DeepFace 알고리즘 구현										부원국
모델 테스트 및 오류 수정										팀원 전체
GUI 설계 및 구현										김태형
실행 파일 설계 및 배포										부원국
시연 영상 촬영, 보고서 작성										팀원 전체

## 6. 향후 계획과 기대효과

### [향후 계획]

1. 음성 인식 기술을 추가하여 시험 도중 다른 사람의 목소리나 응시자의 말하기를 감지하도록 기능을 추가할 계획입니다.
2. UCLASS 서버와 연동하여 비대면 시험의 공정한 시험 환경을 제공할 계획입니다.
3. “ICT 비즈니스 모델 아이디어 경진대회” 출품하여, 본 작품을 통한 수상을 계획하고 있습니다.

### [기대효과]

본 졸업작품인 “딥러닝 기반 비대면 시험 부정행위 감지 시스템”은 다음과 같은 기대효과를 가지고 있습니다.

1. 공정성 향상 : 비대면 시험의 공정성 향상을 통해 응시자들 간의 불평등을 해소할 수 있습니다.
2. 자원의 절약 : 본 졸업작품을 통해 딥러닝 기반의 자동화된 시험 감독 시스템을 도입하여 학교나 기관에서 물리적, 인적 자원을 절약할 수 있습니다.
3. 신뢰성 향상 : 본 졸업작품을 통해 비대면 시험의 결과에 대한 신뢰성을 향상시켜 비대면 시험의 보급화를 통해, 시험자의 지리적, 시간적 제약을 해소하는데 기여할 수 있습니다.