종합설계 프로젝트 수행 보고서

프로젝트명	감정 인식 영상 채팅 어플리케이션
팀번호	S5-4
문서제목	수행계획서( ) 2차발표 중간보고서( ) 3차발표 중간보고서( ) 최종결과보고서(O)

2020.12.03

팀원 : 탁 원 준 (팀장)

금 기 륜 (팀원)

지도교수 : 방 영 철 교수 (인)

문서 수정 내역

작성일	대표작성자	버 전 (Revision)	수정내용	
2020.01.22	탁원준(팀장)	1.0	수행계획서	최초 작성
2020.03.02	탁원준(팀장)	2.0	중간발표1	상세 설계 추가
2020.05.02.	탁원준(팀장)	3.0	중간발표2	Prototype 구현
2020.06.27	탁원준(팀장)	4.0	학기말 발표	시험 결과
2020.12.03.	탁원준(팀장)	5.0	최종	결론

문서 구성

진행단계	프로젝트 계획서 발표	중간발표1 (2월)	중간발표2 (4월)	학기말발표 (6월)	최종발표 (10월)
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~3) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~4) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~5) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~7) 참고자료	Ⅰ Ⅱ Ⅲ

이 문서는 한국산업기술대학교 컴퓨터공학부의
"종합설계" 교과목에서 프로젝트
"감정 인식 영상 채팅 어플리케이션"을 수행하는
S5-4팀(탁원준, 금기륜)이 작성한 것으로
사용하기 위해서는 팀원들의 허락이 필요합니다.

목 차

Ⅰ. 서론

1. 작품선정 배경 및 필요성 ····· 04
2. 기존 연구/기술동향 분석 ····· 04
3. 개발 목표 ····· 04
4. 팀 역할 분담 ····· 04
5. 개발 일정 ····· 05
6. 개발 환경 ····· 05

Ⅱ. 본론

1. 개발 내용 ····· 06
2. 문제 및 해결방안 ····· 06
3. 시험시나리오 ····· 07
4. 상세 설계 ····· 08
5. Prototype ····· 14
6. 구현 알고리즘 ····· 18
7. 시험 / 테스트 결과 ····· 21

Ⅲ. 결론 ····· 23

참고자료 ····· 23

Ⅰ. 서론

1. 작품선정 배경 및 필요성

일상생활에서 기계로부터 제공받는 서비스는 매우 다양해졌으며, IOT 와 기술의 개발에 따라 더욱 다양해질 전망이다. 반면 사용자의 행동패턴 혹은 관심사에 대해서 맞춤형 서비스가 제공되지만 사용자에 감정과는 무관하다는 한계점이 존재한다. 더 유연한 서비스를 제공하기 위해서는 기계가 사용자의 감정을 파악해서 제공할 필요가 있다.

본 프로젝트는 감정을 인식하는 인공지능 구축을 통하여 실시간으로 사용자의 감정을 파악하는 것이 주 목적이며, 나아가 감정에 따른 다양한 서비스를 제공 할 수 있는 환경을 구축하기 위한 프로젝트이다.

2. 기존 연구/기술동향 분석

얼굴 인식 - Naver(CFR API), Google(Vision API)

감정 인식 - Google(Vision API), Microsoft(Face API)

세계적인 기업들은 이미 사람의 얼굴에서 감정을 다루는 기술들을 많이 개발해왔으며 오픈소스로 공개한 상태이다. 이를 이용하여 많은 개발자들이 다양한 얼굴 및 감정 인식 프로그램을 개발하였다. 그러나, 대부분의 경우가 하나의 이미지를 대상으로 하였을 때는 얼굴 인식률과 감정 인식률이 만족스러운 결과가 나왔으나, 영상에서 실시간으로 감정을 처리하는 방식은 속도 차원에서 제대로 된 처리가 안되거나 정확도가 부족하였다.

3. 개발 목표

1. 영상 데이터에서 얼굴 검출 후 표정의 특징 추출, 이를 통해 감정을 인식하는 인공지능 제작
2. 수신한 영상에 대해서 인공지능이 판단한 결과 값을 응답해주는 서버 구축
3. 다중 클라이언트의 요청 환경을 구현하기 위한 채팅프로그램 구현

4. 팀 역할 분담

	탁원준	금기륜
자료수집	서버 제작 클라이언트/서버 통신 기법	학습 데이터 Python/C 모듈 통합 인공지능 관련 논문
설계	서버/클라이언트 구조 응용계층 프로토콜 설계 UI 디자인	얼굴인식 모듈 데이터 전처리 모듈 통합 모듈 구조 설계
	인공지능 모듈 설계	
구현	서버/클라이언트 응용계층 프로토콜 UI 디자인	데이터 전처리 모듈 모듈 간 통합 및 처리 속도 최적화
	인공지능 모듈 개발 예측 정확도 개선	

5. 개발 일정

추진사항	12월	1월	2월	3월	4월	5월	6월	7~9월
기획 및 설계								
개발								
테스팅								
추가 보안								
데모								
중간 보고서								
발표 및 시연								

6. 개발 환경

Server / Client 구동 환경 - Windows

Library - MFC, opencv, tensorflow

개발툴 - Visual Studio 2017, Pycharm 2018.2, Jupyter notebook

H/W - Notebook Web Camera

Database - csv File

Ⅱ. 본론

1. 개발 내용

< 서버 >

- Windows10 환경에서 작동시킬 수 있는 서버 프로그램 제작
- 개방한 포트(9000~)에서 외부에서 접근할 수 있도록 Windows inbound 설정
- Thread를 통해 연결된 다수의 클라이언트에 대한 병렬적 처리
- Opencv2 라이브러리를 통해 수신된 영상을 전송하기 위한 데이터 처리
- 제작된 AI 모듈을 통한 예측값 도출

< 클라이언트 >

- PC, Windows10 환경에서 사용가능 하도록 Windows Programming, MFC 로 어플리케이션 제작
- Opencv2 라이브러리를 활용하여 기본 캠(노트북의 경우)으로 영상 촬영

< AI >

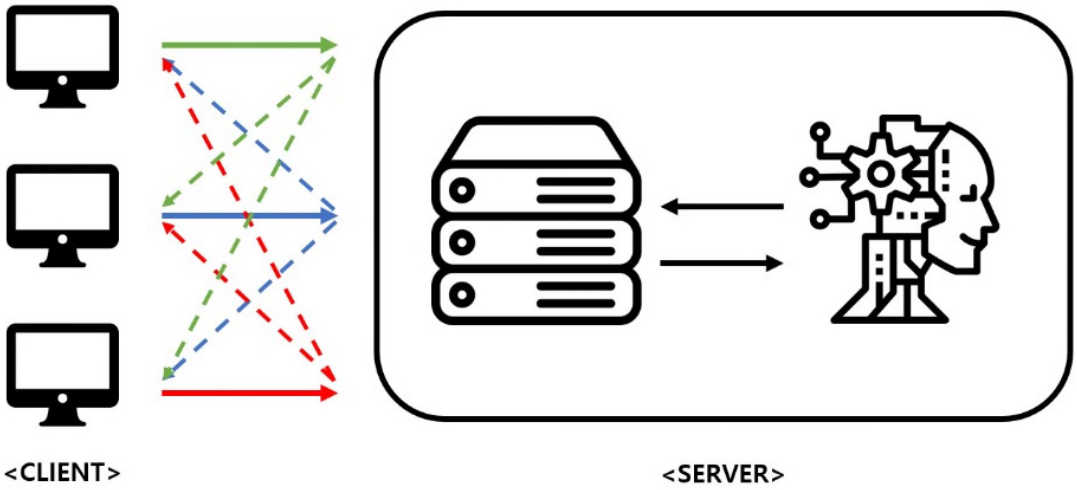
- 학습 데이터 수집을 통한 DB 구축
- Python Opencv 라이브러리를 활용한 얼굴 인식 모듈 제작
- 학습 데이터 전처리 모듈 제작
- Python Tensorflow 라이브러리를 활용한 딥러닝 모델 제작 및 학습
- AI 모델 정확도 개선

2. 문제 및 해결방안

1. 인공지능의 정확도가 높지 않아 제대로된 인식을 하지 못하는 경우
   - 앙상블 기법을 통한 모델 제작
   - 은닉 유닛 및 층 수 수정
   - 학습률 등 하이퍼 파라메타 수정
   - 데이터 추가 수집

2. 각 클라이언트와 서버간에 전송되는 영상의 시간 차이와 해당 영상에 관한 인공지능의 연산결과가 실시간으로 이루어지지 않는 경우
   - 은닉 층 및 은닉 유닛 개수 수정을 이용, 정확도가 낮아지는 단점이 있으나 처리 속도 문제 개선
   - 사용되는 언어 자체의 속도 문제일 경우 Python 에서 C++ 로 변경

3. 시험 시나리오



시나리오	내용
서버 동작	서버 : AI 학습 / 클라이언트가 연결 요청할 수 있도록 포트 개방
클라이언트 연결	클라이언트 : 개방된 포트로 클라이언트 TCP 방식으로 연결 요청
	서버 : 해당 통신방식으로 제어 신호 전송 클라이언트 : 연결된 TCP 통신으로 채팅 가능
영상 전송 설정	서버 : 정상적으로 연결된 클라이언트로 UDP통신이 진행될 수 있도록 포트 배포
영상 전송	클라이언트 : 배포된 포트로 자신의 영상 송신
클라이언트 추가 연결	서버 : 기존의 연결된 클라이언트에게 새로 연결된 클라이언트의 식별번호와 그와 관련된 포트번호 배포 클라이언트 : 새로 배포된 다른 클라이언트의 식별번호와 포트번호를 통해 새로운 UDP 연결 요청
	서버 : 클라이언트의 식별번호를 키값으로 각 클라이언트에게 영상 전송
영상 채팅	서버 : AI를 통해 실시간으로 수신되는 영상의 감정 인식 결과를 전송 영상에 포함
	클라이언트 : 다른 클라이언트의 영상 및 감정 인식 결과를 사용자가 볼 수 있도록 적절하게 출력

4. 상세 설계

- UI

제작의 용이성 및 확장성을 위해 WPF(C#)을 사용하여 제작하며, 최소 두 개의 윈도우 구현이 필요된다.

1. 대기 화면

- 채팅이 시작되기 전 윈도우로, 방 생성 및 방 참여 기능을 포함한다.
- 현재 개설된 방 목록이 출력되며 목록에 존재하는 방을 선택해서 참여하면 해당 채팅방으로 입장 되며, 대기 화면의 윈도우는 종료되고 채팅 화면 윈도우가 생성된다.
- 방 생성 시, 생성할 방의 제목과 비밀번호(선택)를 설정할 수 있으며, 주어진 감정상태에 따라서 이벤트를 설정할 수 있다.
- 방을 생성하게 되면, 방의 생성자는 자동으로 방에 입장되며, 해당 방은 방목록에 추가되어 다른 사용자가 참여할 수 있게 된다.
- [방 목록 리스트], [참여 버튼], [방 생성 버튼] 이 필요하며, 참여 버튼을 직접 누르지 않고 목록에 방을 더블클릭을 통해 참여할 수 있도록 구현한다.

2. 채팅 화면

- 채팅이 이루어지는 윈도우로, 실시간 채팅 및 영상 송수신이 기능이 포함된다.
- 입력 텍스트 박스에 채팅할 내용을 적고 전송할 시 서버로 작성한 내용이 전달되며, 서버로부터 전송된 대화내용은 출력 텍스트 박스에 출력된다.
- 자신을 포함한 채팅방의 참여된 모든 사용자의 실시간 영상이 이미지 박스에 출력되며, 서버의 감정인식 결과 값에 따라 채팅방에 지정된 설정에 맞는 이벤트가 처리된다.
- 나가기 버튼을 클릭시 해당 채팅방에서 윈도우가 종료되고 대기 화면 윈도우가 다시 생성된다.
- [채팅 박스], [메시지 입력 박스], [전송 버튼], [나가기 버튼], [각 사용자의 이미지 박스]가 필요하며, 전송 버튼을 누르지 않고, 엔터를 누르면 작성된 내용이 서버로 전달되도록 한다.

- SERVER / PROTOCOL

서버 프로그램이 시작되면 각 클라이언트가 접근할 수 있도록 기본으로 9000번 포트를 TCP 방식으로 개방한다. 연결된 클라이언트가 방을 생성하고 이에 대한 요청이 수신될 경우 해당 방을 리스트 형식으로 저장한다. 해당 리스트는 다른 클라이언트의 접속 혹은 목록 요청이 발생했을 때 제공된다.

방이 생성될 경우 해당 방을 관리하는 프로세스를 생성하며 해당 프로세스가 관리할 수 있는 포트 영역대를 할당해 준다. 방을 관리하는 프로세스는 할당받은 포트 번호로 클라이언트가 연결할 수 있도록 TCP 방식으로 개방하며, 해당 포트로 클라이언트의 연결 관리 및 시그널을 보내는데 사용하고 또한, 클라이언트의 채팅 내용을 송·수신하는데 사용한다. 동시에 연결되는 클라이언트를 효율적으로, 비동기적으로 관리하기 위해 SELELCT 형식으로 소켓을 관리한다. 하나의 소켓으로 채팅과 여러 시그널을 동시에 사용하기 때문에, 메시지가 혼동될 수 있으므로 메시지의 맨 앞 글자를 구분자로 메시지를 구분한다.

! : 채팅 메시지를 의미하는 구분자로, 내용은 전부 채팅 메시지
@ : 새 연결을 요청하는 구분자로, 내용은 요청하는 클라이언트의 포트 번호
# : 클라이언트의 연결을 알리는 구분자로, 내용은 클라이언트의 포트와 아이디
$: 새로 연결된 클라이언트의 식별을 아리는 구분자로, #과 같은 내용

클라이언트가 새로 연결될 경우 해당 클라이언트가 서버로 영상을 송신할 수 있도록 개방할 포트번호를 할당해주며 새로운 쓰레드를 생성하여 해당 포트를 UDP 방식으로 개방한다. UDP 방식으로 개방된 포트를 통해 클라이언트의 영상을 전송받는다. 다른 클라이언트가 새로 연결될 경우 기존에 연결된 클라이언트와 새로 연결된 클라이언트가 서로 식별할 수 있도록 하며, 식별된 각 클라이언트가 서로의 영상을 수신받을 수 있도록 송신과 같은 방식으로 전용 포트를 할당하고 UDP 방식으로 개방한다. 전송받은 클라이언트의 영상을 각 클라이언트에게 실시간으로 전송해준다.

각 채팅방을 관리하는 프로세스가 실행될 때, 감정인식 프로그램을 실행시키고 해당 감정인식 프로그램과 파일 통신 방식으로 연결한다. 감정인식 프로그램에 클라이언트의 영상을 전송하여 감정을 인식하도록 하며, 감정인식 결과를 받아 각 클라이언트에게 전송한다.

- 얼굴 검출 모듈 / 감정 인식 모듈

감정인식 모듈은 이미지에서 특징을 추출하는 과정에서 크게 2가지로 나뉜다.

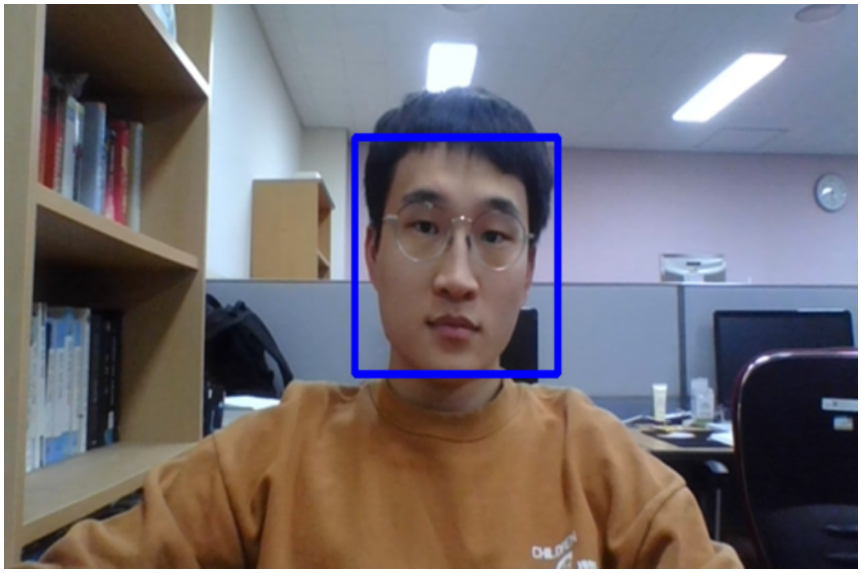
1. Opencv Harr Cascade

이미지에서 얼굴 영역을 검출한 후 영역전체에 대한 패턴을 분석하여 감정을 예측하는 방식이다. CNN 알고리즘 사용한다.

2. dlib 68 landmarks

이미지에서 얼굴 검출 후 외각, 눈, 입 등 68가지 점을 찍어 특징을 검출하는 방식이다. SVM 혹은 Random Forest 알고리즘 사용한다.

* Opencv - Haarcascade 방식



Harr Cascade 는 머신러닝 기반의 오브젝트 검출 알고리즘이다. 2001년 논문 “Rapid Object Detection using a Boosted Cascade of Simple Features” 에서 Paul Viola와 Michael Jones가 제안한 특징(feature)을 기반으로 비디오 또는 이미지에서 오브젝트를 검출하기 위해 사용된다.

본 프로젝트에서는 얼굴이 포함된 이미지와 얼굴이 없는 이미지를 사용하여 미리 학습되어 있는 Haar Cascade Classifier(하르 특징 분류기)를 사용하여 얼굴을 검출한다. 입력으로 grayscale 되어있는 이미지를 넣으면 얼굴에 해당하는 좌표값이 반환되며 이미지에서 해당 좌표값에 포함되는 영역을 잘라내서 전처리 및 인공지능 모듈에 적용한다.

얼굴 검출에서 Haar Cascade 방식을 사용하여 영역을 얻은 방식을 사용한다면, 인공지능 모듈에서는 얼굴 영역 전체에 대한 패턴을 예측하기 위해 CNN 알고리즘을 사용한다.

* Dlib - 68_face_landmarks 방식

사진은 얼굴의 68개 특징에 점을 찍은 모습이다. 미리 학습 후 공개되어 있는 shape_predictor_68_face_landmarks 모델을 사용하였다. 입력으로 graysclae 되어 있는 이미지를 넣으면 얼굴 검출 후 68개 특징점의 좌표가 반환된다. 인공지능에 적용하기 위해서는 해당 점에서 특정 지점 사이의 거리, 혹은 특정 지점의 변화량 등을 분류를 위한 특징으로 구하여 감정을 예측한다.

위와 같은 방식을 사용하여 특징을 얻는다면, 인공지능 모듈에서는 비선형적인 데이터의 구분을 위해 SVM(Support Vector Mechine) 혹은 Random Forest 알고리즘을 사용한다.

그러나, 이 방식을 사용하기 위해서는 다음과 같은 한계점들이 존재한다.

1. 뽑아낸 특징이 각각의 감정에 얼마나 영향을 미치는 지 알 수 없다.
2. 사람마다 특징값의 차이가 크기 때문에, 정확한 감정예측이 힘들다.

해결책으로는 각 사용자의 무표정을 서버에 저장한 후, 얼굴 표정에 따라 변화하는 특징들의 패턴을 분석하여 감정을 예측하는 방법이 있으나, 사용자들의 이미지를 모두 서버가 가지고 있어야 하며 사용자가 추가될 때마다 인공지능 학습시간이 추가로 요구되는 등 많은 에러사항이 존재한다. 따라서, 본 프로젝트에서는 Haar Cascade Classifier를 사용하여 얼굴 검출 후 CNN 알고리즘을 이용한다.

- 학습 데이터

본 프로젝트에서는 caggle 에서 다운받은 28,709개 Training Set + 7,181개의 Test Set 의 감정 결과가 Labeling 된 이미지 데이터를 사용한다. 7,181개의 Test

set 은 모델 최적화에 사용될 3,590 개의 Validation Set 과 최종 모델 정확도 확인에 사용될 3,591 개의 Test Set 으로 분할하여 사용한다. 이미지 파일은 csv 형태로 제작되어 있으며, 정확도 및 속도 개선 등 최적화를 위해서 데이터를 추가하거나 변형하는 작업을 수행한다.

- 인공지능 모델 제작

감정인식을 위해 사전에 모델을 제작해야 한다. tensorflow 기반으로 최적화 및 더 높은 수준의 API를 제공하기 위해 설계된 tflearn 라이브러리를 사용하여 모델을 제작한다.
모델 제작은 다음의 단계를 거친다.

1. csv 파일로 저장된 데이터를 numpy의 배열로 읽어온다.
2. 감정인식에 사용될 형태로 이미지를 변형한다.
3. 평균, 표준편차 등을 이용한 전처리를 수행한다.
4. 데이터를 뒤집거나 늘리는 등의 데이터 증강을 수행한다.
5. Input 데이터 형태 지정, 활성화 함수 등 Hyper Parameter 를 지정하고 Network 구조를 만든다.
6. 만들어진 Network 를 최적화한다.
7. 최적화된 Network 로 Model을 만들고 학습시킨다.
8. 모델을 평가한 후, 만족스러운 결과가 나오면 저장한다. tflearn 라이브러리를 사용할 경우 checkpoint, DATA-00000-OF-00001, INDEX, META 4가지 파일이 저장된다.

- 감정인식

감정인식은 이미지에서 검출된 얼굴 데이터를 이용하여 진행한다. 만들어진 인공지능 모델을 불러와 grayscale, reshape 등 필수적인 전처리 후 예측을 수행한다. 알 수 있는 감정은 Angry(화남), Disgust(역겨움), Fear(공포), Happy(행복), Sad(슬픔), Surprise(놀람), Neutral(무표정) 의 7가지이며, predict 함수를 통하여 각 감정에 대한 확률(총 합이 1 이 되는 정규화 된 값)을 얻는다. 7가지 값 중 가장 높은 확률을 가진 값을 결과값으로 정한다. 이 때, 따로 데이터를 저장하여 특정 기준을 넘는 경우에만 표시하거나 전체적으로 고르게 분포된 결과값이 나와 제대로된 인식이 안되는 경우가 나올 수 있어 예측 결과값을 따로 저장, 관리한다.

저장된 데이터는 데이터 증강, 전처리 및 Model 수정 등을 통해 정확도를 개선하는 사전 데이터로 사용한다.

- 데이터 추가

학습 데이터가 부족할 경우 다음의 방법으로 데이터를 추가한다.

1. 만들어진 Model을 이용한다. 영화 및 동영상에서 감정인식 모듈을 실행하여 정확도가 특정 기준을 넘어갈 경우 데이터를 저장한다. 저장 기준을 특정 감정에 대하여 90%의 높은 기준으로 하여 에러 데이터를 줄인다.

2. 직접 데이터를 구한다. 구글 등 공개되어있는 사진에서 눈으로보고 감정을 분류하고 파일로 저장한다. 이 경우 전처리를 수행하는 모듈을 추가로 구현한다.

3. 사용자 개인의 사진을 학습 데이터에 추가한다. 프로그램에서 카메라 캡처 기능을 추가한 후, 특정 감정을 제시하고 사용자가 그 감정의 표정을 짓고 직접 캡처를 할 수 있게 하여 데이터를 추가한다.

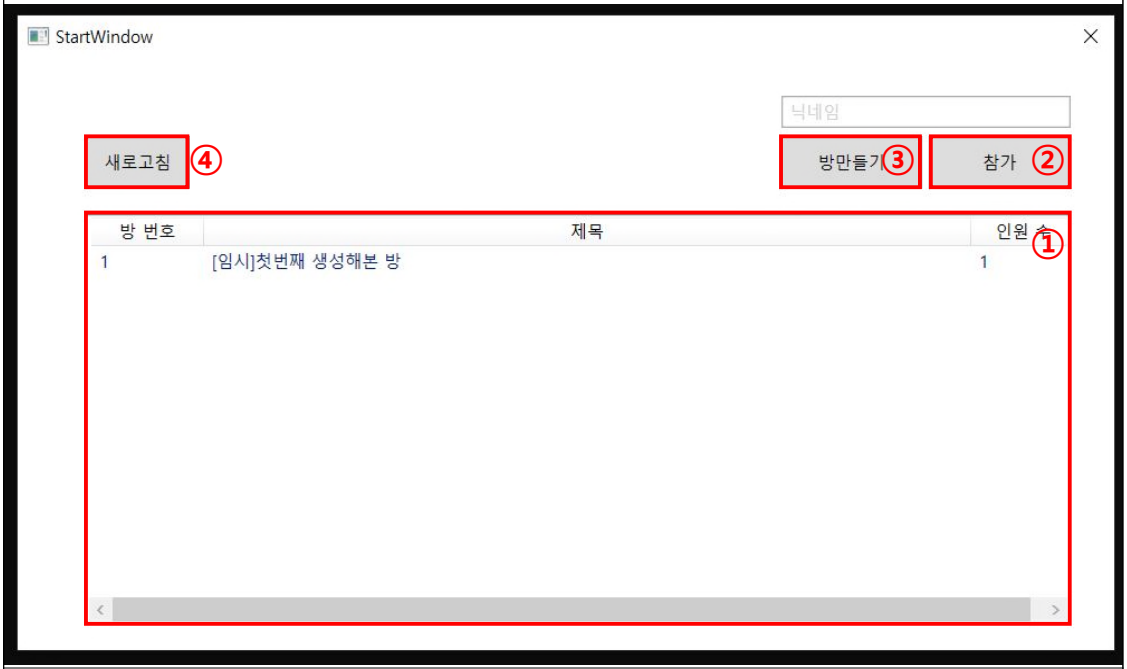
- 모듈 통합

본 프로젝트는 크게 채팅, 얼굴 검출, 감정 인식 3가지 모듈이 존재한다. 하나의 프로그램으로 만들기위해 먼저 이미지에서 얼굴을 검출하여 필요한 전처리 작업을 거치는 얼굴 검출 모듈과 감정 인식 결과를 반환하는 감정 인식 모듈을 인공지능 모듈로 통합한다. 그 후, 인공지능 모듈을 서버에서 호출하여 반환받은 감정값을 각 클라이언트에게 전달하며, 감정값에 따라 클라이언트에서는 정해진 이벤트를 처리한다.

고려할 점은, Python 라이브러리를 사용하여 제작된 인공지능 모듈을 다른 언어로 제작된 서버에서 호출하여야 하는데, 속도 등 동기화 문제를 해결해야 한다. 또한 전송, 출력 등 이미지가 사용되는 곳에서 mat, char array, numpy array 로 이미지의 형태가 변화되는데, 데이터의 손실여부 등 발생가능한 에러사항에 대하여 주의해야 한다.

5. 프로토 타입

(1) 대기화면



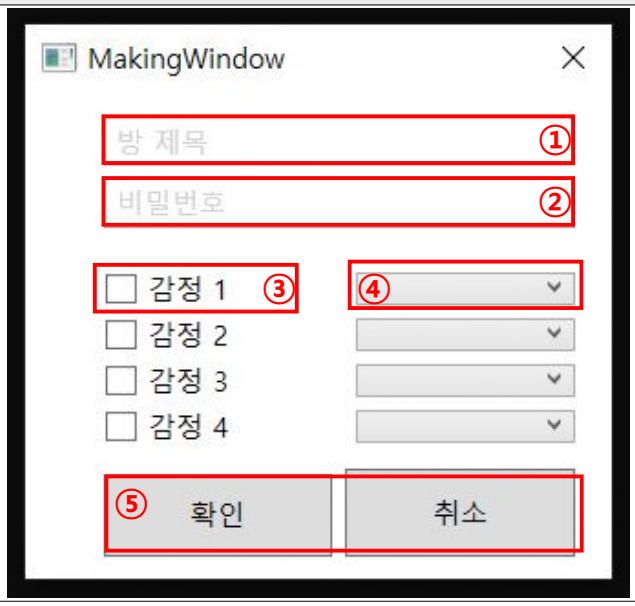
① 현재 존재하는 방 목록이 표시될 리스트 뷰로, 서버에 연결될 경우 혹은 새로고침 요청을 하게 될 경우 서버로부터 방 목록 리스트를 전달받아 출력한다. 출력되는 방 목록들 중 특정 방을 선택하여 입장할 수 있으며, 리스트를 더블 클릭해도 해당 방에 입장할 수 있다. 방에 입장할 경우 해당 윈도우를 종료시키고 채팅 화면 윈도우를 생성한다.

② 리스트에서 현재 선택된 방에 입장한다. 리스트에 어떠한 방도 선택되지 않은 경우 활성화되지 않는다. 방에 입장할 경우 해당 윈도우를 종료시키고 채팅 화면 윈도우를 생성한다.

③ 새로운 방을 만드려고 할 때 사용하는 버튼으로 방만들기 윈도우를 생성한다.

④ 새로고침 버튼으로, 서버로부터 리스트를 제공하도록 요청한다.

(2) 방 만들기



① 방 제목 입력란으로 입력된 제목은 대기화면 리스트에 표시될 내용으로, 방 제목을 보고 다른 사용자가 입장할 수 있다.

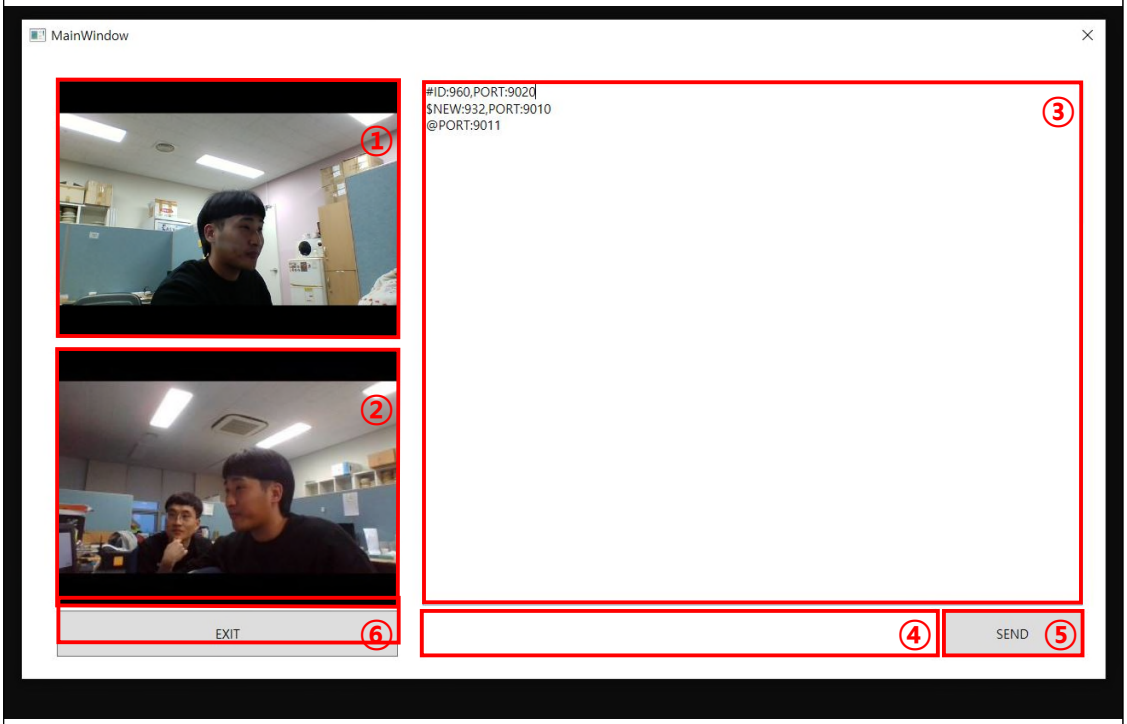
② 방 비밀번호 설정 란으로 입력시 '*' '●' 와 같은 특수기호로 표시되며 비밀번호가 설정된 방에 입장하려는 경우 설정된 비밀번호와 일치하는 비밀번호를 입력해야한다.

③ 감정 선택란(예정)으로 감정(화, 역겨움, 공포, 행복, 슬픔, 놀람, 무표정)을 선택하여 해당 감정에 이벤트를 설정할 수 있으며, 감정을 선택하지 않음으로써 이벤트를 발생시키지 않을 수 있다.

④ 이벤트 선택란으로 생성된 방 내부에서 해당 감정이 발생할 경우 발생할 이벤트를 설정한다.

⑤ 확인 버튼을 클릭 시 기입한 내용을 토대로 서버로 방 생성을 요청하며 요청 결과에 따라서 즉시 해당 방으로 입장한다. 대기화면 윈도우는 종료시키고 채팅화면 윈도우를 생성한다. 취소 버튼을 클릭 시 해당 윈도우를 종료시킨다.

(3) 채팅 화면



① 현재 사용자(자신)의 이미지로 실시간으로 출력되며, (예정)감정상태에 따라서 이벤트가 발생되어 출력된다.

② 다른 사용자의 이미지로 ①과 같음.

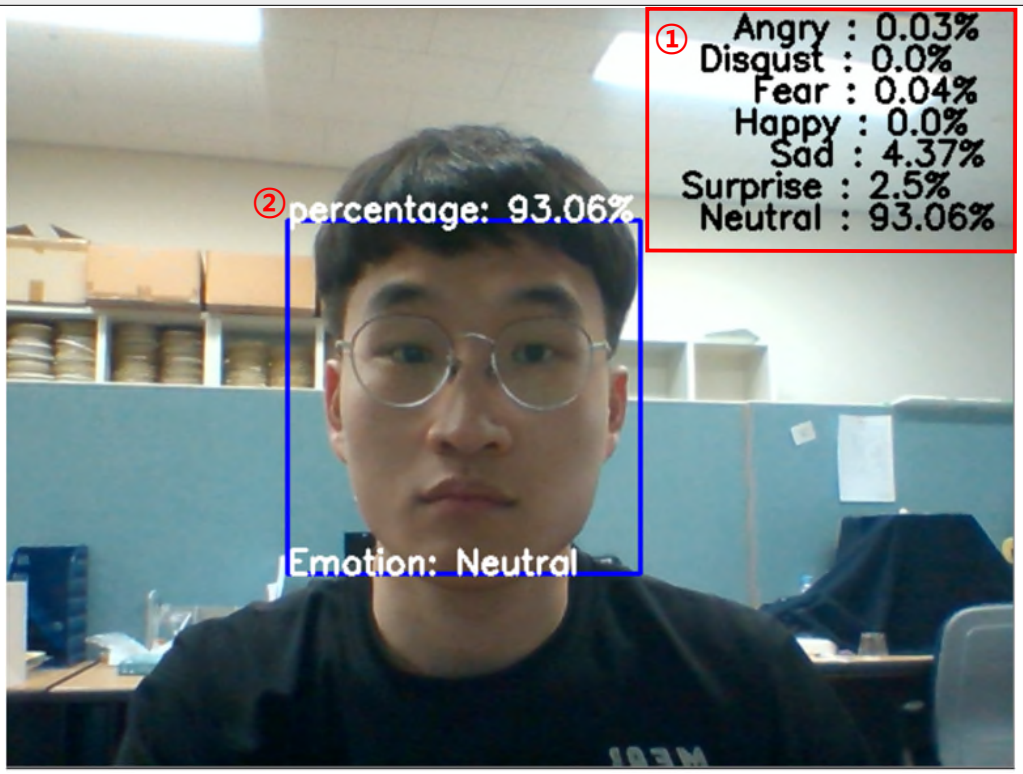
③ 메시지 리스트 뷰로, 채팅 방에 접속한 사용자 간에 주고받은 채팅 메시지가 기록된다. (시스템 메시지는 실제로 출력되지 않는다.)

④ 보내려고 하는 채팅 메시지를 적는 텍스트 박스로 메시지를 적고 전송버튼을 눌러서 메시지를 전송할 수도 있지만, 내용을 적고 엔터를 쳐서 전송시킬 수 있다.

⑤ 메시지 전송버튼으로 텍스트 박스에 있는 메시지를 서버로 전송한다.

⑥ 종료 버튼으로 현재 채팅방에서 나간다. 채팅화면 윈도우를 종료시키고 대기화면 윈도우를 호출한다.

(4) 얼굴검출 및 감정인식



위 사진은 얼굴검출 및 감정인식 모듈을 실행한 사진이다. 실제 프로그램에서는 표시된 결과값들만 이용되며, 인식 성능 확인을 위해 제작하였다.

① 감정인식 모듈은 7가지 감정에 대한 결과값을 반환한다. 값들의 합은 1이며, 가독성을 위해 퍼센트 형태로 100을 곱한 후, 소수점 2자리까지 출력하였으며, 가장 높은 값이 최종적인 감정 인식 결과로 판단된다. 모든 결과값들은 후에 감정에 대한 이벤트 처리 시에 사용된다.

② 얼굴검출 모듈은 얼굴에 해당하는 부분의 좌표를 4가지 변수(좌측 위, 우측 아래 좌표, (x,y),(w,h))로 반환한다. 사진의 파란색 테두리는 반환된 좌표값을 이용하여 얼굴에 해당하는 부분을 그린 것이다. 또한, 사진에서 좌표에 해당하는 부분만을 잘라내어 감정인식 모듈의 입력에 사용한다.

6. 구현 알고리즘

- 통신

(1) 서버 - 클라이언트

하나의 클라이언트에서 서버로의 연결은 비동기적 처리가 필요로 하기때문에 단일 연결이 아닌 다중연결을 구현하였다. 따라서 각 클라이언트의 다중연결을 관리하기 위해서 클라이언트와 서버를 TCP 연결을 통하여 다중연결을 관리하였으며, TCP 연결들을 관리하기 위해서 서버에서는 SELECT 방식을 통해 TCP 연결들을 관리하였다.

클라이언트에서 서버로 이미지를 전송할 때, 실시간으로 지속적으로 데이터가 전송되며 통신 과정에서 발생하는 데이터 손실 등의 문제를 최소화, 무시하기 위해서 UDP 연결을 통해 데이터를 전송한다. 실시간으로 비동기적으로 전송되는 이미지 데이터와 다르게 각 클라이언트의 연결을 관리하거나 이벤트 처리를 하기 위한 제어 데이터는 비동기적이나 실시간으로 지속적으로 데이터가 발생하지 않고 발생 되는 데이터에도 발생하는 시간 차이가 무시할 수 있을 만큼 미묘하기 때문에 기존의 TCP 연결을 통해 처리하였다.

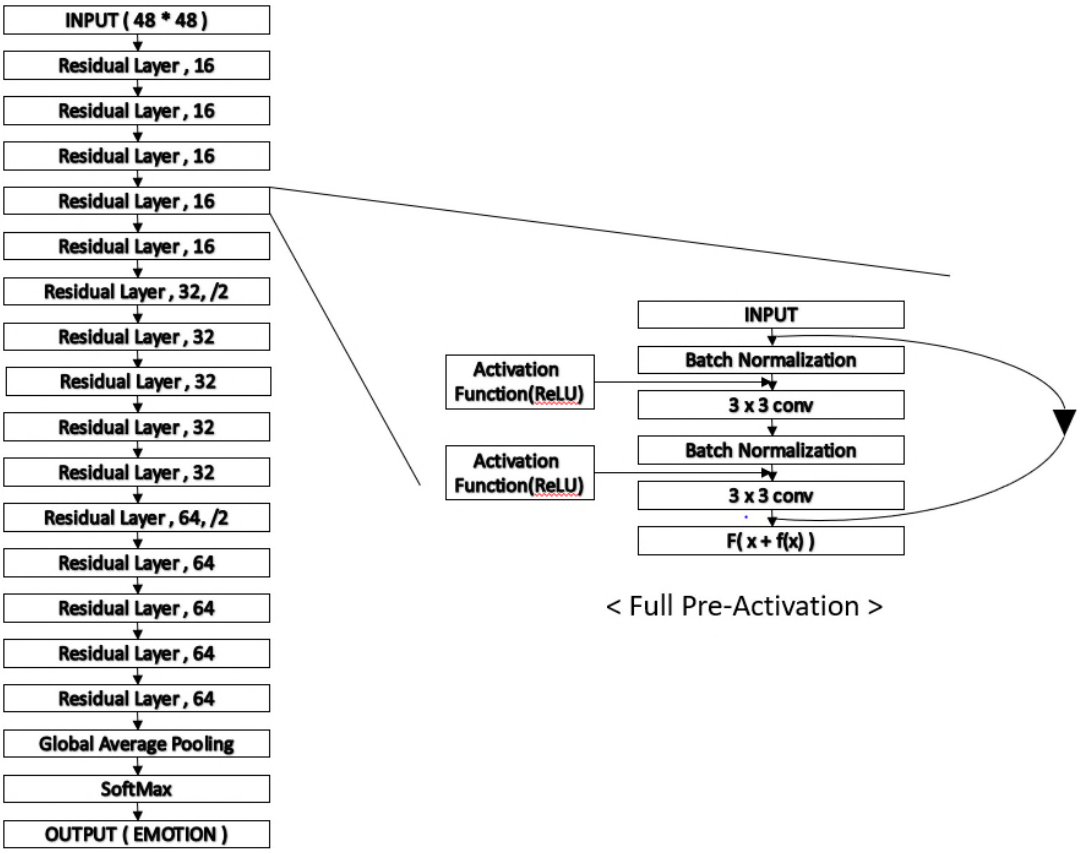
(2) 서버 - 인공지능

모듈 제작의 난이도, 라이브러리 보편화 등의 문제로 각 모듈의 제작 환경이 달라져서 클라이언트는 C#, 서버는 C++, 인공지능은 PYTHON으로 제작하였다. 클라이언트 프로그램의 필요 성능을 고려하여 인공지능은 서버 컴퓨터에서만 동작하기 때문에 서버(C++)와 인공지능(PYTHON) 간에 데이터 교환이 이루어져야 한다. 서버와 인공지능은 같은 컴퓨터에서 동작하기 때문에 IPC 통신기법을 사용하였으며, 서버와 인공지능 간의 데이터 교환이 비교적 실시간으로 이루어져야 하기 때문에 일련의 처리 속도가 매우 중요해진다. 따라서 다양한 IPC 기법 중 속도가 가장 빠른 공유 메모리(Shared Memory) 기법을 통해 두 프로세스를 연결하였다.

인공지능의 예측값을 실시간으로 전송할 경우 이벤트가 실시간으로 발생하여 오히려 이벤트를 알아볼 수 없는 상황이 발생한다. 따라서 발생한 이벤트를 사용자가 인지할 수 있도록 일정 시간 동안 지연시키며 사용자들의 이벤트를 동시에 처리하기 위해서 TCP 통신방식과 같이 메모리 맵 데이터에 trigger 데이터를 추가하였다. 또한 서버-클라이언트의 연결과 마찬가지로 제어 데이터를 저장할 메모리와 이미지 데이터를 저장할 메모리를 구분하여 처리하였다.

- AI

인공지능 모델의 네트워크 구조는 아래 그림과 같다.



- ResNet (Residual Network)

CNN 의 구조 중 하나로 하나의 Block 을 편의상 Layer 라고 표현하였다.

하나의 residual layer 는 우측의 그림과 같은 구조를 가진다. 성능 개선을 위해 Full Pre-activation 구조를 사용하였으며, Input 값을 계산된 결과가 합하여 최종적으로 출력을 하는데, 이는 여러 가지 장점을 가진다. 첫 번째로, 신경망이 깊어질 때 발생하는 Gradient Vanishing / Exploding 혹은 Degradation 을 해결할 수 있다. 두 번째로, 학습의 목표가 input 이 더해진 F(x + f(x)) 를 최소화 하는 방향이기 때문에 학습 시 사용되는 Forward / Backward 연산이 단순해진다. 또한, 이런 장점을 가지기 위해서 사용된 skip-connection 을 덧셈 연산 한번으로 구현 가능하기 때문에, 연산 횟수에 크게 영향이 없다.

본 프로젝트에서 사용된 전체 네트워크 구조는 왼쪽과 같다. Layer 를 지나며 피처맵의 크기가 반으로 작아지는 경우 Layer 연산량 보존을 위해 출력맵의 개수를 2배로 하며, 복잡도를 줄이기 위해 pooling, dropout 연산은 사용하지 않았다.

- Data Preprocessing & Augmentation

학습에 사용하는 데이터는 다음과 같은 방법으로 전처리 과정을 수행한다.

1. 모든 샘플의 중심을 0 으로 설정
2. 표준편차에 따라 각 표본의 크기를 수정
3. 랜덤한 확률로 이미지 좌/우 반전
4. 랜덤한 확률로 가우시안 필터 적용 - 이미지 블러링 효과
5. 랜덤한 확률로 랜덤한 각도 이미지 회전

- Hyper Parameter Optimization

최적화를 통해 결정된 값은 다음과 같다.

Epochs : 200 / Learning rate : 0.01 / batch size : 128 /
Activation Function : ReLU / Optimizer : Adam

결정 요인 1. 학습 횟수를 적당히 늘릴수록 정확도가 높아졌다.

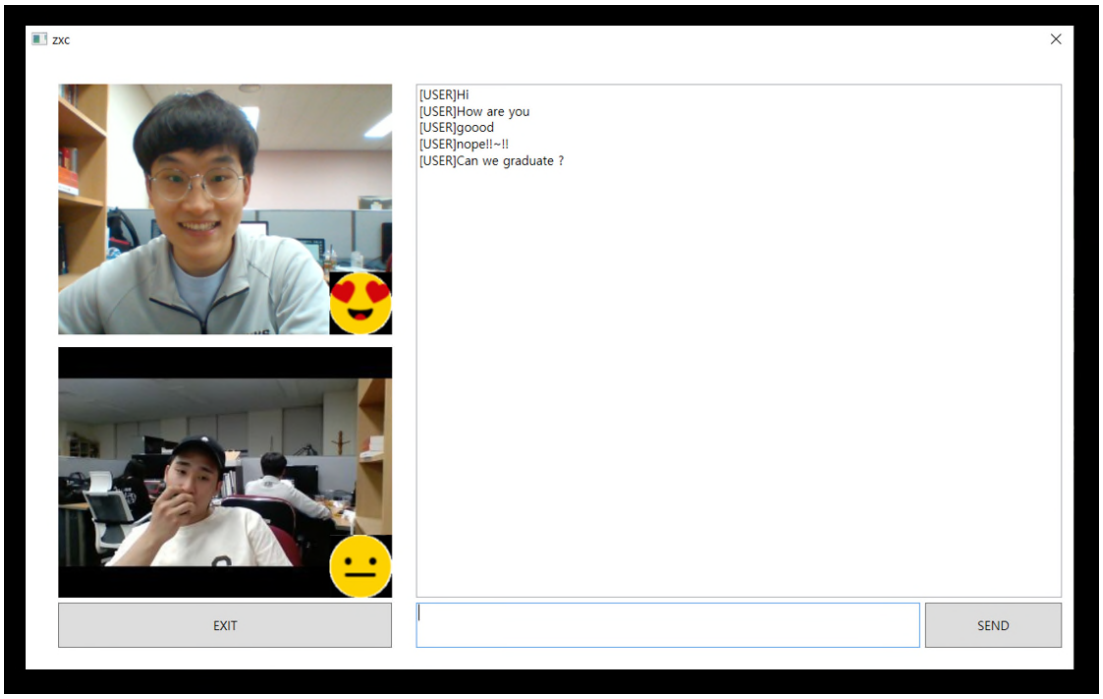
결정 요인 2. 학습 횟수를 늘릴수록 Overfitting 현상이 발생하였다. 적당한 수준의 학습 횟수의 증가는 불가피한 요소이므로, Overfitting 을 방지하기 위하여 L2 Regularization 를 적용하였다.

결정 요인 3. 너무 낮은 Learning Rate 는 optimizer 에 상관없이 global 한 최적값을 찾지 못하였다.

결정 요인 4. 정확도와 속도 문제를 동시에 개선할 만한 Batch size 는 128 이였다.

7. 시험 / 테스트 결과

- 채팅 어플리케이션



문제점

- 서버-클라이언트 통신 불안정
- 최초 연결까지 발생 되는 너무 긴 지연시간
- PC 환경에 따른 예외 처리 미흡
- 모듈 간 통합에서 예외 처리 미흡, 느린 속도

수정 방향

- 서버 안정화. 이미지 데이터 전송 방법 수정
- 다양한 PC 환경에서 동작 테스트
- 일반화된 로직으로 수정
- UI 디자인 수정

- **모델 제작 결과**

  평균 학습 시간 : 10시간

  최종 정확도 : 63.4%

문제점

- AI 의 낮은 정확도
- 하나의 모델 제작에 너무 긴 시간이 요구됨

개선 방향

- 정확도 개선을 위한 더 깊은 신경망 구조 구현
- 다양한 Hyper Parameter 수정을 통한 최적화 수행
- 분산 학습 등의 학습시간 개선을 위한 방법 적용

Ⅲ. 결론

본 연구의 주목적은 맞춤형 서비스를 제공하기 위해 사람의 감정을 활용할 수 있도록 감정인식 인공지능을 구현하는 것이다. 감정을 인식하는 방법 중 가장 대표적인 요소로 표정을 선택했고 이를 기반으로 연구를 진행했다.

표정을 7가지 기준으로 분류하였으나 헛웃음, 비웃음과 같이 특정한 감정에서 나오는 사람의 모순된 표정은 시각적으로 분류하기에 어려움이 존재했다. "일차원적인" 감정에서 나오는 표정은 근접하게 인식할 수 있으나 "복합적인" 감정에서 나오는 표정의 인식은 높은 확률의 정확도를 기대하기 어렵다.

사람은 일상생활에서 "일차원적인" 감정보다 "복합적인" 감정을 더 많이 느끼기 때문에 표정만으로 감정을 파악하기에는 한계가 존재한다. 표정뿐만 아니라 목소리 등의 다른 요소들을 함께 사용한다면 감정의 판단에 더 높은 정확도를 얻을 수 있을 것으로 기대된다.

Github : https://github.com/tojbabo/KEUM_TAK_PROJECT

참고 자료

- Paul Viola, Michael Jones (2001). “Rapid Object Detection using a Boosted Cascade of Simple Features”

- pyimagesearch, "Facial landmarks with dlib, OpenCV, and Python", https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun(2016) Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) “Deep Residual Learning for Image Recognition”

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun(2016). “Identity Mappings in Deep Residual Networks”

- 학습 데이터 셋 : https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge